



@server

IBM Systems - iSeries
Programming - Qshell

Version 5 Release 4





@server

IBM Systems - iSeries
Programming - Qshell

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices."

Ninth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Qshell	1	dot (.) - Run commands in current environment	52
What's new.	2	env - Set environment for command invocation	52
How to see what's new or changed.	2	eval - Construct command by concatenating arguments	53
Printable PDF	2	exec - Run commands and open, close, or copy descriptors	53
Tutorial	3	exit - Exit from the shell	54
Qshell command language features	3	help - Display information for built-in utility	55
Qshell utility features	5	nohup - Run utility without hangups.	55
Putting it all together in a script	7	qsh - Qshell command language interpreter	56
Command language	8	rexc - Run remote command	57
Quoting	9	rexx - Run REXX procedure	58
Parameters	9	source - Run commands in current environment	59
Variables	10	system - Run CL command	59
Variables set by qsh	10	type - Find type of command	61
Variables used by qsh	12	whence - Determine how command is interpreted	62
Other variables	14	xargs - Construct argument lists and invoke utility	62
Word expansions	15	Utilities for managing data	64
Tilde expansions.	15	cmp - Compare two files	64
Parameter expansions	15	cut - Cut out selected fields of each line of a file	65
Command substitutions	17	egrep - Search a file for an extended regular expression pattern	66
Arithmetic expansions.	18	fgrep - Search a file for a fixed string pattern	66
Field splitting.	20	grep - Search a file for a pattern	66
Path name expansion	20	iconv - Convert characters from one CCSID to another CCSID	68
Quote removal	20	sed - Stream editor	69
Patterns	20	sort - Sort, merge, or sequence check text files	74
Redirection	21	split - Split files into pieces	75
Simple commands	22	tr - Translate characters	76
Pipelines	23	uniq - Report or filter out repeated lines in a file	78
Lists.	23	wc - Word, line and byte/character count	79
Compound commands	24	Utilities for DB2 Universal Database	80
Grouping commands	25	Utilities for working with files and directories.	80
If command	25	attr - Get or set attributes for files	81
Conditional command.	26	basename - Return non-directory portion of path name.	86
Case command	26	cat - Concatenate and print files	86
Select command	27	catplf - Concatenate and print spool files	87
While command	27	cd - Change working directory	88
Until command	27	chgrp - Change file group ownership.	89
For command	28	chmod - Change file modes	90
Functions	28	chown - Change file ownership.	93
Using Qshell	29	compress - Compress data	94
Using a Qshell interactive session	29	cp - Copy files	95
Running Qshell commands from CL	31	dirname - Return directory portion of path name	97
Running Qshell commands from PASE	32	file - Determine file type	97
Customizing your environment.	32	find - Find files	98
National language support (NLS) considerations	32		
Performance considerations	36		
Developing your own utilities	36		
Editing files with Qshell Interpreter	37		
Differences with other interpreters.	37		
Utilities.	38		
Utilities for defining aliases	49		
alias - Define or display aliases.	49		
unalias - Remove alias definitions	50		
Utilities for running commands.	50		
builtin - Run a shell built-in utility	51		
command - Run a simple command	51		

gencat - Generate a formatted message	
catalog	101
getconf - Get configuration values	102
head - Copy the first part of files	104
ln - Link files	104
ls - List directory contents	105
mkdir - Make directories	108
mkfifo - Make FIFO special files	109
mv - Move files.	110
od - Dump files in various formats	110
pax - Portable archive interchange	112
pr - Print files	119
pwd - Return working directory name	121
pwdx - Print working directory expanded	121
Rfile - Read or write record files	122
rm - Remove directory entries.	123
rmdir - Remove directories.	124
setccsid - Set CCSID attribute for file	125
tail - Display the last part of a file	125
tar - File archiver	126
touch - Change file access and modification	
times	128
umask - Get or set the file mode creation	
mask	129
uncompress - Expand compressed data.	130
zcat - Expand and concatenate data	131
Utilities for reading and writing input and	
output.	131
dspmsg - Display message from message	
catalog	131
echo - Write arguments to standard output	132
print - Write output	133
printf - Write formatted output	133
read - Read a line from standard input	135
Utilities for developing Java [™] programs	135
ajar - Alternative Java ^(™) archive	136
appletviewer - View Java [™] applet	139
extcheck - A utility to detect JAR conflicts	139
jar - Archive Java [™] files.	139
Files in the integrated file system.	139
jarsigner - JAR signing and verification.	140
java - Run Java [™] interpreter	140
javac - Compile a Java [™] program	141
javadoc - Generate Java [™] documentation	141
javah - Generate C header or stub file	141
javakey - Manage Java [™] security keys and	
certificates	142
javap - Disassemble a compiled Java [™]	
program	142
keytool - Key and certificate management	
tool.	142
native2ascii - Convert native characters to	
ASCII	143
policytool - Policy file creation and	
management tool	143
rmic - Compile Java [™] RMI stubs.	143
rmid - The Java [™] RMI activation system	143
rmiregistry - Start a remote object registry	143
serialver - Return serial version	143
tnameserv - Naming service	144
Utilities for managing jobs	144
getjobid - Display job information	144
hash - Remember or report utility locations	145
jobs - Display status of jobs in current	
session	146
kill - Terminate or signal processes	147
liblist - Manage library list	148
ps - Display process status	149
sleep - Suspend invocation for an interval	151
trap - Trap signals.	151
wait - Wait for process completion	152
Utilities for Kerberos credentials and key tables	153
Utilities for LDAP directory server	153
Utilities for working with parameters and	
variables	154
declare - Declare variables and set attributes	154
export - Set export attribute for variables	155
local - Assign a local variable in a function	156
printenv - Display values of environment	
variables	157
readonly - Set read-only attribute for	
variables	157
set - Set or unset options and positional	
parameters	158
shift - Shift positional parameters.	160
typeset - Declare variables and set attributes	160
unset - Unset values of variables and	
functions	161
Utilities for writing scripts	162
break - Exit from for, while, or until loop	162
colon (;) - Null utility.	162
continue - Continue for, while, or until loop	163
false - Return false value	163
getopts - Parse utility options	163
let - Evaluate arithmetic expression	164
return - Return from a function	164
test - Evaluate expression	165
true - Return true value	167
Miscellaneous utilities	168
clrtmp - Clear the /tmp directory	168
dataq - Send or receive messages from	
i5/OS ^(™) data queue	169
datarea - Read or write i5/OS ^(™) data area	170
date - Write the date and time.	170
expr - Evaluate arguments as an expression	172
hostname - Display the name of the current	
host system	173
id - Return user identity.	173
ipcrm - Remove interprocess communication	
identifier	174
ipcs - Report interprocess communication	
status	175
locale - Get locale specific information	180
logger - Log messages	181
logname - Display user's login name	182
sysval - Retrieve system values or network	
attributes.	182
tee - Duplicate standard input.	182
ulimit - Set or display resource limits	183
uname - Return system name	184
Application Programming Interfaces.	185
QzshSystem() - Run a QSH Command	185

QzshCheckShellCommand() - Find QSH	
Command	187
Examples: Using a remote client that connects to a	
qsh session	188
Example: Server program	189
Example: Client program	196
Example: Creating and running the server	
program	205
Example: Creating and running the client	
program	207

Appendix. Notices	209
Programming Interface Information	211
Trademarks	211
Terms and conditions.	211
Code license and disclaimer information	212

Qshell

Qshell is a command environment based on POSIX and X/Open standards. It consists of two parts:

- The shell interpreter (or **qsh**) is a program that reads commands from an input source, interprets each command, and then runs the command using the services of the operating system.
- The utilities (or commands) are external programs that provide additional functions and can be quite simple or very complex.

Together, the shell interpreter and utilities provide a powerful, standards-based scripting environment. As you use the new programming models offered by i5/OS^(TM), Qshell provides an extensible command environment that allows you to:

- Manage files in any file system supported by the Integrated File System.
- Run threaded programs that do thread-safe I/O to and from an interactive session.
- Write shell scripts that can be run without modification on other systems using a cross-platform command language.
- Write your own utilities to extend the functions provided by Qshell.

This topic provides both new and experienced users with the information needed to use Qshell commands and write Qshell scripts.

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 212.

“What’s new” on page 2

Select this link to learn about what is new in Qshell this release.

“Printable PDF” on page 2

Select this link to print a PDF of this topic.

“Tutorial” on page 3

Select this link to learn about using the Qshell command language and utilities. Start here if you are a new to using shells and shell commands.

“Command language” on page 8

Select this link to view the detailed reference information for the Qshell command language. Start here if you are writing shell scripts or are an experienced user of shells.

“Using Qshell” on page 29

Select this link to find out how to use the QSH CL command, how to configure the Qshell environment, and how to develop utilities.

“Utilities” on page 38

Select this link to view the list of utilities provided with Qshell.

“Application Programming Interfaces” on page 185

Select this link to view the list of application programming interfaces (APIs) provided with Qshell.

“Examples: Using a remote client that connects to a qsh session” on page 188

Select this link to view an example of a remote client and server for starting an interactive Qshell session.

Note: In this information, the terms "job" and "process" are used interchangeably. The term "job" is from i5/OS^(TM) and the term "process" is from POSIX.

What's new

The following changes were made in Qshell for this release:

"attr - Get or set attributes for files" on page 81
Added support for new attributes.

"cmp - Compare two files" on page 64
Added support for the QIBM_CMP_FILE_SIZE environment variable.



"ipcs - Report interprocess communication status" on page 175
Added support for -j option.

"rexec - Run remote command" on page 57
Added support for -i option.

"sed - Stream editor" on page 69
Added support for -C option.

How to see what's new or changed

To help you see where technical changes have been made, this information uses:




- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to users.

Printable PDF


To view or download the PDF versions of this topic, select Qshell Reference (about 215 pages).

You can also view or print any of the following PDFs:

- Manuals:
 - IBM^(R) Developer Kit for Java^(TM) 
 - IBM^(R) Toolbox for Java^(TM) 
- Redbook:
 - Building AS/400^(R) Internet-Based Applications with Java  (about 334 pages)

To save a PDF on your workstation for viewing and printing:

1. Open the PDF in your browser (click one of the links above).
2. In the menu of your browser, click **File**.
3. Click **Save As...**
4. Navigate to the directory in which you would like to save the PDF.
5. Click **Save**.

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site  .

Tutorial

This topic provides a tutorial to help you get started using the Qshell command language and utilities.

“Qshell command language features”

Select this link to view information about commands, input and output redirection, path name expansion, parameter expansion, and command substitution.

“Qshell utility features” on page 5

Select this link to view information about navigating in the Integrated File System and working with files and directories.

“Putting it all together in a script” on page 7

Select this link to view an example that shows how to write a shell script.

Qshell command language features

The shell interpreter can be used for either an interactive session or for writing shell scripts. A shell script is just a text file that contains shell commands. The Qshell command language is identical for either interactive use or for writing scripts. Any command that you run from an interactive command line can be put in a shell script and it runs the same way. The Qshell command language is interpreted so a shell script is read and interpreted each time it is run.

Commands

A “Simple commands” on page 22 is the name of a utility that you want to run. If you specify a fully-qualified path name to the command, for example “/usr/bin/ls”, **qsh** runs that command. If you specify a relative path name to the command, for example “ls”, **qsh** searches the directories specified by the PATH variable to find it. The PATH variable is a colon delimited list of directories that tells **qsh** where to find commands. If the PATH variable is set to

```
/usr/bin:/QOpenSys/usr/bin:
```

qsh first looks for the command in the “/usr/bin” directory, then in the “/QOpenSys/usr/bin” directory, and finally in the current working directory. When the PATH variable begins or ends with a colon or contains two adjacent colons, **qsh** searches in the current working directory.

By default, **qsh** waits for the command to complete before running the next command. When the command is completed, it sets an exit status that describes the result of the command. An exit status of zero means that the command was successful. An exit status that is greater than zero means that the command was unsuccessful. Typically, the exit status is one when a command fails. Although, **qsh** sets the exit status to 126 when the command was found but could not be run and sets the exit status to 127 when the command was not found.

The “Compound commands” on page 24 include the if-then-else conditional, [[...]] conditional, case conditional, select conditional, while loop, until loop, for loop, and functions. These commands provide the features you would expect in a high-level programming language and allow you to write complex shell scripts.

A “Pipelines” on page 23 allows you to chain several commands together so the output from one command is the input to the next command. For example, in the pipeline

```
ls | grep ^apple
```

the output from the “ls - List directory contents” on page 105 utility becomes the input to the “grep - Search a file for a pattern” on page 66 utility. The **ls** utility lists the contents of a directory and the **grep** utility searches for matches to a pattern. The final output of the above pipeline is a list of the files in the current directory that begin with “apple”.

You can chain more than two commands in a pipeline. This is a very powerful feature of **qsh** that allows you to combine several commands together to accomplish a complex task.

There are two other types of lists that are like pipelines. An "and" list stops when the first command in the list has non-zero exit status. An "or" list stops when the first command in the list has a zero exit status.

An "Lists" on page 23 runs a command in the background. For example, the command
`mypgm &`

allows you to start mypgm and then run other commands before mypgm completes. If you have a long running command, an asynchronous list allows you to start the command and not wait for the command to complete.

Input and output redirection

"Redirection" on page 21 allow you to change where input for a command comes from and where output for the command goes to. For Qshell commands, input and output work on descriptors. A descriptor can be opened to either an object in the Integrated File System or to a TCP/IP socket. Input comes from descriptor 0 or standard input, regular output goes to descriptor 1 or standard output, and error output goes to descriptor 2 or standard error.

You can change where input comes from by redirecting standard input. For example, in the command
`grep orange <fruits.list`

when the **grep** utility reads from standard input it receives the contents of the file fruits.list.

You can change where output goes to by redirecting standard output. For example, in the command
`grep apple fruits.list >apple.list`

when the **grep** utility writes the results to standard output, the results are written to the file apple.list.

You can also send standard output and standard error to the same file. For example, in the command
`grep apple fruits.list >apple.list 2>&1`

standard output (descriptor 1) is written to the file apple.list and standard error (descriptor 2) is redirected to the same place as descriptor 1.

While most of the time redirections are only used to control standard input, standard output, and standard error, you can control the descriptors from 0 to 9 using redirections.

Path name expansions

A "Path name expansion" on page 20 substitutes a "Patterns" on page 20 for all of the files that match the pattern. A shell pattern uses:

- A ***** to match any string of characters. For example, in the command

```
ls *.java
```

qsh expands *.java to all of the files that end with .java in the current working directory.

- A **?** to match any single character. For example, in the command

```
ls *.*?
```

qsh expands *.*? to all of the files that have a single character extension.

- A [] for a character class. With a character class, **qsh** matches a set or range of characters. For example, in the command

```
ls *. [ch]
```

qsh expands *. [ch] to all of the files that end in either .c or .h in the current working directory. You can also specify a range of characters. For example, in the command

```
ls *.jav[a-c]
```

qsh expands *.jav[a-c] to all of the files that end in .java, .javb, or .javc.

Parameter expansions

A “Parameter expansions” on page 15 substitutes the value of a variable. In the simplest form

```
$myvar
```

qsh substitutes the value of the variable myvar.

There are modifiers to use default or alternate values or to indicate an error if the variable is unset or null. For example, in the parameter expansion

```
${counter:=0}
```

qsh sets the default value of the variable counter to zero if the variable is unset or null. If the variable counter was already set, the value is not changed and the current value is substituted.

There are also modifiers to remove small or large prefix or suffix patterns. The patterns are the same as the ones used for path name expansions. There are four pattern modifiers:

- The % modifier means to remove the smallest suffix pattern.
- The %% modifier means to remove the largest suffix pattern.
- The # modifier means to remove the smallest prefix pattern.
- The ## modifier means to remove the largest prefix pattern.

For example, if the variable pathname is set to “/fruits/apples/grannysmith”, then in the parameter expansion

```
${pathname%/*}
```

qsh removes the smallest right pattern that matches “/*” and “/fruits/apples” is substituted.

Command substitutions

A “Command substitutions” on page 17 allows the output of a command to be substituted in place of the command name. For example, in the command substitution

```
$(grep apple fruit.list)
```

qsh substitutes the output of the **grep** command. This is an easy way to capture the output of a command for further processing in a script.

An older form of command substitution that uses backquotes (`) is supported but should not be used because of its ambiguous quoting rules.

Qshell utility features

There are over 100 utilities provided with Qshell that provide many functions. A utility is one of two types:

- A built-in utility (page 22) is one qsh can run directly without having to search for it. It runs in the same process as the shell interpreter.
- A regular utility (page 23) is a separate program object that qsh finds by searching for it. It runs in a new process started by the shell interpreter.

A Qshell utility has the following format. The square brackets indicate something that is optionally specified.

```
utility [ options ] [ parameters ]
```

Some utilities allow single letter options preceded by a minus sign (-). For example, several utilities use the -r option for recursively working on a directory tree. More than one option can be specified and all options must be specified before any parameters. If a parameter begins with a minus sign, you can use the — option to indicate the end of options. For example, in the command line

```
utility -r -- -1
```

the -1 is treated as a parameter because the — marked the end of the options.

Navigating in the Integrated File System

When navigating in the Integrated File System, you always have a current working directory. If a file or directory is specified without a leading slash (/), it is assumed to be in the current working directory.

You can change the current working directory with the “cd - Change working directory” on page 88 utility. For example to change the current working directory to /usr/bin, use this command:

```
cd /usr/bin
```

You can display your current working directory with either the “pwd - Return working directory name” on page 121 or “pwdx - Print working directory expanded” on page 121 utilities. The **pwdx** utility resolves symbolic links and displays the absolute current working directory.

You can list the contents of a directory with the “ls - List directory contents” on page 105 utility. With no parameters, **ls** lists the contents of the current working directory. You can also specify one or more directories as parameters. With the -l (lowercase ell) option, **ls** lists detailed information about each object in the directory, including the permissions for the object, the owner and group of the object, the size of the object, and the date that the object was last accessed.

Working with files and directories

You can create a new directory with the “mkdir - Make directories” on page 108 utility. When the -p option is specified, **mkdir** creates all of the directories in the path. For example, to create the new directories “/fruits” and “/fruits/pears”, use this command:

```
mkdir -p /fruits/pears
```

You can copy files with the “cp - Copy files” on page 95 utility. For example, to copy the file “/fruits/apples/delicious” to the file “/fruits/apples/grannysmith”, use this command:

```
cp /fruits/apples/delicious /fruits/apples/grannysmith
```

You can rename or move objects with the “mv - Move files” on page 110 utility. For example, to move the file orange in the current directory to the file “tangerine” in the “/fruits” directory, use this command:

```
mv orange /fruits/tangerine
```

You can delete an object with the “rm - Remove directory entries” on page 123 utility and delete a directory with the “rmdir - Remove directories” on page 124 utility. When the -r option is specified, **rm**

recursively deletes all of the objects in a directory tree. This is an easy way to delete a large number of objects with one command. For example, to delete all of the files and directories in the “/fruits” directory tree, use this command:

```
rm -r /fruits
```

Putting it all together in a script

The following example shows a simple shell script that illustrates the features of the shell interpreter and utilities. The script takes one input parameter that is the name of a directory. The script then copies all of the files with the .java extension from the input directory to the current directory, keeping a count of the files it copied.

```
1 # Get a list of files
2 filelist=$(ls ${1}/*.java)
3 count=0
4 # Process each file
5 for file in $filelist ; do
6   # Strip directory name
7   target=${file##*/}
8   # Copy file to current directory
9   cp $file $target
10  count=$((count+1))
11  # Print message
12  print Copied $file to $target
13 done
14 print Copied $count files
```

On lines 1, 4, 6, 8, 11, the # character denotes a comment. Any characters after the # character are not interpreted by qsh.

On line 2, the variable filelist is set to the output from the “ls - List directory contents” on page 105 command. The \${1} expands to the first input parameter and the *.java expands to all of the files with the .java extension.

On line 3, the variable count is set to zero.

On line 5 is a for loop. For each iteration of the loop, the variable file is set to the next element in the variable filelist. Each element is delimited by a field separator. The default field separators are tab, space, and newline. The semicolon character is a command delimiter and allows you to put more than one command on a line.

On line 7, the variable target is set to the file name from the fully-qualified path name. The \${file##*/} parameter expansion removes the largest pattern starting from the left that matches all characters up to the last slash character.

On line 9, the file is copied with the “cp - Copy files” on page 95 utility from the specified directory to the current working directory.

On line 10, the variable count is incremented by one.

On line 12, a message is printed using the “print - Write output” on page 133 utility with the files that were copied.

On line 13, the done marks the end of the for loop.

On line 14, a message is printed with the total number of files that were copied.

If the directory /project/src contained two files with the .java extension and the script is called using the command:

```
javacopy /project/src
```

then the output from the script is

```
Copied /project/src/foo.java to foo.java  
Copied /project/src/bar.java to bar.java  
Copied 2 files
```

Command language

qsh is a program that:

- reads input from either a file or a terminal
- breaks the input into tokens
- parses the input into simple and compound commands
- performs various expansions on each command
- performs redirection of input and output
- runs the commands
- optionally waits for the commands to complete

qsh implements a command language that has flow control constructs, variables, and functions. The interpretative language is common to both interactive and non-interactive use (shell scripts). So the same commands that are entered at an interactive command line can also be put in a file and the file can be run directly by **qsh**.

For more information about particular features of **qsh**, see the following topics:

“Quoting” on page 9

Select this link to view information about quoting, including the escape character, literal quotes, and grouping quotes.

“Parameters” on page 9

Select this link to view information about parameters, including positional parameters and special parameters.

“Variables” on page 10

Select this link to view information about variables, including the variables set by and variables used by **qsh**.

“Word expansions” on page 15

Select this link to view information about word expansions, including tilde expansion, parameter expansion, command substitution, arithmetic expansion, field splitting, path name expansion, and quote removal.

“Redirection” on page 21

Select this link to view information about input and output redirections.

“Simple commands” on page 22

Select this link to view information about running simple commands.

“Pipelines” on page 23

Select this link to view information about running multiple commands in a pipeline.

“Lists” on page 23

Select this link to view information about running multiple commands in a list.

“Compound commands” on page 24

Select this link to view information about compound commands, including grouping commands, conditional commands, loop commands, and functions.

Quoting

Use quoting to remove the special meaning of certain characters to **qsh**. The following characters may be used:

- The escape character (backslash) to remove the special meaning of the following character with the exception of <newline>. If a <newline> follows the backslash, **qsh** interprets it as a line continuation. For example, `\$` removes the special meaning of the dollar sign.
- Literal (or single) quotation marks ('...') to remove the special meaning of all characters except the single quotation mark.
- Grouping (or double) quotation marks ("...") to remove the special meaning of all characters except dollar sign (\$), back quotation mark (`), and backslash (\). The backslash retains its special meaning as an escape character only when it is followed by a dollar sign (\$), back quotation mark (`), double quotation mark ("), backslash (\), or <newline>.

Parameters

A parameter is used to store data. You can access the value of a parameter by preceding its name with a dollar sign (\$) and surrounding the name with brackets ({ }). The brackets are optional when the name is a single digit, is a special parameter, or is a single identifier. See “Parameter expansions” on page 15 for more information about expanding the value of a parameter.

Positional parameters

A positional parameter is a decimal number starting from one. Initially, **qsh** sets the positional parameters to the command line arguments that follow the name of the shell script. The positional parameters are temporarily replaced when a shell function is called and can be reassigned using the “set - Set or unset options and positional parameters” on page 158 and “shift - Shift positional parameters” on page 160 utilities.

Special parameters

A special parameter is denoted by one of these special characters:

*** (Positional parameters)**

(Asterisk) Expands to the positional parameters, starting from one. When the expansion occurs within a string with quotation marks, it expands to a single field with the value of each parameter separated by the first character of the **IFS** variable, or by a <space> if **IFS** is unset.

@ (Positional parameters)

(At sign) Expands to the positional parameters, starting from one. When the expansion occurs within quotation marks, each positional parameter expands as a separate argument. If there are no positional parameters, the expansion of @ generates zero arguments, even when @ is in quotation marks.

(Number of positional parameters)

(Number sign) Expands to the decimal number of positional parameters. It is initially set to the number of arguments when **qsh** is invoked. It can be changed by the “set - Set or unset options and positional parameters” on page 158, “shift - Shift positional parameters” on page 160, or “dot (.) - Run commands in current environment” on page 52 utilities or by calling a function.

? (Exit status)

(Question mark) Expands to the decimal exit status of the most recent command. A value of zero indicates successful completion. A non-zero value indicates an error. A command ended by a signal number has an exit status of 128 plus the signal number.

- (Option flags)

(Minus) Expands to the current option flags (the single-letter option names concatenated into a string) as specified when **qsh** is invoked, by “set - Set or unset options and positional parameters” on page 158, or implicitly by **qsh**.

\$ (Process ID of current shell)

(Dollar sign) Expands to the decimal process ID of the current shell. A subshell retains the same value of \$ as the current shell even if the subshell is running in a different process.

! (Background process ID)

(Exclamation mark) Expands to the decimal process ID of the most recent background command run from the current shell. For a pipeline, the process ID is that of the last command in the pipeline.

0 (Name of shell script)

(Zero) Expands to the name of the shell or shell script.

Variables

When it is started, **qsh** initializes shell variables from the defined environment variables. A variable is used to store data. You can change the value of an existing variable or create a new variable by using one of these methods:

- Assigning a variable using name=value.
- Calling the “read - Read a line from standard input” on page 135 or “getopts - Parse utility options” on page 163 utilities.
- Using the *name* parameter in a **for** loop or **select** conditional construct.
- Using the \${name=value} parameter expansion.
- Calling the “declare - Declare variables and set attributes” on page 154 or “typeset - Declare variables and set attributes” on page 160 utilities.

Variable names can contain alphabetic characters, numeric characters, or the underscore (_). A variable name cannot begin with a numeric character.

Variables set by qsh

_ (Temporary variable)

This variable is set by **qsh** to the last argument of the previous simple command.

EGID (Effective primary group identifier)

This variable set by **qsh** to the effective primary group identifier of the process at the time **qsh** is started. This variable is read-only.

EUID (Effective user identifier)

This variable set by **qsh** to the effective user identifier of the process at the time **qsh** is started. This variable is read-only.

GID (Primary group identifier)

This variable set by **qsh** to the primary group identifier of the process at the time **qsh** is started. This variable is read-only.

HOSTID (IP identifier of host)

This variable set by **qsh** to the IP address of the host system.

HOSTNAME (Name of host)

This variable set by **qsh** to the name of the host system.

HOSTTYPE (Type of host)

This variable set by **qsh** to a string that represents the type of the host system. The value is set to “powerpc”.

JOBNAME (Qualified job name)

This variable is set by **qsh** to the qualified job name of the current job. The qualified job name is used by CL commands to identify a job.

LAST_JOBNAME (Qualified job name of last job)

This variable is set by **qsh** to the qualified job name of the last job it started. The qualified job name is used by CL commands to identify a job.

LINENO (Line number)

This variable is set by **qsh** to the current line number (decimal) in a script or function before it runs each command.

MACHTYPE (Machine type)

This variable is set by **qsh** to a string that represents the machine type. The value is set to "powerpc-ibm-os400".

OLDPWD (Previous working directory)

This variable is set by "cd - Change working directory" on page 88 to the previous working directory after the current working directory is changed.

OPTARG (Option argument)

This variable is set by "getopts - Parse utility options" on page 163 when it finds an option that requires an argument.

OPTIND (Option index)

This variable is set by "getopts - Parse utility options" on page 163 to the index of the argument to look at for the next option. The variable is set to one when **qsh**, a script, or a function is invoked.

OSTYPE (Operating system type)

This variable is set by **qsh** to a string that represents the operating system type. The value is set to "os400".

PPID (Parent process ID)

This variable is set by **qsh** to the decimal process ID of the process that invoked the current shell. In a subshell, the value of the variable is not changed even if the subshell is running in a different process.

PWD (Working directory)

This variable is set by "cd - Change working directory" on page 88 to the current working directory after it is changed.

QSH_VERSION (Current version)

This variable is set by **qsh** to a string that represents the current version. The string is in the form VxRyMz where x is the version number, y is the release number, and z is the modification number. This variable is read-only.

RANDOM (Random number generator)

This variable is set by **qsh** to an integer random number between 1 and 32767 each time it is referenced. You can seed the random number generator by setting the variable.

REPLY (Reply variable)

This variable is set by "read - Read a line from standard input" on page 135 to the characters that are read when you do not specify any arguments and by the select compound command to the contents of the input line read from standard input.

TERMINAL_TYPE (Type of terminal)

This variable is set by **qsh** to the type of terminal attached to the standard file descriptors. The value is set to "5250" when attached to a 5250 display, to "REMOTE" when attached to a remote client, or to "PIPELINE" when attached to pipes.

UID (User identifier)

This variable set by **qsh** to the user identifier of the process at the time **qsh** is started. This variable is read-only.

Variables used by qsh**CDPATH (Search path for cd)**

If the directory you specify for “cd - Change working directory” on page 88 does not begin with a slash (/), **qsh** searches the directories listed in **CDPATH** in order for the specified directory. The value of the variable is a colon separated list of directories. The current working directory is specified by a period (.) or a null directory before the first colon, between two colons, or after the last colon. There is no default value.

ENV (Environment file)

When **qsh** is invoked, it performs parameter expansion, command substitution, and arithmetic expansion on this variable to generate the path name of a shell script to run in the current environment. It is typically used to set aliases, define functions, or set options. There is no default value.

HOME (Home directory)

The value of this variable is the path name of your home directory. The value is used for tilde expansion and as the default argument for “cd - Change working directory” on page 88. The value is set by default to the value specified in your user profile.

IFS (Internal field separators)

The value is a string treated as a list of characters that is used for field splitting and to split lines into fields with “read - Read a line from standard input” on page 135. The first character of the value is used to separate arguments when expanding the * special parameter. The default value is “<space><tab><newline>”.

LANG (Language locale)

This variable defines the locale category used for categories that are not specifically set with a variable starting with **LC_**. There is no default value.

LC_ALL (Locale settings)

This variable overrides the value of any variables starting with **LC_**. There is no default value.

LC_COLLATE (Locale collation)

This variable defines the collation relations between characters. There is no default value.

LC_CTYPE (Locale character classes)

This variable defines character types such as upper-case, lower-case, space, digit and, punctuation. There is no default value.

LC_MESSAGES (Locale message formatting)

This variable defines the format and values for affirmative and negative responses from applications. There is no default value.

LC_MONETARY (Locale monetary formatting)

This variable defines the monetary names, symbols, and other details. There is no default value.

LC_NUMERIC (Locale numeric formatting)

This variable defines the decimal point character for formatted input/output and string conversion functions. There is no default value.

LC_TIME (Locale time formatting)

This variable defines the date and time conventions, such as calendar used, time zone, and days of the week. There is no default value.

LC_TOD (Locale time zone)

This variable defines the time zone name, time zone difference, and Daylight Savings Time start and end. There is no default value.

NLSPATH (Search path for message catalogs)

When opening a message catalog, the system searches the directories listed in the order specified until it finds the catalog. The value of the variable is a colon separated list of directories. There is no default value.

PATH (Search path for commands)

If the command you specify does not begin with a slash (/), **qsh** searches the directories listed in the order specified until it finds the command to run. The value of the variable is a colon separated list of directories. The current working directory is specified by a period (.) or a null directory before the first colon, between two colons, or after the last colon. The default value is `"/usr/bin:"`.

PS1 (Primary prompt string)

When the interactive option is set, **qsh** performs parameter expansion, command substitution, and arithmetic expansion on the variable and displays it on stderr when **qsh** is ready to read a command. The default value is `"$"`.

PS2 (Secondary prompt string)

When you enter <newline> before completing a command **qsh** displays the value of this variable on stderr. The default value is `">"`.

PS3 (Select command prompt)

When the **select** compound command is run, **qsh** performs parameter expansion, command substitution, and arithmetic expansion on the variable and displays it on stderr to prompt the user to select one of the choices displayed by **select**. The default value is `"#?"`.

PS4 (Debug prompt string)

When the execution trace option is set and the interactive option is set, **qsh** performs parameter expansion, command substitution, and arithmetic expansion on the variable and displays it on stderr before each line in the execution trace. The default value is `"+"`.

QIBM_CCSID (CCSID for translation)

When this variable is set to a numeric value, **qsh** and various utilities use the value for creating files and translating data from the CCSID of the job. The default value is `"0"` for the default job CCSID. A value of `"65535"` means no translation is done.

QIBM_CHILD_JOB_SNDINQMSG (Send inquiry message when child process starts)

When this variable is set to a positive numeric value, the parent process is sent an inquiry message with the qualified job name of the child process. The child process is held until you reply the message. By setting this variable, you can debug the program running in the child process by setting breakpoints before the program runs. The value of the variable is the level of descendant processes to debug. When set to 1, child processes are held, when set to 2 child and grandchild processes are held, etc. There is no default value.

QIBM_MULTI_THREADED (Start multi-thread capable processes)

This variable determines if processes started by **qsh** can create multiple threads. When the value of the variable is `"Y"`, all child processes started by **qsh** can start threads. The default value is `"N"`.

QSH_REDIRECTION_TEXTDATA (Process data as text for file redirection)

This variable determines if data read from or written to a file specified on a "Redirection" on page 21 is treated as text data or binary data. When the value of the variable is `"Y"`, **qsh** treats the data read from or written to the file as text data. When the value of the variable is not `"Y"`, **qsh** treats the data read from or written to the file as binary data. The default value is `"Y"`.

QSH_USE_PRESTART_JOBS (Use pre-start jobs when available)

This variable determines if processes started by **qsh** use prestart jobs when available. When the value of the variable is `"Y"`, **qsh** uses prestart jobs if they are available in the current subsystem. When the value of the variable is not `"Y"`, or prestart jobs are not available, the processes started by **qsh** are batch immediate jobs. The default value is `"Y"`.

SHELL (Path name of the shell)

When running a script file that does not contain “#!” on the first line, **qsh** uses the value of this variable as the path name of the shell interpreter to run the script. There is no default value.

TRACEFILE (Path name of trace file)

When the trace option is set, **qsh** uses the value of this variable as the path name of the file to store the trace information. The default value is “\$HOME/qsh_trace”.

TRACEOPT (Options for trace file)

When the trace option is set, **qsh** uses the value of this variable to determine how to handle the trace file. When the value of the variable is “UNLINK”, **qsh** unlinks the trace file before opening it in a root shell. When the value of the variable is “KEEP”, **qsh** keeps the current trace file. The default value is “UNLINK”.

Other variables

» QIBM_CMP_FILE_SIZE

This variable controls the maximum file size in bytes that **cmp** reads into an internal buffer for better performance. For files larger than the maximum size, **cmp** reads the files one byte at a time. «

QIBM_OD_OUTPUT_FORMAT (Output format for od)

This variable controls the output format for the “od - Dump files in various formats” on page 110 utility. If the value is “OLD”, **od** uses the old format from previous releases. The old format is not compatible with the current industry standard and its use is discouraged. There is no default value.

QIBM_QSH_CMD_ESCAPE_MSG (Send escape messages from QSH CL command)

This variable controls how messages are sent by the QSH CL command when the CMD parameter is specified. If the value is “Y”, the QSH0005 message is sent as an escape message if the exit status is greater than zero and the QSH0006 and QSH0007 messages are always sent as escape messages. There is no default value.

QIBM_QSH_CMD_OUTPUT (Control output of QSH CL command)

This variable controls the output from the QSH CL command when the CMD parameter is specified. If the value is “STDOUT”, the output is displayed on the C runtime terminal session. If the value is “NONE”, the output is discarded. If the value is “FILE”, the output is written to the specified file. If the value is “FILEAPPEND”, the output is appended to the specified file. The default value is “STDOUT”.

QIBM_QSH_INTERACTIVE_CMD (Initial interactive command)

When this variable is set to a command string, **qsh** runs the command when an interactive session is started. The variable must be set before calling the QSH CL command to have **qsh** run the command. There is no default value.

QIBM_QSH_INTERACTIVE_TYPE (Type of interactive session)

This variable sets the type of the interactive session started by the QSH CL command. If the value is “NOLOGIN”, the interactive session is not a login session. Otherwise the interactive session is a login session. There is no default value.

QIBM_SYSTEM_ALWMLTTHD (Allow multi-threaded jobs for system)

This variable controls how the “system - Run CL command” on page 59 utility behaves in a multi-thread capable job. If the value of the variable is “Y” and there is only one thread in the job, **system** runs the CL command in the job. Otherwise, **system** starts a new job to run the CL command. There is no default value.

QIBM_SYSTEM_USE_ILE_RC

Set this environment variable to control how the “system - Run CL command” on page 59 utility sets the exit status. If the value of the variable is “Y”, **system** sets the exit status to the ILE return code of the program called by the CL command, or zero if the program did not set a return code. There is no default value.

Word expansions

For more information about the word expansions supported by **qsh**, see the following topics:

“Tilde expansions”

Select this link to view information about how **qsh** expands tilde characters.

“Parameter expansions”

Select this link to view information about how **qsh** expands parameters.

“Command substitutions” on page 17

Select this link to view information about how **qsh** expands command substitutions.

“Arithmetic expansions” on page 18

Select this link to view information about how **qsh** expands arithmetic expressions.

“Field splitting” on page 20

Select this link to view information about how **qsh** splits fields into words expands path names using patterns, and remove quotation marks.

“Patterns” on page 20

Select this link to view information about how **qsh** expands patterns.

Tilde expansions

An unquoted tilde character (~) at the beginning of a word is expanded according to the following rules:

- ~ expands to the value of the **HOME** variable (the current user’s home directory).
- ~**user** expands to the home directory of the specified user. All the characters up to a slash (/) or the end of the word are treated as a user name.
- ~+ expands to the value of the **PWD** (working directory) variable.
- ~~ expands to the value of the **OLDPWD** (previous working directory) variable if it is set.

Examples

1. Change the current directory to the user’s home directory:

```
cd ~
```

2. Change the current directory to the bin directory in user smith’s home directory:

```
cd ~smith/bin
```

Parameter expansions

The format for parameter expansion is as follows:

```
`${expression}
```

where *expression* consists of all characters until the matching right brace (}). Any right brace characters escaped by a backslash or within a string with quotation marks, as well as characters in embedded arithmetic expansions, command substitutions, and variable expansions, are not examined in determining the matching right brace.

The simplest form for parameter expansion is:

```
`${parameter}
```

The value, if any, of *parameter* is substituted. The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when *parameter* is followed by a character that might be interpreted as part of the name. If a parameter expansion occurs inside double quotation marks, then:

1. Path name expansion is not performed on the results of the expansion.
2. Field splitting is not performed on the results of the expansion, with the exception of @ special parameter.

A parameter expansion can be modified by using one of the following formats:

`${parameter:-word}`

Use Default Values. If *parameter* is unset or null, the expansion of *word* is substituted. Otherwise, the value of *parameter* is substituted.

`${parameter:=word}`

Assign Default Values. If *parameter* is unset or null, the expansion of *word* is assigned to *parameter*. In all cases, the final value of *parameter* is substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.

`${parameter:?word}`

Indicate Error if Null or Unset. If *parameter* is unset or null, the expansion of *word* (or a message indicating it is unset if *word* is omitted) is written to standard error and a non-interactive shell exits with a nonzero exit status. Otherwise, the value of *parameter* is substituted.

`${parameter:+word}`

Use Alternate Value. If *parameter* is unset or null, null is substituted. Otherwise, the expansion of *word* is substituted.

In the preceding four parameter expansions, using a colon in the format results in a test for a parameter that is unset or null; removing the colon results in a test for a parameter that is only unset.

`${#parameter}`

String Length. If *parameter* is @ or *, the number of positional parameters is substituted. Otherwise, the length of the value of *parameter* is substituted.

`${parameter%word}`

Remove Smallest Suffix Pattern. The *word* is expanded to produce a “Patterns” on page 20. Then the result is *parameter* after removing the smallest portion of the suffix matched by the pattern.

`${parameter%%word}`

Remove Largest Suffix Pattern. The *word* is expanded to produce a “Patterns” on page 20. Then the result is *parameter* after removing the largest portion of the suffix matched by the pattern.

`${parameter#word}`

Remove Smallest Prefix Pattern. The *word* is expanded to produce a “Patterns” on page 20. Then the result is *parameter* after removing the smallest portion of the prefix matched by the pattern.

`${parameter##word}`

Remove Largest Prefix Pattern. The *word* is expanded to produce a “Patterns” on page 20. Then the result is *parameter* after removing the largest portion of the prefix matched by the pattern.

`${parameter:offset}`

`${parameter:offset:length}`

Substring Starting at Offset. The value of this expansion is the substring starting at the byte specified by *offset* for *length* bytes. If *length* is not specified or the value of *length* causes the expansion to exceed the length of *parameter*, the substring ends with the last byte of *parameter*. Both *offset* and *length* are “Arithmetic expansions” on page 18 and must evaluate to a value that is greater than or equal to zero. The first byte of *parameter* is defined by an offset of zero.

`${parameter/pattern/string}`

`${parameter//pattern/string}`

Substitute String for Pattern. The value of this expansion is the value of *parameter* with the longest match of *pattern* replaced with *string*. In the first form, only the first match of *pattern* is

replaced. In the second form, all matches of *pattern* are replaced. If *pattern* begins with #, it must match at the beginning of *parameter*. If *pattern* begins with a %, it must match at the end of *parameter*.

Examples

1. Expand the variable QSH_VERSION.
echo \${QSH_VERSION}
2. Expand the variable filename and use a default value.
echo \${filename:-/tmp/default.txt}
3. Expand the variable index and assign a default value.
echo \${index:=0}
4. Expand the variable filename and indicate an error if unset.
echo \${filename:?Variable is not set}
5. Expand the variable DIRLIST using string length.
DIRLIST=/usr/bin:/home/mike
echo \${#DIRLIST}
6. Expand the variable DIRLIST using remove smallest suffix pattern.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST%/*}
7. Expand the variable DIRLIST using remove largest suffix pattern.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST%:*}
8. Expand the variable DIRLIST using remove smallest prefix pattern.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST#/usr}
9. Expand the variable DIRLIST using remove largest prefix pattern.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST##*/}
10. Expand the variable DIRLIST using a substring starting at offset.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST:5:3}
11. Expand the variable DIRLIST using a substitute string for pattern.
DIRLIST=/usr/bin:/home/mike
echo \${DIRLIST/m?ke/joel}

Command substitutions

Command substitution allows the output of a command to be substituted in place of the command name itself. Command substitution occurs when the command is enclosed as follows:

```
$(command)
```

or by using backquotes:

```
`command`
```

The backquoted version is provided for compatibility. Its use is discouraged.

The shell expands the command substitution by running *command* in a subshell environment and replacing the command substitution with the standard output of the *command*, removing sequences of one or more <newline>s at the end of the substitution. Embedded <newline>s before the end of the output are not removed; however, during field splitting, they may be translated into <space>s, depending on the value of the IFS variable and quoting that is in effect.

Examples

1. Set the variable list to the output of the ls command:

```
list=$(ls)
```

Arithmetic expansions

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion is:

```
$(expression)
```

The *expression* is treated as if it were in double quotation marks, except that a double quotation mark inside *expression* is not treated specially. The shell expands all tokens in *expression* for parameter expansion, command substitution, and quote removal. **qsh** treats the result as an arithmetic expression and substitutes the value of the expression.

Arithmetic expressions

An arithmetic expression can be specified in the following situations:

- in an arithmetic expansion
- for each argument of the “let - Evaluate arithmetic expression” on page 164 utility
- for the argument of the “shift - Shift positional parameters” on page 160 utility
- for the operands of the arithmetic formats of the “printf - Write formatted output” on page 133 utility
- for the operands to the arithmetic comparison operators of the “test - Evaluate expression” on page 165 utility
- for the argument of the “ulimit - Set or display resource limits” on page 183 utility
- in the “Substring Starting at Offset” parameter expansion

qsh performs either integer or floating point arithmetic based on the setting of the float option. When the float option is set on, **qsh** performs floating point arithmetic.

An integer number has the format [*base#*]*number* where:

- *base* is a decimal integer between 2 and 36 that specifies the arithmetic base. The default is base 10.
- *number* is a non-negative number. For a base greater than 10, numbers greater than 9 or represented using a letter of the alphabet. For example, when using base 16, the decimal number 10 is represented using A.

A floating point number has the format [+|-] *number*[.*number*] [*exponent*] where:

- *number* is a non-negative decimal number.
- *exponent* is E or e followed by + or - and a non-negative decimal number.

Arithmetic expressions use the following ANSI C language operators and precedence.

(*expression*)

 Parenthesis overrides precedence rules

Unary operators

 +*expression* Unary +

 -*expression* Unary -

 ~*expression* Bitwise negation

 !*expression* Logical negation

Multiplicative operators

expression * *expression* Multiplication

expression / expression Division

expression % expression Remainder

Additive operators

expression + expression Addition

expression - expression Subtraction

Bitwise shift operators

expression << expression Left shift the first expression by the number of bits given in the second expression

expression >> expression Right shift the first expression by the number of bits given in the second expression

Relational operators

expression < expression Less than

expression <= expression Less than or equal to

expression > expression Greater than

expression >= expression Greater than or equal to

Bitwise AND operator

expression & expression Bitwise and where the result contains a 1 in each bit position where there is a 1 in both expressions and a 0 in all other bit positions.

Bitwise Exclusive OR operator

expression ^ expression Bitwise exclusive or where the result contains a 1 in each bit position where there is a 1 in only one of the expressions and a 0 in all other bit positions.

Bitwise OR operator

expression | expression Bitwise or where the result contains a 1 in each bit position where there is a 1 in either expression and a 0 in all other bit positions.

Logical AND operator

expression && expression Logical and where the result is true if both expressions are true

Logical OR operator

expression || expression Logical or where the result is true if one of the expressions is true

Conditional operator

expression ? expression : expression Conditional operator where when the first expression is true, the second expression is evaluated. Otherwise the third expression is evaluated.

Assignment operators

expression = expression Simple assignment

*expression *= expression* Assign and multiply

expression /= expression Assign and divide

expression %= expression Assign and remainder

expression += expression Assign and add

expression -= expression Assign and subtract

expression <<= expression Assign and shift left

expression >>= expression Assign and shift right

expression &= expression Assign and bitwise AND

expression ^= expression Assign and bitwise exclusive OR

expression |= expression Assign and bitwise OR

Note:

When using floating point arithmetic the remainder, left shift, right shift, bitwise AND, bitwise exclusive OR, and bitwise OR operators are not supported.

Examples

1. Add two decimal numbers:

```
echo $((2+2))
```

2. Add two hexadecimal numbers:

```
echo $((16#A + 16#20))
```

3. Increment the variable index by one:

```
let index+=1
```

4. Evaluate a complex expression:

```
echo $((5+9-2*3/2))
```

5. Add two floating point numbers:

```
set -F
echo $((5.75+9.157))
set +F
```

Field splitting

After parameter expansion, command substitution, and arithmetic expansion, **qsh** scans the results of expansions and substitutions that did not occur in double quotation marks for field splitting. Multiple fields can result.

qsh treats each character of the **IFS** variable as a delimiter and uses the delimiters to split the results of parameter expansion and command substitution into fields. If the value of the **IFS** variable is null, no field splitting is performed.

Path name expansion

When the **noglob** option is not set, path name expansion is performed after field splitting is complete. Each word is viewed as a series of "Patterns," separated by slashes. The process of expansion replaces the word with the names of all existing files whose names can be formed by replacing each pattern with a string that matches the specified pattern. There are two restrictions:

1. a pattern cannot match a string containing a slash
2. a pattern cannot match a string starting with a period unless the first character of the pattern is a period

Quote removal

The quote characters, backslash (\), single quotation mark ('), and double quotation mark ("), are removed unless the character has been quoted.

Patterns

A pattern consists of normal characters, which match themselves, and meta-characters. The meta-characters are:

!, *, ?, and [

These characters lose their special meanings if they are quoted. When command or variable substitution is performed and the dollar sign (\$) or backquote (`) are not double quoted, the value of the variable or the output of the command is scanned for these characters and they are turned into meta-characters.

An asterisk (*) matches any string of characters.

A question mark (?) matches any single character.

A left bracket ([) introduces a character class. The end of the character class is indicated by a right bracket (]). If the right bracket is missing then the left bracket matches a [rather than introducing a character class. A character class matches any of the characters between the square brackets. A range of characters may be specified using a minus (-). The character class may be complemented by making an exclamation mark (!) the first character of the character class.

Note:

Specifying a range of characters may produce different results from other systems because EBCDIC characters are not contiguous.

To include a right bracket in a character class, make it the first character listed (after the !, if any). To include a minus in a character class, make it the first or last character listed.

Redirection

Redirections are used to change where a command reads its input or sends its output. In general, redirections open, close, or duplicate an existing reference to a file. The overall format used for redirection is as follows:

[*n*] *redir-op file*

where *redir-op* is one of the redirection operators listed below and *n* is an optional number that refers to a file descriptor. Following is a list of the possible redirections.

[*n*]< *file*

Redirect standard input (or *n*) from *file*.

[*n1*]<&*n2*

Duplicate standard input (or *n1*) from file descriptor *n2*.

[*n*]<&-

Close standard input (or *n*).

[*n*]> *file*

Redirect standard output (or *n*) to *file*.

[*n*]>| *file*

Redirect standard output (or *n*) to *file*, but override the noclobber option.

[*n*]>> *file*

Append standard output (or *n*) to *file*.

[*n1*]>&*n2*

Duplicate standard output (or *n1*) from *n2*.

[*n*]>&-

Close standard output (or *n*).

➤ It is best not to use the /QSYS.LIB/QTEMP.LIB directory for redirections since it is deleted when a job ends and a new job is started and ended for each command. ⬅

Here-documents

The format of a here-document is:

[*n*]<<[-] *delimiter*

here-doc-text ...

delimiter

All the text on successive lines up to *delimiter* is saved and made available to the command on standard input, or file descriptor *n* if it is specified. If *delimiter* as specified on the initial line is quoted, then *here-doc-text* is treated literally, otherwise the text is subjected to parameter expansion, command substitution, and arithmetic expansion. If the operator is <<- instead of <<, then leading tabs in *here-doc-text* are stripped.

Simple commands

A simple command is a sequence of optional variable assignments and redirections followed by a command name. When a simple command is recognized by **qsh**, it performs the following actions:

1. Leading words of the form *name=value* are stripped off and assigned to the environment of the simple command. Redirection operators and their arguments are saved for processing in step 3.
2. The remaining words are expanded as described in “Word expansions” on page 15, and the first remaining word is considered the command name. Any additional words are considered the arguments of the command. If no command name is found, then the *name=value* variable assignments recognized in step 1 affect the current shell.
3. Redirections are performed as described in “Redirection” on page 21.

Path search

If a simple command does not contain any slashes, **qsh** finds the command by searching:

1. for a special built-in utility of that name, then
2. for a shell function of that name, then
3. for a regular built-in utility of that name, then
4. each directory in the **PATH** variable in turn for the regular utility.

Command names containing a slash (/) are run as a regular utility without performing any of the above searches.

A built-in utility is run internal to the shell, without starting a new process. A special built-in utility is different from a regular built-in utility in these respects:

1. A syntax error in a special built-in utility causes a non-interactive shell to exit.
2. Variable assignments specified with a special built-in utility remain in effect after the utility completes.

These are the special built-in utilities: “break - Exit from for, while, or until loop” on page 162, “colon (:)- Null utility” on page 162, “continue - Continue for, while, or until loop” on page 163, “declare - Declare variables and set attributes” on page 154, “dot (.) - Run commands in current environment” on page 52, “eval - Construct command by concatenating arguments” on page 53, “exec - Run commands and open, close, or copy descriptors” on page 53, “exit - Exit from the shell” on page 54, “export - Set export attribute for variables” on page 155, “local - Assign a local variable in a function” on page 156, “readonly - Set read-only attribute for variables” on page 157, “return - Return from a function” on page 164, “set - Set or unset options and positional parameters” on page 158, “shift - Shift positional parameters” on page 160, “source - Run commands in current environment” on page 59, “trap - Trap signals” on page 151, “typeset - Declare variables and set attributes” on page 160, and “unset - Unset values of variables and functions” on page 161.

When a shell function is run, all of the shell positional parameters (except the special parameter **0**, which remains unchanged) are set to the arguments of the shell function. The variables which are explicitly placed in the environment of the command (by placing assignments to them before the function name) are made local to the function and are set to the specified values. The positional parameters are restored to their original values when the shell function completes.

When a regular utility is run, **qsh** starts a new process, passing the arguments and the environment to the program. If the program is a shell script, **qsh** will interpret the program in a subshell. **qsh** will reinitialize itself in this case, so that the effect will be as if a new shell had been invoked to handle the shell script.

Command exit status

Each command has an exit status that can influence the behavior of other shell commands. By convention, a command exits with zero for normal or success, and non-zero for failure, error, or a false indication. The documentation for each command describes the exit codes it returns and what they mean. The exit status can be one of these values:

- 0 for success.
- 1 to 125 for failure.
- 126 when **qsh** finds the command but it is not executable.
- 127 when **qsh** cannot find the command.
- 128 and above when the command is ended by a signal. The value is 128 plus the signal number.

Pipelines

A pipeline is a sequence of one or more commands separated by the pipeline control operator (`|`). The standard output of all but the last command is connected to the standard input of the next command.

The format for a pipeline is:

```
[ ! ] command1 [ | command2 ... ]
```

The standard output of *command1* is connected to the standard input of *command2*. The standard input, standard output, or both of a command is considered to be assigned by the pipeline before any redirection specified by redirection operators that are part of the command. The exit status of the pipeline is the exit status of the last *command*.

If the pipeline is not in the background (described below), **qsh** waits for all commands to complete.

If the reserved word `!` does not precede the pipeline, the exit status is the exit status of the last command specified in the pipeline. Otherwise, the exit status is the logical not of the exit status of the last command. That is, if the last command returns zero, the exit status is 1; if the last command returns greater than zero, the exit status is zero.

Because pipeline assignment of standard input or standard output or both takes place before redirection, it can be modified by redirection. For example:

```
command1 2>&1 | command2
```

sends both the standard output and standard error of *command1* to the standard input of *command2*.

Lists

A list is a sequence of commands separated by an ampersand (`&`) or a semicolon (`;`), and optionally terminated by a `<newline>`, ampersand, or semicolon. An AND-OR list is a sequence of commands separated by a `&&` or `||`. Both operators have the same priority.

Asynchronous lists

If a command is terminated by the control operator ampersand (`&`), **qsh** runs the command asynchronously. That is, **qsh** does not wait for the command to finish before running the next command. The format for running a command in the background is:

command1 & [*command2* & ...]

If the interactive option is not set, the standard input of any asynchronous command is set to `/dev/qsh-stdin-null`. The exit status of an asynchronous list is the exit status of the last *command*.

Sequential lists

Commands that are separated by a semicolon (;) are run sequentially. The format for a sequential list is:

command1 [; *command2* ...]

The commands in the list are run in the order they are written. The exit status of a sequential list is the exit status of the last *command*.

AND lists

The format for an AND list is:

command1 [&& *command2* ...]

With an AND list, **qsh** runs *command1*, and then runs *command2* if the exit status of the *command1* is zero and so on until a command has a non-zero exit status or there are no commands left to run. The exit status of an AND list is the exit status of the last *command* that is run.

OR lists

The format for an OR list is:

command1 [|| *command2* ...]

With an OR list, **qsh** runs *command1*, and then runs *command2* if the exit status of the *command1* is non-zero and so on until a command has a zero exit status or there are no commands left to run. The exit status of an OR list is the exit status of the last *command* that is run.

Compound commands

Compound commands provide control flow for other commands. Each compound command starts with a reserved word and has a corresponding reserved word at the end.

For more information about the compound commands supported by **qsh**, see the following topics:

- **“Grouping commands” on page 25**
Select this link to view information about the grouping commands.
- **“If command” on page 25**
Select this link to view information about the if-then-else-fi command.
- **“Conditional command” on page 26**
Select this link to view information about the conditional command.
- **“Case command” on page 26**
Select this link to view information about the case-esac command.
- **“Select command” on page 27**
Select this link to view information about the select-do-done command.
- **“While command” on page 27**
Select this link to view information about the while-do-done command.
- **“Until command” on page 27**
Select this link to view information about the until-do-done command.

“For command” on page 28

Select this link to view information about the for-do-done command.

“Functions” on page 28

Select this link to view information about functions.

Grouping commands

You can group commands using either

```
( list )
```

or

```
{ list; }
```

In the first case, **qsh** runs *list* in a subshell environment.

Examples

1. Group two commands in a subshell.

```
( ls | grep apple )
```

If command

The syntax of the **if** command is

```
if list1
```

```
then list2
```

```
[ elif list3
```

```
then list4 ] ...
```

```
[ else list5 ]
```

```
fi
```

First, **qsh** runs *list1* and if its exit status is zero then **qsh** runs *list2*. Otherwise, each *elif list3* is run and if its exit status is zero then **qsh** runs *list4*. Otherwise, **qsh** runs *list5*.

Examples

1. An if-then-fi command.

```
x=4
y=9
if test $x -lt $y
then
  echo $x is less than $y
fi
```

2. An if-then-else-fi command.

```
x=10
y=9
if test $x -lt $y
then
  echo echo $x is less than $y
else
  echo echo $x is greater than or equal to $y
fi
```

3. An if-then-elif-else-fi command.

```

x=4
y=4
if test $x -lt $y
then
  echo echo $x is less than $y
elif test $x -eq $y
then
  echo $x is equal to $y
else
  echo $x is greater than or equal to $y
fi

```

Conditional command

The syntax of the `[[...]]` command is

```
[[ expression ]]
```

It returns a status of 0 or 1 depending on the evaluation of the conditional expression *expression*. The format of a conditional expression is the same as the expressions evaluated by the “test - Evaluate expression” on page 165 utility. **qsh** performs tilde expansion, parameter expansion, arithmetic expansion, command substitution, and quote removal on *expression* before it is evaluated.

Examples

1. A conditional command that uses a command substitution.

```

if [[ $(grep -c apple fruits.txt) -eq 0 ]]
then
  echo There are no apples in fruit.txt
fi

```

Case command

The syntax of the **case** command is

```
case word in
```

```
pattern1 ) list1 ;;
```

```
pattern2 | pattern3 ) list2 ;;
```

```
...
```

```
esac
```

qsh expands each *pattern* in turn and sees if it matches the expansion of *word*. When there is a match, **qsh** runs the corresponding *list*. After the first match, no more patterns are expanded. See “Patterns” on page 20 for more details on patterns.

Examples

1. A case command for processing command line options.

```

while getopts ap:t: c ; do
  case $c in
    a) aflag=1;;
    p) pflag=1
       path=$OPTARG;;
    t) time=$OPTARG;;
    *) print -u2 "Invalid option"
       exit 1;;
  esac
done

```

Select command

The syntax of the **select** command is

```
select name [ in word ... ]
```

```
do list
```

```
done
```

The *words* are expanded, generating a list of items. If *word* is not specified, the positional parameters are expanded. The set of expanded words is written to standard error, each preceded by a number. The PS3 prompt is then displayed and a line is read from standard input. If the line consists of a number corresponding to one of the displayed words, **qsh** sets the value of *name* to the word corresponding to the number. If the line is empty, **qsh** displays the list again. The REPLY variable is set to the contents of the input line.

qsh runs the commands in *list* until a “break - Exit from for, while, or until loop” on page 162, “return - Return from a function” on page 164, or “exit - Exit from the shell” on page 54 command is run. **select** also completes if EOF is read from standard input.

Examples

1. A select command to select from a list.

```
PS3="Please select a number "  
list="alpha beta gamma delta epsilon"  
select value in $list ; do  
    echo Value for selection $REPLY is $value  
    break  
done
```

While command

The syntax of the **while** command is

```
while list1
```

```
do list2
```

```
done
```

qsh runs the two lists repeatedly while the exit status of *list1* is zero. When the exit status of *list1* is non-zero the command completes.

Examples

1. A while command to iterate until a condition is met.

```
max=100  
index=0  
while [[ $index -lt $max ]] ; do  
    echo Index is $index  
    let index+=1  
done
```

Until command

The syntax of the **until** command is

```
until list1
```

```
do list2
```

done

qsh runs the two lists repeatedly while the exit status of *list1* is non-zero. When the exit status of *list1* is zero the command completes.

Examples

1. An until command to iterate until a condition is met.

```
max=100
index=0
until [[ $index -eq $max ]] ; do
    echo Index is $index
    let index+=1
done
```

For command

The syntax of the **for** command is

for *variable in word ...*

do *list*

done

The *words* are expanded, and then *list* is run repeatedly with *variable* set to each *word* in turn. You can replace **do** and **done** with braces (`{ }`).

Examples

1. A for command to process a list of objects.

```
list=$(ls *.class)
for object in $list
do
    system "DSPJVAPGM $object"
done
```

Functions

The syntax of a function definition is

[**function**] *name* () *command*

A function definition is a statement that when run installs a function named *name* and returns an exit status of zero. The *command* is normally a list enclosed between braces (`{ }`).

When *name* is specified as a simple command, **qsh** runs *command*. The arguments to the simple command temporarily become the positional parameters while the function is running. The special parameter **0** is unchanged. By using **local**, you can declare local variables inside of the function. By using **return**, you can end the function and resume execution with the next command after the function call.

Examples

Here is an example of a function that provides a **qsh** interface to the PING CL command.

```
ping()
{
    # Initialize variables and make them local to this function
    local nbrpkt='' waittime='' intnetadr='' msgmode='' pktlen='' ipttl='' host=''
    local c

    # Process the options
    while getopt c:i:I:qs:T:v c

```

```

do case $c in
c) nbrpkt="NBRPKT($OPTARG)";;
i) waittime="WAITTIME($OPTARG)";;
I) intnetadr="INTNETADR('$OPTARG')"  
  host="*INTNETADR";;
q) msgmode='MSGMODE(*QUIET)';;
s) pktlen="PKTLEN($OPTARG)";;
T) ipttl="IPTTL($OPTARG)";;
v) msgmode='MSGMODE(*VERBOSE)';;
\?) print -u2 "Usage: ping [-c count] [-i seconds] [-I ipaddr] [-q] \  
  "[-s size] [-T ttl] [-v] hostname"  
  return 1;;
esac
done

# Run the command
shift $OPTIND-1
system ping ${host:-$1} $intnetadr $nbrpkt $waittime $msgmode $pktlen $ipttl
}

```

Using Qshell

This topic provides information on using the QSH CL command, configuring the Qshell environment, and developing your own utilities.

"Using a Qshell interactive session"

Select this link to find out how to use the interactive session started by the QSH CL command.

"Running Qshell commands from CL" on page 31

Select this link to find out how to run Qshell commands from the CL command environment.

"Running Qshell commands from PASE" on page 32

Select this link to find out how to run Qshell commands from the PASE environment.

"Customizing your environment" on page 32

Select this link to find out how to customize the Qshell environment on your system.

"National language support (NLS) considerations" on page 32

Select this link to find out how to configure Qshell for different languages.

"Performance considerations" on page 36

Select this link to find out how to configure Qshell for the best performance on your system.

"Developing your own utilities" on page 36

Select this link to view tips for writing your own utility programs.

"Editing files with Qshell Interpreter" on page 37

Select this link to find out how to edit stream files used for shell scripts.

"Differences with other interpreters" on page 37

Select this link to view the list of differences between Qshell and other shell interpreters.

Using a Qshell interactive session

The Start QSH (STRQSH) command, also known as QSH, is a CL (control language) command that either starts a Qshell interactive session or runs a Qshell command.

If running in an interactive job with no parameters, STRQSH starts an interactive Qshell session. If a Qshell session is not already active in the job, then the following events occur:

1. A new Qshell session is started and a terminal window is displayed.
2. **qsh** runs the commands from the file `/etc/profile` if it exists.
3. **qsh** runs the commands from the file `.profile` in the user's home directory if it exists.
4. **qsh** runs the commands from the file specified by the expansion of the **ENV** variable if it exists.

If a Qshell session is already active in an interactive job, you are reconnected to the existing session.

From the terminal window, you can enter Qshell commands and view output from the commands. The terminal window has two parts:

- an input line for entering commands
- an output area that contains an echo of the commands you entered and any output generated by the commands

You can use these function keys:

Function key	Description
F3 (Exit)	Close the terminal window and end the Qshell session.
F5 (Refresh)	Refresh the output area.
F6 (Print)	Print the output area to a spool file.
F7 (Up)	Roll output area up one page. If a number is on the command line, the output area is rolled up by that number of lines.
F8 (Down)	Roll output area down one page. If a number is on the command line, the output area is rolled down by that number of lines.
F9 (Retrieve)	Retrieve a previous command. You can press this key multiple times to retrieve any previous command. For example, to retrieve the second to last command you entered, press this key two times. You can also select a specific command to be run again by placing the cursor on that command and pressing this key. When the interactive job is running in a double-byte CCSID, this key is not available.
F11 (Toggle line wrap)	Toggle the line wrap/truncate mode in the output area. In line wrap mode, lines longer than the width of the terminal window are wrapped to the next line. In truncate mode, the portion of a line beyond the width of the terminal window is not shown.
F12 (Disconnect)	Disconnect from the Qshell session. This key only closes the terminal window and does not end the Qshell session. You can redisplay the disconnected Qshell session by running <code>STRQSH</code> again.
F13 (Clear)	Clear the output area.
F14 (Adjust command line length)	Adjust the command line length to four lines. If a number is on the command line, the command line length is adjusted to that number of lines.
F17 (Top)	Display top of output area.
F18 (Bottom)	Display bottom of output area.
F19 (Left)	Shift output area to the left. If a number is on the command line, the output area is shifted by that number of columns.

Function key	Description
F20 (Right)	Shift output area to the right. If a number is on the command line, the output area is shifted by that number of columns.
F21 (Command entry)	Display a command entry window where you can enter CL commands.
SysReq 2	Interrupt the currently running command by sending the SIGINT signal to all child processes.

Running Qshell commands from CL

The Start QSH (STRQSH) command, also known as QSH, is a CL (control language) command that either starts a Qshell interactive session or runs a Qshell command.

If called with the CMD parameter, STRQSH runs the specified Qshell command. The possible values of the CMD parameter are:

*NONE

No command is provided and an “Using a Qshell interactive session” on page 29 is started. If CMD(*NONE) is specified and STRQSH is run in a batch job, STRQSH does nothing.

command

A Qshell command to run. The command can be a maximum of 5000 bytes in length. If a blank or other special characters are used, the command must be enclosed in apostrophes. If an apostrophe is intended, two apostrophes must be used.

When running a command, STRQSH starts **qsh**, runs the specified Qshell command, displays any output generated by the command to the C runtime terminal session, and ends **qsh**. Note that **qsh** does not run any profile files when started to run a command.

You can control what happens to the output by setting the QIBM_QSH_CMD_OUTPUT environment variable. The environment variable can have these values:

STDOUT

Display the output to the C runtime terminal session. This is the default value.

NONE

Throw away any output that is produced.

FILE=pathname

Store the output in the file specified by pathname. The file is truncated before output is written to the file.

FILEAPPEND=pathname

Store the output in the file specified by pathname. The output is appended to end of the file.

When the command ends, STRQSH sends one of three messages:

- QSH0005 when the process running the command ends normally. The message includes the exit status of the process.
- QSH0006 when the process running the command ends by signal. The message includes the signal number.
- QSH0007 when the process running the command ends by exception.

By default, the messages are sent as completion messages. You can have the messages sent as escape messages by setting the environment variable QIBM_QSH_CMD_ESCAPE_MSG. When the value of the environment variable is "Y", the QSH0006, and QSH0007 messages are always sent as escape messages and the QSH0005 message is sent as an escape message if the exit status is greater than zero.

Running Qshell commands from PASE

i5/OS[™] PASE provides a `qsh` command that invokes “`qsh` - Qshell command language interpreter” on page 56 to either run an interactive session or a command. You can use it to run any Qshell command from any i5/OS[™] PASE shell.

Customizing your environment

You can customize your Qshell environment using three profile files. Each profile file is a shell script that can contain any Qshell command. [➤](#) See the “Variables” on page 10 article for the complete list of supported environment variables. [⏪](#)

Global profile file

If the file `/etc/profile` exists, `qsh` runs it in the current environment when you login. It is typically maintained by an administrator to set system-wide defaults for all users. This file should be secured by setting the public authority to read and execute.

Here is a sample `/etc/profile` file:

```
# Sample /etc/profile file
export PATH=/usr/bin:
```

Profile file

If the file `.profile` exists in the user’s home directory, `qsh` runs it in the current environment when you login. It is used to customize your login environment.

Here is a sample `.profile` file.

```
# Sample .profile file
export ENV=$HOME/.qshrc
export PATH=$PATH:$HOME/bin
```

Environment file

If the file specified by the expansion of the `ENV` variable exists, `qsh` runs it in the current environment when starting an interactive shell. The environment file is typically used to set aliases, define functions, or set options for an interactive shell session.

Here is a sample environment file:

```
# Sample environment file
PS1='$PWD'
```

National language support (NLS) considerations

When `qsh` starts, it initializes internal tables for processing commands based on the CCSID of the job. When reading files, `qsh` and many utilities dynamically translate files from the CCSID of the file to the CCSID of the job. For everything to run correctly, you must configure your environment as documented in the tables below.

A locale contains information about a language and country or region, including how to sort and classify characters and the formats for dates, times, numbers, and monetary values. A locale is set by setting the `LANG` environment variable to the path name to a locale object. For example, to set the locale for US English, the `LANG` environment variable is set as follows:

```
LANG=/QSYS.LIB/EN_US.LOCALE
```


» It is best to set the LANG environment variable before starting **qsh**. Some utilities will not work correctly if the locale is not valid for the CCSID and language ID of the job as shown in the tables below.
«

There can be problems in the following situations:

- In an interactive session, if the CCSID of a job is different from the CCSID of the display device, **qsh** does not recognize certain special characters.
- If there is no support for translating between the CCSID of a script file and the CCSID of the job, then the file cannot be opened.

Supported CCSIDs

The following table shows the supported CCSIDs. It is indexed by CCSID number. If a CCSID is not in the table, **qsh** sends message 001-0072 and runs as if it was started in CCSID 37.

Supported CCSIDs		
CCSID	Code Page	Description
00037	00037	USA, Canada
00256	00256	International #1
00273	00273	Germany, Austria
00277	00277	Denmark, Norway
00278	00278	Finland, Sweden
00280	00280	Italy
00284	00284	Spain, Latin America
00285	00285	United Kingdom
00297	00297	France
00424	00424	Israel (Hebrew)
00425	00425	Arabic
00500	00500	Belgium, Canada, Switzerland
00833	00833	Korea Extended Single-byte
00836	00836	Simplified Chinese Extended Single-byte
00838	00838	Thailand Extended
00870	00870	Latin-2 Multilingual
00871	00871	Iceland
00875	00875	Greece
00880	00880	Cyrillic Multilingual
00905	00905	Turkey Extended
00918	00918	Pakistan
00933	00833, 00834	Korea Extended Mixed
00935	00836, 00837	Simplified Chinese Extended Mixed
00937	00037, 00835	Traditional Chinese Extended Mixed
00939	01027, 00300	Japan English Extended Mixed
01025	01025	Cyrillic Multilingual
01026	01026	Turkey
01027	01027	Japan Latin Extended Single-byte

Supported CCSIDs		
CCSID	Code Page	Description
01097	01097	Farsi
01112	01112	Baltic Multilingual
01122	01122	Estonian
01123	01123	Cyrillic Ukraine
01130	01130	Vietnam
01132	01132	Lao
01137	01137	Devanagari
01140	01140	USA, Canada euro
01141	01141	Germany, Austria euro
01142	01142	Denmark, Norway euro
01143	01143	Finland, Sweden euro
01144	01144	Italy euro
01145	01145	Spain, Latin America euro
01146	01146	United Kingdom euro
01147	01147	France euro
01148	01148	Belgium, Canada, Switzerland euro
01149	01149	Iceland euro
01153	01153	Latin-2 Multilingual euro
01154	01154	Cyrillic Multilingual euro
01155	01155	Turkey euro
01156	01156	Baltic Multilingual euro
01157	01157	Estonia euro
01158	01158	Cyrillic Ukraine euro
01160	01160	Thailand Extended euro
01164	01164	Vietnam euro
01388	00836, 00837	Simplified Chinese Host Data Mixed
01399	01399, 00300	Japan English Extended Mixed euro
05035	01027, 00300	Japan English Extended Mixed
05123	01399	Japan English Extended Single-byte euro
09030	00838	Thailand Extended Single-byte
13124	00836	Simplified Chinese Host Data Single-byte
28709	00037	Traditional Chinese Extended

Supported Languages

The following table shows the supported languages. It is indexed by language. In the Language field, the value in parentheses is the value to use for the LANGID parameter of the CHGJOB CL command. In the Country or Region field, the value in parentheses is the value to use for the CNTRYID parameter of the CHGJOB CL command.

Note that there are more valid combinations of Language, Country or Region, CCSID, and Locale than are listed in the table. For example, there is only one entry for the Spanish language even though it is used in more than one country or region.

When running Qshell, the LANGID, CNTRYID, CCSID job attributes must be set to the values listed in the table, and the LANG environment variable must be set to the listed locale.

Supported Languages			
Language	Country or Region Id	CCSID	Locale
Albanian (SQI)	Albania (AL)	00500	/QSYS.LIB/SQ_AL.LOCALE
Arabic (ARA)	Arabic Area (AA)	00425	/QSYS.LIB/AR_AA.LOCALE
Belgian Dutch (NLB)	Belgium (BE)	00500	/QSYS.LIB/NL_BE.LOCALE
Belgian Dutch Euro (NLB)	Belgium (BE)	01148	/QSYS.LIB/NL_BE_E.LOCALE
Belgian French (FRB)	Belgium (BE)	00500	/QSYS.LIB/FR_BE.LOCALE
Belgian French Euro (FRB)	Belgium (BE)	01148	/QSYS.LIB/FR_BE_E.LOCALE
Belgium English (ENB)	Belgium (BE)	00500	/QSYS.LIB/EN_BE.LOCALE
Brazilian Portugese (PTB)	Brazil (BR)	00037	/QSYS.LIB/PT_BR.LOCALE
Bulgarian (BGR)	Bulgaria (BG)	00037	/QSYS.LIB/BG_BG.LOCALE
Canadian French (FRC)	Canada (CA)	00500	/QSYS.LIB/FR_CA.LOCALE
Croatian (HRV)	Croatia (HR)	00870	/QSYS.LIB/HR_HR.LOCALE
Czech (CSY)	Czech Republic (CZ)	00870	/QSYS.LIB/CS_CZ.LOCALE
Danish (DAN)	Denmark (DK)	00277	/QSYS.LIB/DA_DK.LOCALE
Dutch (NLD)	Netherlands (NL)	00037	/QSYS.LIB/NL_NL.LOCALE
Dutch Euro (NLD)	Netherlands (NL)	01140	/QSYS.LIB/NL_NL_E.LOCALE
English Upper Case (ENP)	United States (US)	00037	/QSYS.LIB/EN_UPPER.LOCALE
Estonian (EST)	Estonia (EE)	01122	/QSYS.LIB/ET_EE.LOCALE
Finnish (FIN)	Finland (FI)	00278	/QSYS.LIB/FI_FI.LOCALE
Finnish Euro (FIN)	Finland (FI)	01143	/QSYS.LIB/FI_FI_E.LOCALE
French (FRA)	France (FR)	00297	/QSYS.LIB/FR_FR.LOCALE
French Euro (FRA)	France (FR)	01147	/QSYS.LIB/FR_FR_E.LOCALE
German (DEU)	Germany (DE)	00273	/QSYS.LIB/DE_DE.LOCALE
German Euro (DEU)	Germany (DE)	01141	/QSYS.LIB/DE_DE_E.LOCALE
Greek (ELL)	Greece (GR)	00875	/QSYS.LIB/EL_GR.LOCALE
Hebrew (HEB)	Israel (IL)	00424	/QSYS.LIB/IW_IL.LOCALE
Hungarian (HUN)	Hungary (HU)	00870	/QSYS.LIB/HU_HU.LOCALE
Icelandic (ISL)	Iceland (IS)	00871	/QSYS.LIB/IS_IS.LOCALE
Italian (ITA)	Italy (IT)	00280	/QSYS.LIB/IT_IT.LOCALE
Italian Euro (ITA)	Italy (IT)	01144	/QSYS.LIB/IT_IT_E.LOCALE
Japanese Katakana (JPN)	Japan (JP)	05035	/QSYS.LIB/JA_5035.LOCALE
Japanese Full (JPN)	Japan (JP)	13488	/QSYS.LIB/JA_13488.LOCALE
Korean (KOR)	South Korea (KR)	00933	/QSYS.LIB/KO_KR.LOCALE
Latvian (LVA)	Latvia (LV)	01112	/QSYS.LIB/LV_LV.LOCALE
Lithuanian (LTU)	Lithuania (LT)	01112	/QSYS.LIB/LT_LT.LOCALE
Macedonian (MKD)	Macedonia (MK)	01025	/QSYS.LIB/MK_MK.LOCALE

Supported Languages			
Language	Country or Region Id	CCSID	Locale
Norwegian (NOR)	Norway (NO)	00277	/QSYS.LIB/NO_NO.LOCALE
Polish (PLK)	Poland (PL)	00870	/QSYS.LIB/PL_PL.LOCALE
Portugese (PTG)	Portugal (PT)	00037	/QSYS.LIB/PT_PT.LOCALE
Portugese Euro (PTG)	Portugal (PT)	01140	/QSYS.LIB/PT_PT_E.LOCALE
Romanian (ROM)	Romania (RO)	00870	/QSYS.LIB/RO_RO.LOCALE
Russian (RUS)	Russia (RU)	01025	/QSYS.LIB/RU_RU.LOCALE
Serbian Cyrillic (SRB)	Serbia (SQ)	01025	/QSYS.LIB/SR_SP.LOCALE
Serbian Latin (SRL)	Serbia (SQ)	00870	/QSYS.LIB/SH_SP.LOCALE
Simplified Chinese (CHS)	China (CN)	00935	/QSYS.LIB/ZH_CN.LOCALE
Slovakian (SKY)	Slovakia (SK)	00870	/QSYS.LIB/SK_SK.LOCALE
Slovenian (SLO)	Slovenia (SI)	00870	/QSYS.LIB/SL_SI.LOCALE
Spanish (ESP)	Spain (ES)	00284	/QSYS.LIB/ES_ES.LOCALE
Spanish Euro (ESP)	Spain (ES)	01145	/QSYS.LIB/ES_ES_E.LOCALE
Swedish (SVE)	Sweden (SE)	00278	/QSYS.LIB/SV_SE.LOCALE
Swiss French (FRS)	Switzerland (CH)	00500	/QSYS.LIB/FR_CH.LOCALE
Swiss German (DES)	Switzerland (CH)	00500	/QSYS.LIB/DE_CH.LOCALE
Thai (THA)	Thailand (TH)	00838	/QSYS.LIB/TH_TH.LOCALE
Turkish (TRK)	Turkey (TR)	00905	/QSYS.LIB/TR_TR.LOCALE
Ukrainian (UKR)	Ukraine (UA)	01025	/QSYS.LIB/UK_UA.LOCALE
UK English (ENG)	United Kingdom (GB)	00285	/QSYS.LIB/EN_GB.LOCALE
US English (ENU)	United States (US)	00037	/QSYS.LIB/EN_US.LOCALE

For more details, see the iSeries Globalization IBM Code Pages  .

Performance considerations

The following tips can help improve performance when using **qsh**.

- Do not use command substitutions in the value of the **PS1** variable. This causes a new process to be started every time you press the <enter> key.
- Use input redirection instead of cat. For example, the following command:

```
cat myfile | grep Hello
```

can be replaced with this command:

```
grep Hello < myfile
```
- Use built-in utilities whenever possible because they are run in the current process.
- Leave the **SHELL** variable unset. If a script file does not contain a "#!" on the first line, the script is run in the current activation of **qsh**.

Developing your own utilities

You can develop your own utility programs using any language, although ILE/C, ILE/C++, and Java have the best runtime support. When creating ILE/C or ILE/C++ programs, you should use Integrated File System I/O when creating all of the modules in your utility program.

A utility reads input from standard input or descriptor 0, writes output to standard output or descriptor 1, and writes errors to standard error or descriptor 2.

If your utility program uses the ILE/C or ILE/C++ standard files for I/O, you can run your utility from either the **qsh** command line or the QCMD command line. If your utility reads and writes directly from descriptors 0, 1, and 2, you can only run your utility from the Qshell command line.

Editing files with Qshell Interpreter

You can edit files from any file system using the EDTF CL command. It is an editor that is similar to the Source Entry Utility (SEU) for editing stream files or database files. Also, you can display a stream file or database file using the DSPF CL command.

Another alternative is to connect to the server using iSeries^(TM) Navigator and edit the file using an editor running on the client. The file can be stored in ASCII and still be used by Qshell.

A shell script is just a text file that contains shell commands. It is important to use the right file system for storing shell scripts. Shell scripts are stream data and should be stored in the "root" file system. While it is possible to store shell scripts in source physical files in the QSYS.LIB file system, it causes the shell scripts to use more storage and to run slower.

Differences with other interpreters

While **qsh** is compatible with other standard shell interpreters, there are several differences:

- There is no support for the `<>` redirection operator.
- There is no support for a command history list, the **HISTSIZE** and **HISTFILE** variables, or the **fc** (or **hist**) built-in utility. As an alternative, the QSH CL command has support for command retrieval.
- There is no support for command line editing and the **EDITOR** variable.
- There is no support for the **MAIL**, **MAILCHECK**, and **MAILPATH** variables.
- There is no support for job control. There is no concept of a foreground or background process group on i5/OS^(TM). This means it is possible for multiple jobs to be reading from the terminal at the same time. **qsh** does not support:
 - The **fg** or **bg** built-in utilities.
 - Using the *Suspend* key (typically `<ctrl>z`) to send the SIGTSTP signal to the foreground process group.
 - Using the *Stop* key (typically `<ctrl>s`) to send the SIGSTOP signal to the foreground process group.
 - Using the *Restart* key (typically `<ctrl>q`) to send the SIGCONT signal to the foreground process group.
 - Using the *Interrupt* key (typically `<ctrl>c`) to send the SIGINT signal to the foreground process group. As an alternative, you can use SysReq 2 from an interactive shell session to send the SIGINT signal to the shell interpreter process and any currently running child processes.
- There is no support for the *End-of-file* key (typically `<ctrl>d`). As an alternative, use a here-document to redirect text entered at the command line to standard input of a utility.
- When calling a program, there is a limit to the maximum number of parameters you can pass to the command. If the program was built for a release before V5R3, the limit is 255 parameters. If the program was built for V5R3 or a subsequent release, the limit is 65535 parameters.
- When using path name expansion with some case insensitive file systems, you must use upper case characters in the pattern. For example, to list all of the program objects in the QSHELL library you should use this command:

```
ls /qsys.lib/qshell.lib/*.PGM.
```

Utilities

Qshell utilities are available for accomplishing the following tasks:

- “Utilities for defining aliases” on page 49
- “Utilities for running commands” on page 50
- “Utilities for managing data” on page 64
- “Utilities for DB2 Universal Database” on page 80
- “Utilities for working with files and directories” on page 80
- “Utilities for reading and writing input and output” on page 131
- “Utilities for developing Java™ programs” on page 135
- “Utilities for managing jobs” on page 144
- “Utilities for Kerberos credentials and key tables” on page 153
- “Utilities for LDAP directory server” on page 153
- “Utilities for working with parameters and variables” on page 154
- “Utilities for writing scripts” on page 162
- “Miscellaneous utilities” on page 168

List of all utilities

This alphabetical list of all the utilities is available so that you can easily go directly to the utility you need.

A (page 38) B (page 38) C (page 39) D (page 40) E (page 40) F (page 41) G (page 41) H (page 41) I (page 42) J (page 42) K (page 43) L (page 43) M (page 44) N (page 44) O (page 44) P (page 44) Q (page 45) R (page 45) S (page 46) T (page 47) U (page 47) W (page 48) X (page 48) Z (page 48)

A

- “ajar - Alternative Java archive tool
Alternative
Java™ archive”
on page 136
- “alias - Define or display aliases
display aliases”
on page 49
- “appletviewer - Run applets without a web browser
View Java™
applet” on page
139
- “attr - Get or set attributes for files
attributes for
files” on page 81

B

- “basename - Return non-directory portion of path name
Return
non-directory
portion of path
name” on page
86
- “break - Exit from for, while, or until loop
from for, while,
or until loop” on
page 162

"builtin - Run a shell built-in utility" on page 51

C

"cat - Concatenate and print files" on page 86

"cat - Concatenate and print spool files" on page 87

"cd - Change working directory" on page 88

"chgrp - Change file group permission ownership" on page 89

"chmod - Change file modes (permissions)" on page 90

"chown - Change file ownership" on page 93

"clrtmp - Clear the /tmp directory" on page 168

"cmp - Compare two files" on page 64

"colon (:) - Null utility" on page 162

"command - Run a simple command" on page 51

"compress - Compress data" on page 94

"continue - Continue for, while, or until loop" on page 163

"cp - Copy files" on page 95

"cut - Cut out selected fields of each line of a file" on page 65

D

“dataq - Send or receive messages from i5/OS^(TM) data queue” on page 169

“datarea - Read or write i5/OS^(TM) data area” on page 170

“date - Write the date and time” on page 170

db2profcl DB2 SQLJ profile customizer

db2profp Print DB2 customized version of SQLJ profile

“declare - Declare variables and set attributes” on page 154

“dirname - Return directory portion of path name” on page 97

“dot (.) - Run commands in current environment” on page 52

“dspmsg - Display message from message catalog” on page 131

E

“echo - Write arguments to standard output” on page 132

“egrep - Search a file for an extended regular expression pattern” on page 66

“env - Set environment for command invocation” on page 52

“eval - Construct command by concatenating arguments” on page 53

“exec - Run commands and open, close, or copy descriptors” on page 53 Run commands and open, close, or copy descriptors

“exit - Exit from the shell” on page 54 Exit from the shell

“export - Set export attribute for variables” on page 155 Set export attribute for variables

“expr - Evaluate arguments as an expression” on page 172 Evaluate arguments as an expression

“extcheck - A utility to detect JAR conflicts” on page 139 Detect Java archive conflicts

F

“false - Return false value” on page 163 Return false value

“fgrep - Search a file for a fixed string pattern” on page 66 Search a file for a fixed string pattern

“file - Determine file type” on page 97 Determine file type

“find - Find files” on page 98 Find files

G

“gencat - Generate a formatted message catalog” on page 101 Generate a formatted message catalog

“getconf - Get configuration values” on page 102 Get configuration values

“getjobid - Display job information” on page 144 Display job information

“getopts - Parse utility options” on page 163 Parse utility options

“grep - Search a file for a pattern” on page 66 Search a file for a pattern

H

"hash - Remember or report utility locations
 Remember or report utility locations" on page 145
 "head - Copy the first part of files" on page 104
 "help - Display information for built-in utility" on page 55
 "hostname - Display the name of the current host system" on page 173

I

"iconv - Convert characters from one CCSID to another CCSID" on page 68
 "id - Return user identity" on page 173
 "ipcrm - Remove interprocess communication identifier" on page 174
 "ipcs - Report interprocess communication status" on page 175

J

"jar - Archive Java™ files" on page 139
 "jarsigner - JAR signing and verification" on page 140
 "java - Run Java™ interpreter" on page 140
 "javac - Compile a Java™ program" on page 141
 "javadoc - Generate Java™ documentation" on page 141

“javah - Generate C header or stub file” on page 141	Generate C header or stub file
“jvakey - Manage Java™ security keys and certificates” on page 142	Manage Java security keys and certificates
“javap - Disassemble a compiled Java™ program” on page 142	Disassemble a compiled Java program
“jobs - Display status of jobs in current session” on page 146	Display status of jobs in the current session
K	
kdestroy	Destroy a Kerberos credentials cache
keytab	Manage a Kerberos key table file
“keytool - Key and certificate management tool” on page 142	Key and certificate management tool
“kill - Terminate or signal processes” on page 147	End or signal processes
kinit	Obtain or renew a Kerberos ticket-granting ticket
klist	Display the contents of a Kerberos credentials cache or key table file
ksetup	Manage Kerberos service entries in the LDAP directory for a Kerberos realm
L	
ldapadd	Add LDAP entry tool
ldapchangepwd	Change LDAP Password tool
ldapdelete	Delete LDAP entry tool
ldapdiff	Compare LDAP replication synchronization tool
ldapexop	Extend LDAP operation tool
ldapmodify	Change LDAP entry tool
ldapmodrdn	Change LDAP Relative Distinguished Name (RDN) tool
ldapsearch	Search LDAP server tool
“let - Evaluate arithmetic expression” on page 164	Evaluate arithmetic expression
“liblist - Manage library list” on page 148	Manage library list
“ln - Link files” on page 104	Link files
“local - Assign a local variable in a function” on page 156	Assign a local variable in a function

"locale - Get locale specific information" on page 180
"logger - Log messages" on page 181
"logname - Display user's login name" on page 182
"ls - List directory contents" on page 105

M

"mkdir - Make directories" on page 108
"mkfifo - Make FIFO special files" on page 109
"mv - Move files" on page 110

N

"native2ascii - Convert native characters to ASCII" on page 143
"nohup - Run utility without hangups" on page 55

O

"od - Dump files in various formats" on page 110

P

"pax - Portable archive interchange" on page 112
"policytool - Policy file creation and management tool" on page 143
"pr - Print files" on page 119

“print - Write output” on page 133	Write output
“printenv - Display values of environment variables” on page 157	Display values of environment variables
“printf - Write formatted output” on page 133	Write formatted output
profconv	Convert SQLJ serialized profile instance to Java class
profdb	SQLJ profile auditor installer
profp	Print SQLJ profile
“ps - Display process status” on page 149	Display process status
“pwd - Return working directory name” on page 121	Return working directory name
“pwdx - Print working directory expanded” on page 121	Return working directory expanded

Q

“qsh - Qshell command language interpreter” on page 56	Qshell command language interpreter
--	-------------------------------------

R

“read - Read a line from standard input” on page 135	Read a line from standard input
“readonly - Set read-only attribute for variables” on page 157	Set read-only attribute for variables
“return - Return from a function” on page 164	Return from a function
“rexec - Run remote command” on page 57	Run remote command
“rexx - Run REXX procedure” on page 58	Run REXX procedure

"Rfile - Read or write record files" on page 122 Read or write record files
 "rm - Remove directory entries" on page 123 Remove directory entries
 "rmdir - Remove directories" on page 124 Remove directories
 "rmic - Compile Java[™] RMI stubs" on page 143 Compile Java RMI stubs
 "rmid - The Java[™] RMI activation system" on page 143 Java RMI activation system
 "rmiregistry - Start a remote object registry" on page 143 Start a remote object registry

S

"sed - Stream editor" on page 69 Stream editor
 "serialver - Return serial version" on page 143 Return serial version
 "set - Set or unset options and positional parameters" on page 158 Set or unset options and positional parameters
 "setccsid - Set CCSID attribute for file" on page 125 Set CCSID attribute for a file
 "qsh - Qshell command language interpreter" on page 56 Qshell command language interpreter
 "shift - Shift positional parameters" on page 160 Shift positional parameters
 "sleep - Suspend invocation for an interval" on page 151 Suspend invocation for an interval
 "sort - Sort, merge, or sequence check text files" on page 74 Sort, merge, or sequence check text files

"source - Run commands in current environment" on page 59 Run commands in the current environment
 "split - Split files into pieces" on page 75 Split files into pieces
 sqlj Structured query language for Java (SQLJ) translator
 "system - Run CL command" on page 59 Run CL command
 "sysval - Retrieve system values or network attributes" on page 182 Retrieve system values or network attributes

T

"tail - Display the last part of a file" on page 125 Copy the last part of a file
 "tar - File archiver" on page 126 File archiver
 "tee - Duplicate standard input" on page 182 Duplicate standard input
 "test - Evaluate expression" on page 165 Evaluate expression
 "tnameserv - Naming service" on page 144 Naming service
 "touch - Change file access and modification times" on page 128 Change file access and modification times
 "tr - Translate characters" on page 76 Translate characters
 "trap - Trap signals" on page 151 Trap signals
 "true - Return true value" on page 167 Return true value
 "type - Find type of command" on page 61 Find type of command
 "typeset - Declare variables and set attributes" on page 160 Declare variables and set attributes

U

"ulimit - Set or display resource limits" on page 183 Set or display resource limits
 "umask - Get or set the file mode creation mask" on page 129 Get or set the file mode creation mask
 "unalias - Remove alias definitions" on page 50 Remove alias definitions
 "uname - Return system name" on page 184 Return system name
 "uncompress - Expand compressed data" on page 130 Expand compressed data
 "uniq - Report or filter out repeated lines in a file" on page 78 Report or filter out repeated lines in a file
 "unset - Unset values of variables and functions" on page 161 Unset values and attributes of variables and functions

W

"wait - Wait for process completion" on page 152 Wait for process completion
 "wc - Word, line and byte/character count" on page 79 Word, line and byte/character count
 "whence - Determine how command is interpreted" on page 62 Determine how command is interpreted

X

"xargs - Construct argument lists and invoke utility" on page 62 Construct argument lists and invoke utility

Z

“zcat - Expand and concatenate data and concatenate data” on page 131

Utilities for defining aliases

The following are Qshell utilities for defining aliases:

- “alias - Define or display aliases”
- “unalias - Remove alias definitions” on page 50

alias - Define or display aliases

Synopsis

```
alias [ -p ] [ name [ =value ] ... ]
```

Description

The **alias** utility defines an alias *name* that has the specified *value*. If only *name* is specified, **qsh** displays the name and value of the alias.

When no arguments are specified, **qsh** displays a list of all the aliases and their values.

qsh defines these default aliases:

- float='declare -E'
- functions='declare -f'
- integer='declare -i'

Options

-p Precede each line of the output with the word "alias " so it is displayed in a re-enterable format.

Operands

Each *name* specifies an alias in the current environment. If a *value* is also specified, then the value of the alias is updated.

Exit Status

- 0 when successful.
- >0 when unsuccessful. The value is the number of *names* that are not aliases.

Related information

- “unalias - Remove alias definitions” on page 50

Examples

1. Define an alias to list the contents of a directory:
alias ll='ls -l'
2. Display the value of the ll alias:
alias ll
3. Display the values of all currently defined aliases:
alias

unalias - Remove alias definitions

Synopsis

unalias *name* ...

unalias -a

Description

You can use **unalias** to remove the *names* from the list of defined aliases.

Options

-a Remove all aliases

Operands

Each *name* is a defined alias.

Exit Status

- 0 when successful.
- >0 when unsuccessful. The value is the number of *names* that are not aliases.

Related information

- “alias - Define or display aliases” on page 49

Examples

1. Remove the ll alias: **unalias ll**

Utilities for running commands

The following are Qshell utilities for running commands:

- “builtin - Run a shell built-in utility” on page 51
- “command - Run a simple command” on page 51
- “dot (.) - Run commands in current environment” on page 52
- “env - Set environment for command invocation” on page 52
- “eval - Construct command by concatenating arguments” on page 53
- “exec - Run commands and open, close, or copy descriptors” on page 53
- “exit - Exit from the shell” on page 54
- “help - Display information for built-in utility” on page 55
- “nohup - Run utility without hangups” on page 55
- “qsh - Qshell command language interpreter” on page 56
- “rexec - Run remote command” on page 57
- “rexx - Run REXX procedure” on page 58
- “qsh - Qshell command language interpreter” on page 56
- “source - Run commands in current environment” on page 59
- “system - Run CL command” on page 59
- “type - Find type of command” on page 61
- “whence - Determine how command is interpreted” on page 62
- “xargs - Construct argument lists and invoke utility” on page 62

builtin - Run a shell built-in utility

Synopsis

```
builtin [ utility [ argument ... ] ]
```

Description

The **builtin** utility runs the shell built-in *utility* with the specified *arguments*. You can use **builtin** to run a built-in utility from a shell function of the same name.

Operands

The *utility* is the name of a shell built-in utility (page 22). You can use “command - Run a simple command,” “type - Find type of command” on page 61, or “whence - Determine how command is interpreted” on page 62 to determine the type of a utility

Exit Status

- The exit status of the *utility*
- 1 if *utility* is not a built-in utility

Related information

- “command - Run a simple command”
- “type - Find type of command” on page 61
- “whence - Determine how command is interpreted” on page 62

command - Run a simple command

Synopsis

```
command [ -p ] command_name [ argument ... ]
```

```
command [ -vV ] command_name
```

Description

You can use **command** to run *command_name* with the specified *arguments* with functions eliminated from the search order. If *command_name* is a special built-in utility, then it is treated as a regular built-in utility. Otherwise, the effect of **command** is the same as omitting **command**.

Note that **command -v** is equivalent to **whence** and **command -V** is equivalent to **whence -v**.

Options

- p** Perform the command search using a default value for the **PATH** variable that is guaranteed to find all of the standard utilities.
- v** Write a string that shows the path name or command used by **qsh** to invoke *command_name* in the current environment.
- V** Write a string that shows how *command_name* is interpreted by **qsh** in the current environment.

Operands

command_name is a utility in the current environment.

Exit Status

- 0 when successful.

- >0 when unsuccessful.

Related information

- “dot (.) - Run commands in current environment”
- “eval - Construct command by concatenating arguments” on page 53
- “whence - Determine how command is interpreted” on page 62

Examples

1. Run the export special built-in utility as a regular built-in utility: **command** export ALPHA
2. Display the path name used to invoke the ls utility: **command** -v ls
3. Display how the reserved word for is interpreted: **command** -V for

dot (.) - Run commands in current environment

Synopsis

. name [argument ...]

Description

You can use **dot** to run a script or function in the current environment.

Options

None.

Operands

If *name* refers to a function, **qsh** runs the function in the current environment. Otherwise, **qsh** uses the search path specified by the **PATH** variable to find *name*. If *name* is found, **qsh** reads the contents of the file and runs those commands in the current environment.

If specified, the *arguments* replace the positional parameters while *name* is running. Otherwise the positional parameters are unchanged.

Exit Status

Exit status of last command in *name*.

Related information

- “command - Run a simple command” on page 51
- “eval - Construct command by concatenating arguments” on page 53
- “exec - Run commands and open, close, or copy descriptors” on page 53

env - Set environment for command invocation

Synopsis

env [-i | -] [*name=value ...*] [*utility* [*argument ...*]]

Description

The **env** utility obtains the current environment, modifies it according to the arguments, and then invokes the specified *utility*. Any *arguments* are passed to the *utility*. If no *utility* is specified, the resulting environment is written to standard output with one *name=value* per line.

Options

- Invoke the *utility* with exactly the environment specified on the command. The inherited environment is ignored completely.
- i Same as '-'.

Operands

name=value

This modifies the run-time environment and is placed into the inherited environment before the *utility* is invoked.

utility The name of the command or utility to be invoked.

argument

A string to pass to the invoked command or utility.

Exit Status

- 0 when successful
- >0 when an error occurs

eval - Construct command by concatenating arguments

Synopsis

eval [*argument ...*]

Description

You can use **eval** to construct a command by concatenating *arguments* together, each separated by a <space>. **qsh** then reads and runs the constructed command.

Options

None.

Operands

Each *argument* is expanded twice, once to construct the command and once when the constructed command is run.

Exit Status

Exit status of the constructed command.

Related information

- “command - Run a simple command” on page 51
- “dot (.) - Run commands in current environment” on page 52

exec - Run commands and open, close, or copy descriptors

Synopsis

exec [-c] [*command* [*argument ...*]]

Description

The **exec** utility replaces **qsh** with *command* without creating a new process. The specified *arguments* are arguments to *command*. Any redirections affect the current environment.

When a *command* is not specified, any redirections are processed in the current environment. Any file descriptors greater than 2 that are opened by a redirection are not inherited when **qsh** invokes another program.

Options

-c Run *command* with an empty set of environment variables.

Operands

Each *argument* is assigned in order to the positional parameters of *command*.

Exit Status

Zero if no *command* is specified. Otherwise it does not return to **qsh**.

Related information

- “command - Run a simple command” on page 51
- “dot (.) - Run commands in current environment” on page 52
- “eval - Construct command by concatenating arguments” on page 53
- “nohup - Run utility without hangups” on page 55
- “print - Write output” on page 133
- “read - Read a line from standard input” on page 135

Examples

1. Open a file for reading on descriptor 5:
exec 5<\$/HOME/input
2. Close descriptor 5:
exec 5<&-

exit - Exit from the shell

Synopsis

exit [*n*]

Description

You can use **exit** to end the shell and return to the program that called **qsh**.

Options

None.

Operands

The value of *n* is an integer that is greater than or equal to 0 and less than or equal to 255.

Exit Status

n if specified. Otherwise, the exit status of the preceding command.

Related information

- “return - Return from a function” on page 164

help - Display information for built-in utility

Synopsis

help [*utility* ...]

Description

The **help** utility displays a usage message for the specified built-in *utility*. If no arguments are specified, **help** displays the list of all built-in utilities.

Operands

The *utility* is the name of a shell built-in utility (page 22).

Exit Status

- 0 when successful
- >0 if *utility* is not a built-in utility

Related information

- “builtin - Run a shell built-in utility” on page 51
- “command - Run a simple command” on page 51
- “type - Find type of command” on page 61
- “whence - Determine how command is interpreted” on page 62

nohup - Run utility without hangups

Synopsis

nohup [**-C** *ccsid*] *utility* [*argument* ...]

Description

The **nohup** utility runs the specified *utility* with the specified *arguments*. When *utility* is invoked the SIGHUP signal is set to be ignored. You can use **nohup** to allow *utility* to run even after ending the “Using a Qshell interactive session” on page 29.

If standard output is a terminal, all output written by *utility* to its standard output is appended to the file nohup.out in the current directory. If the file cannot be created or opened for appending, all output is appended to the file \$HOME/nohup.out. If neither file can be created or opened, *utility* is not run. The default permission for the nohup.out file allows only the owner to read and write the file.

If standard error is a terminal, all output written by *utility* to its standard error is redirected to the same descriptor as standard output.

Options

-C *ccsid*

The nohup.out file is created with the specified *ccsid* and all data written to the file is converted from the CCSID of the job to the specified *ccsid*. This option overrides the value of the QIBM_CCSID environment variable.

Operands

The *utility* is the name of a regular utility in the current environment.

Environment Variables

nohup is affected by the following environment variables:

QIBM_CCSID

The value of the environment variable is the CCSID used to create the nohup.out file. All data written to the file is converted from the CCSID of the job to the specified CCSID.

Exit Status

- 126 when *utility* was found but could not be run
- 127 when *utility* was not found or there was an error in **nohup**
- Otherwise, the exit status of *utility*

Related information

- “command - Run a simple command” on page 51
- “env - Set environment for command invocation” on page 52

qsh - Qshell command language interpreter

Synopsis

qsh [-abCefijlmntuvx] [-o *option*] *command_file* arg ...

qsh -c [-abCefijlmntuvx] [-o *option*] *command_string*

qsh -s [-abCefijlmntuvx] [-o *option*] arg ...

Description

The **qsh** utility is the Qshell command language interpreter. In the first synopsis form, **qsh** reads the specified *command_file* and runs the commands contained in the file. In the second synopsis form, **qsh** runs the specified *command_string* and ends. In the third synopsis form, **qsh** reads commands from standard input.

Options

The **a, b, C, e, f, F, j, l, m, n, -o option t, u, v,** and **x** options are described in “set - Set or unset options and positional parameters” on page 158.

- c** Run the command specified in *command_string* and exit.
- i** The shell is interactive. If there are no operands and standard input is connected to a terminal, the **-i** option is set by default.
- s** Read commands from standard input. If there are no operands and the **-c** option is not specified, the **-s** option is set by default.

Operands

The *command_file* is the pathname of a regular file that contains Qshell commands. If the pathname does not contain a slash (/) character, **qsh** searches for *command_file* using the **PATH** variable. The special parameter 0 is set to the value of *command_file*. Each *arg* is a positional parameter.

The *command_string* is any Qshell command, including compound commands.

Exit Status

- 0 when successful.
- 1 when unsuccessful.
- 2 when an error occurred in a script.

- 3 when there was an unexpected exception in a root shell.
- 4 when there was an unexpected exception in an exception handler for a root shell.
- 5 when there was an unexpected exception in a child shell.
- 6 when there was an unexpected exception in an exception handler for a child shell.
- 7 when descriptor 0 was not available.
- 8 when descriptor 1 was not available.
- 9 when descriptor 2 was not available.
- 10 when there was an error opening the message catalog.
- 11-125 when unsuccessful.
- 126 when a command was found but could not be invoked.
- 127 when a command cannot be found.
- >128 when a command was ended by a signal. The value is 128 plus the signal number.

Related information

- “exit - Exit from the shell” on page 54
- “set - Set or unset options and positional parameters” on page 158
- “Command language” on page 8

rexec - Run remote command

Synopsis

» **rexec** [-C *ccsid*] [-p *password*] [-u *user*] [-i] *host command* «

Description

The **rexec** utility runs the specified *command* on the remote system specified by *host*. The remote system must be running a rexec server to process the commands. By default, **rexec** prompts for a valid user name and password for the remote system. The user name and password are not encrypted when they are sent to the remote system.

The standard output and standard error generated by *command* on the remote system are written to standard output and standard error on the local system. Any data read from standard input on the local system is sent to standard input for the *command* running on the remote system » if the **-i** option is not specified. «

By default, the data sent to and from the remote system is encoded in CCSID 819. The CCSID used to encode the data can be specified with either the **-C** option or the QIBM_CCSID variable. If the CCSID value is 65535, then no conversion is done on the data.

Options

-C *ccsid*

Encode the data sent to and from the remote system in the specified *ccsid*. This option overrides the value of the QIBM_CCSID environment variable.

» **-i** Ignore standard input on the local system. «

-p *password*

The password for the user on *host*.

-u *user* A valid user name on *host*.

Operands

The *host* is the name of the remote system where the command is run. The *command* is a command string that is interpreted by the rexec server running on the remote system.

Environment Variables

rexec uses the following environment variables:

QIBM_CCSID

The value of the variable is the CCSID to use to encode the data sent to and from the remote system.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “exec - Run commands and open, close, or copy descriptors” on page 53

rex - Run REXX procedure

Synopsis

```
rex [ -c cmdenv ] [ -t type ] path [ arg ... ]
```

Description

The **rex** utility runs the REXX procedure specified by *path* with the specified *arguments*. For more information about programming with REXX, see the REXX information.

The REXX interpreter cannot read REXX commands from standard input. It can only run REXX procedures stored in members of database files in the QSYS.LIB file system. The interactive debug feature of the REXX interpreter is not supported by the **rex** utility.

The program /QSYS.LIB/QSHELL.LIB/QZSHSHRX.PGM implements the Qshell command environment for REXX procedures. The Qshell command environment sets the REXX return code and condition as follows:

- When the the shell command ends normally with an exit status of zero, the REXX return code is set to zero and no condition is raised.
- If the shell command ends normally with an exit status that is non-zero, the REXX return code is set to the exit status value and the ERROR condition is raised.
- If the shell command ends by signal, the REXX return code is set to the signal number + 128 and the FAILURE condition is raised.
- If the shell command ends by exception, the REXX return code is set to the exception number from wait() and the FAILURE condition is raised.

Options

-c *cmdenv*

Set the command environment program to process commands for the REXX procedure. If the option is not specified, the default value is *command*. The *cmdenv* can be one of the following values:

- *command* for the i5/OS^(TM) CL command environment.
- *cpicomm* for the Common Programming Interface for communications command environment.
- *execsql* for the Structured Query Language (SQL) command environment.
- *qsh* for the Qshell command environment.

- *path* to specify the path to the command environment program. The path must specify a program in the QSYS.LIB file system.
- t type** Control tracing for the REXX procedure. If the option is not specified, the default value is *normal*. The *type* can be one of the following values:
- *all* to trace all clauses before processing.
 - *commands* to trace host commands before processing and display any error return codes.
 - *error* to trace host commands after processing that result in an error return code.
 - *failure* to trace host commands after processing that result in a failure along with the return code.
 - *intermediates* to trace all clauses before processing along with intermediate results during the evaluation of expressions.
 - *labels* to trace labels during processing.
 - *normal* to trace host commands after processing that result in a failure.
 - *off* to turn off all tracing.
 - *results* to trace all clauses before processing.

Operands

The *path* is the path name of the REXX procedure. On i5/OS^(TM), a REXX procedure can only be stored in the QSYS.LIB file system.

Exit Status

- 0 when successful
- 1 when there is an error running the REXX procedure
- >1 when unsuccessful

Related information

- REXX information
- “system - Run CL command”

source - Run commands in current environment

Synopsis

source *name* [*argument* ...]

Description

You can use **source** to run a script or function in the current environment. It is a synonym for the “dot (.) - Run commands in current environment” on page 52 utility.

Related information

- “command - Run a simple command” on page 51
- “dot (.) - Run commands in current environment” on page 52
- “eval - Construct command by concatenating arguments” on page 53
- “exec - Run commands and open, close, or copy descriptors” on page 53

system - Run CL command

Synopsis

system [-iKknpqsv] *CLcommand* [*arg* ...]

Description

The **system** utility runs a CL command. Any spool file output generated by *CLcommand* is written to standard output. By default, the spool files are deleted after they are written and the job log of the job running **system** is deleted.

Any messages generated by *CLcommand* are written to standard error. By default, all messages generated by *CLcommand* are written using the following format:

MsgId: Text

where “MsgId” is the seven character i5/OSTM message identifier (for example CPF0001) and “Text” is the text of the message. Use the **-n** option to not include the “MsgId” prefix.

By default, **system** checks the number of threads running in the job. If there is more than one thread running, it starts a second job and runs *CLcommand* in the second job. Use the **-i** option to force **system** to always run *CLcommand* in the current job.

Options

- i** Always run *CLcommand* in the current job and set the exit status to the ILE return code of the program called by *CLcommand*. Note that some CL commands do not run in a multi-thread capable job or when there are multiple threads running in the job.
- K** Keep all spool files generated by *CLcommand* and the job log of the job running **system**. If this option is not specified, all spool files are deleted after they are written and the job log is deleted.
- k** Keep all spool files generated by *CLcommand*. If this option is not specified, all spool files are deleted after they are written.
- n** Do not include the message identifier when writing the messages to standard error. Only the message text of the messages are written to standard error. This option is ignored if the **-q** option is also specified.
- p** Only write the messages sent to the program’s message queue by *CLcommand* to standard error. This option is ignored if the **-q** option is also specified.
- q** Do not write messages generated by *CLcommand* to standard error.
- s** Do not write spool files generated by *CLcommand* to standard output.
- v** Write the complete command string to standard output before running it.

Note that for compatibility with the PASE system utility, **system** does not return an error if the **-b**, **-e**, **-E**, **-I**, or **-O** options are specified, but the options are ignored.

Operands

Each *arg* is a parameter to *CLcommand*. You may need to enclose *CLcommand* and *args* in quotes to prevent **qsh** from expanding any special characters in them. Both CL and **qsh** use some of the same special characters, for example, the asterisk (*) character.

Environment Variables

The **system** utility is affected by the following environment variables:

QIBM_SYSTEM_ALWMLTTHD

Set this environment variable to control how the **system** utility behaves in a multi-thread capable job. If the value of the variable is “N”, **system** starts a new job to run the CL command when the current job is multi-thread capable even if there is only one thread running in the job. There is no default value.

QIBM_SYSTEM_USE_ILE_RC

Set this environment variable to control how the **system** utility sets the exit status. If the value of the variable is "Y", **system** sets the exit status to the ILE return code of the program called by *CLcommand*, or zero if the program did not set a return code. There is no default value. The environment variable is ignored if the **-i** option is specified.

Exit Status

- 0 when *CLcommand* is successful
- >0 when *CLcommand* is unsuccessful or when set by the program called by *CLcommand*

When the **-i** option is specified or the environment variable QIBM_SYSTEM_USE_ILE_RC=Y is set, **system** sets the exit status to the ILE return code of the program called by *CLcommand*, or zero if the program did not set a return code.

Related Information

- CL command finder
- system - Run a CL command for i5/OS^(TM) PASE

Examples

1. List all of the active jobs:
system wrkactjob
2. Create a test library:
system "CRTLIB LIB(TESTDATA) TYPE(*TEST)"
3. Delete a library and do not write any messages:
system -q "DLTLIB LIB(TESTDATA)"

type - Find type of command

Synopsis

type [**-apt**] *name* ...

Description

The **type** utility displays the type of each specified *name*. The *name* can be an alias, function, special shell built-in, shell built-in, reserved word, or file.

Options

- a** Show all uses for *name*.
- p** Do not check to see if *name* is a reserved word, a built-in utility, an alias, or a function.
- t** Display a one word description for the type of *name*.

Operands

Each *name* is a utility in the current environment.

Exit Status

- 0 when every *name* is found
- >0 when unsuccessful

Related information

- "command - Run a simple command" on page 51

- “whence - Determine how command is interpreted”

whence - Determine how command is interpreted

Synopsis

whence [**-afpv**] *name* ...

Description

The **whence** utility displays how each specified *name* is interpreted. The name can be an alias, function, special shell built-in, shell built-in, reserved word, or file.

Note that **whence** is equivalent to **command -v** and **whence -v** is equivalent to **command -V**.

Options

- a Show all uses for *name*.
- f Do not check to see if *name* is a function.
- p Do not check to see if *name* is a reserved word, a built-in utility, an alias, or a function.
- v Display the type of *name*.

Operands

Each *name* is a utility in the current environment.

Exit Status

- 0 when every *name* is found
- >0 when unsuccessful

Related information

- “command - Run a simple command” on page 51
- “dot (.) - Run commands in current environment” on page 52
- “eval - Construct command by concatenating arguments” on page 53
- “type - Find type of command” on page 61

Examples

1. Find the type of the reserved word for:

```
whence -v for
```

xargs - Construct argument lists and invoke utility

Synopsis

xargs [**-t**] [**-e**[*eofstring*]] [**-E** *eofstring*] [**-l**[*number*]] [**-L** *number*] [**-n** *number* [**-x**]] [**-s** *size*] [*utility* [*arguments* ...]]

Description

The **xargs** utility reads space, tab, newline and end-of-file delimited *arguments* from the standard input and runs the specified *utility* with them as arguments.

The *utility* and any *arguments* specified on the command line are given to the *utility* upon each invocation, followed by some number of the *arguments* read from standard input. The *utility* is repeatedly run until standard input is exhausted.

Spaces, tabs and newlines may be embedded in arguments using single (') or double (") quotation marks or backslashes (\). Single quotation marks escape all non-single quotation mark characters, excluding newlines, up to the matching single quotation marks. Double quotation marks escape all non-double quotation mark characters, excluding newlines, up to the matching double quotation marks. Any single character, including newlines, may be escaped by a backslash.

If no *utility* is specified, **echo** is used by default.

Undefined behavior may occur if *utility* reads from the standard input.

The **xargs** utility exits immediately (without processing any further input) if a command line cannot be assembled, *utility* cannot be invoked, an invocation of the *utility* is ended by a signal, or an invocation of the *utility* exits with a value of 255.

Options

-E *eofstring*

Specify a logical end-of-file string. **xargs** reads standard input until either end-of-file or the logical end-of-file string is encountered.

-e[*eofstring*]

This option is equivalent to the **-E** option. If *eofstring* is not specified, the default value is `_` (a single underscore).

-L *number*

Run *utility* for each non-empty *number* lines of arguments read from standard input. The last invocation of *utility* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first newline character unless the last character of the line is a blank character. A trailing blank character signals continuation to the next non-empty line, inclusive. The **-L** and **-n** options are mutually exclusive. The last one specified takes effect.

-l[*number*]

This option is equivalent to the **-L** option. If *number* is not specified, the default value is 1.

-n *number*

Set the maximum number of *arguments* read from standard input for each invocation of the *utility*. An invocation of *utility* will use less than *number* standard input arguments if the number of bytes accumulated (see the **-s** option) exceeds the specified size or there are fewer than *number* arguments remaining for the last invocation of *utility*. The maximum number of arguments i5/OS^(TM) can pass to a program is 255. The default value for *number* is 250. The **-n** and **-L** options are mutually exclusive. The last one specified takes effect.

-s *size* Set the maximum number of bytes for the command line length provided to *utility*. The sum of the length of the utility name and the arguments passed to *utility* (including NULL terminators) will be less than or equal to *size*. The default value for *size* is 16 252 928 bytes.

-t Turn on trace mode. The command to be run is written to standard error immediately before it is run.

-x Force **xargs** to end immediately if a command line containing *number* arguments will not fit in the specified (or default) command line length.

Exit Status

- 0 when all invocations of *utility* returned exit status 0.
- 1-125 when at least one invocation of *utility* returned a non-zero exit status or there was an error.
- 126 when *utility* was found but could not be invoked.
- 127 when *utility* cannot be found.
- >128 when *utility* was ended by a signal. The value is 128 plus the signal number.

Related information

- “echo - Write arguments to standard output” on page 132
- “eval - Construct command by concatenating arguments” on page 53
- “find - Find files” on page 98

Utilities for managing data

The following are Qshell utilities for managing data:

- “cmp - Compare two files”
- “cut - Cut out selected fields of each line of a file” on page 65
- “egrep - Search a file for an extended regular expression pattern” on page 66
- “fgrep - Search a file for a fixed string pattern” on page 66
- “grep - Search a file for a pattern” on page 66
- “iconv - Convert characters from one CCSID to another CCSID” on page 68
- “sed - Stream editor” on page 69
- “sort - Sort, merge, or sequence check text files” on page 74
- “split - Split files into pieces” on page 75
- “tr - Translate characters” on page 76
- “uniq - Report or filter out repeated lines in a file” on page 78
- “wc - Word, line and byte/character count” on page 79

cmp - Compare two files

Synopsis

```
cmp [-l | -s] [-t] file1 file2 [skip1 [skip2]]
```

Description

You can use **cmp** to compare two files. By default, a byte for byte binary comparison is done. If no differences are found, no output is written. If no option flags are specified, **cmp** writes a message with the byte and line number of the first difference and exits with an error. Bytes and lines are numbered beginning with 1.

Options

- l (Lower case ell) Write the byte number in decimal and the differing bytes in octal for all differences.
- s Silent mode where no output is written for differing files; only the exit status is set.
- t Text mode where the files are opened in text mode and translated to the CCSID of the job before comparing byte for byte.

Operands

The *file1* and *file2* operands are the two files to be compared byte for byte. The optional *skip1* and *skip2* are the number of bytes to skip from the beginning of each file, respectively, before the comparison begins.

» Environment Variables

cmp is affected by the following environment variables:

QIBM_CMP_FILE_SIZE

Controls the maximum file size in bytes that **cmp** reads into an internal buffer for better performance. For files larger than the maximum size, **cmp** reads the files one byte at a time.



Exit Status

- 0 when the files are identical
- 1 when the files are different
- >1 when an error occurred

Related information

- “sed - Stream editor” on page 69
- “sort - Sort, merge, or sequence check text files” on page 74
- “split - Split files into pieces” on page 75
- “uniq - Report or filter out repeated lines in a file” on page 78

Examples

1. Find the exact position where two files differ. It is better to place the reference or good file first and then the changed or new file second.

```
cmp myApplet.java.old myApplet.java.new
```

cut - Cut out selected fields of each line of a file

Synopsis

```
cut -b list [file ...]
```

```
cut -c list [file ...]
```

```
cut -f list [-d string] [-s] [file ...]
```

Description

The **cut** utility selects portions of each line as specified by *list* from each *file* (or the standard input by default), and writes them to the standard output. The items specified by *list* can be in terms of column position or in terms of fields delimited by a special character. Column numbering starts from 1.

The *list* is a comma or whitespace separated set of increasing numbers and/or number ranges. Number ranges consist of a number, a dash (-), and a second number and select the fields or columns from the first number to the second, inclusive. Numbers or number ranges may be preceded by a dash, which selects all fields or columns from 1 to the first number. Numbers or number ranges may be followed by a dash, which selects all fields or columns from the last number to the end of the line. Numbers and number ranges may be repeated, overlapping, and in any order. It is not an error to select fields or columns not present in the input line.

Options

-b *list* The *list* specifies byte positions.

-c *list* The *list* specifies character positions.

-d *string*

Use the first character of *string* as the field delimiter character instead of the tab character.

-f *list* The *list* specifies fields, delimited in the input by a single tab character. Output fields are separated by a single tab character.

- s Suppresses lines with no field delimiter characters. Unless specified, lines with no delimiters are passed through unmodified.

Exit Status

- 0 on success
- 1 if an error occurred.

Related information

- “grep - Search a file for a pattern”
- “tr - Translate characters” on page 76
- “wc - Word, line and byte/character count” on page 79

egrep - Search a file for an extended regular expression pattern

Synopsis

```
egrep [-c|-l|-q] [-ihnsvwxy] [-e pattern_list] [-f pattern_file] [pattern] [file ...]
```

Description

The **egrep** utility is equivalent to running the **grep** utility with the **-E** option. See “grep - Search a file for a pattern” for the complete description.

Related information

- “fgrep - Search a file for a fixed string pattern”
- “grep - Search a file for a pattern”

fgrep - Search a file for a fixed string pattern

Synopsis

```
fgrep [-c|-l|-q] [-ihnsvwxy] [-e pattern_list] [-f pattern_file] [pattern] [file ...]
```

Description

The **fgrep** utility is equivalent to running the **grep** utility with the **-F** option. See “grep - Search a file for a pattern” for the complete description.

Related information

- “egrep - Search a file for an extended regular expression pattern”
- “grep - Search a file for a pattern”

grep - Search a file for a pattern

Synopsis

```
grep [-E|-F] [-c|-l|-q] [-R [-H | -L | -P] ] [-ihnsvwxy] [-e pattern_list] [-f pattern_file] [pattern] [file ...]
```

Description

The **grep** utility searches the given input *files* selecting lines which match one or more *patterns*. The type of patterns is controlled by the options specified. By default, a pattern matches an input line if any regular expression (RE) in the pattern matches the input line without its trailing newline. A null RE matches every line. Each input line that matches at least one of the patterns is written to the standard output.

If **-E** and **-F** options are both specified, the last one specified is used.

Options

- E** Use Extended Regular Expressions (ERE).
- F** Do not recognize regular expressions.
- H** If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed.
- L** If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P** If the **-R** option is specified, no symbolic links are followed.
- R** If *file* designates a directory, **grep** searches each file in the entire subtree connected at that point.
- c** Only a count of selected lines is written to standard output.
- e** *pattern_list* specifies one or more search patterns. Each pattern should be separated by a newline character.
- f** *pattern_file* specifies a file containing search patterns. Each pattern should be separated by a newline character.
- h** Do not print filename headers.
- i** The case of letters is ignored in making comparisons. That is, upper and lower case are considered identical.
- l** Only the names of files containing selected lines are written to standard output. Pathnames are listed once per file searched. If the standard input is searched, the pathname "-" is written.
- n** Each output line is preceded by its relative line number in the file; each file starting at line 1. The line number counter is reset for each file processed. This option is ignored if the **-c**, **-l**, or **-s** options are specified.
- q** Quiet mode where no messages are printed. Only the exit status is returned.
- s** Suppress the error messages ordinarily written for nonexistent or unreadable files. Other messages are not suppressed.
- v** Selected lines are those not matching the specified patterns.
- w** The expression is searched for as a whole word (as if surrounded by "[[:<:]]" and "[[:>:]]").
- x** Match line if pattern is the **only** thing on the line. This option takes precedence over the **-w** option. If both are specified, the **-w** option is ignored.
- y** Ignore case (same as **-i**).

Operands

Each *file* specifies the path to a text file. If no *file* operands are specified, the standard input is used.

Exit Status

- 0 when one or more lines were selected.
- 1 when no lines were selected.
- >1 when an error occurred.

Extended Regular Expressions (ERE)

The following characters are interpreted by **grep**:

- \$** Align the match from the end of the line.

- ^ Align the match from the beginning of the line. (NOTE: This character may not work correctly from a 5250 terminal session.)
- | Add another pattern (see example below).
- ? Match one or less sequential repetitions of the pattern.
- + Match one or more sequential repetitions of the pattern.
- * Match zero or more sequential repetitions of the pattern.
- .
- [] Match any single character or range of characters enclosed in the brackets.

Escape special characters which have meaning to **grep**, that is, the set of {`.,^,[,|,?,+,*,(,)`}.

Related information

- “cut - Cut out selected fields of each line of a file” on page 65
- “egrep - Search a file for an extended regular expression pattern” on page 66
- “fgrep - Search a file for a fixed string pattern” on page 66
- “tr - Translate characters” on page 76
- “wc - Word, line and byte/character count” on page 79

Examples

1. Find all occurrences of the word `patricia` in a file.


```
grep patricia myfile
```
2. Find all occurrences of the pattern `".Pp"` at the beginning of a line. The single quotation marks assure the entire expression is evaluated by **grep** instead of by the shell. The carat (^) means from the beginning of a line.


```
grep '^Pp' myfile
```
3. Find either 19, 20 or 25 in the file `calendar`.


```
grep '19|20|25' calendar
```
4. Find the total number of lines that matches a character in the range of "a" to "z".


```
grep -c '[a-z]' reference/alphabet.text
```
5. Display all lines that have a dollar sign (\$) character in them. You must escape the dollar sign character so **grep** will not interpret the character. Also, display the line number as well as the line that contains the match.


```
grep -n '\$' valid.file
```

iconv - Convert characters from one CCSID to another CCSID

Synopsis

```
iconv -f fromCCSID -t toCCSID [ file ... ]
```

Description

The **iconv** utility converts the encoding of characters read from either standard input or the specified *file* from one CCSID to another CCSID and then writes the results to standard output. The input data is assumed to be in the CCSID specified by the *fromCCSID* parameter. If *file* is not specified, the **iconv** utility reads from standard input.

You must specify valid i5/OS^(TM) CCSIDs with a supported conversion for the *fromCCSID* and *toCCSID* parameters.

Options

-f *fromCCSID*

The input data is encoded in the *fromCCSID*.

-t *toCCSID*

The output data is encoded in the *toCCSID*.

Operands

The *file* operand specifies a path name to a regular file.

Exit Status

- 0 when successful
- 1 when the conversion is not supported or there is an error with *file*
- 2 when there is an error during the conversion

Related information

- “locale - Get locale specific information” on page 180
- “tr - Translate characters” on page 76

sed - Stream editor

Synopsis

» **sed** [-an] [-C *ccsid*] *command file* ...

sed [-an] [-C *ccsid*] [-e *command*] [-f *command_file*] *file* ... ◀◀

Description

The **sed** utility reads the specified *files*, or the standard input if no files are specified, modifying the input as specified by a list of *commands*. The input is then written to the standard output.

A single *command* may be specified as the first argument to **sed**. Multiple commands may be specified by using the **-e** or **-f** options. All commands are applied to the input in the order they are specified regardless of their origin.

Options

-a By default, the files listed as parameters for the **w** functions are created (or truncated) before any processing begins. The **-a** option causes **sed** to delay opening each file until a command containing the related **w** function is applied to a line of input.

» **-C** *ccsid*

Any files created by **sed** are created with the CCSID specified by *ccsid*. This option overrides the value of the QIBM_CCSID environment variable. ◀◀

-e *command*

Append the editing commands specified by the *command* argument to the list of commands.

-f *command_file*

Append the editing commands found in the file *command_file* to the list of commands. The editing commands should each be listed on a separate line.

-n

By default, each line of input is echoed to the standard output after all of the commands have been applied to it. The **-n** option suppresses this behavior.

Operands

The form of a **sed** command is as follows:

`[address[,address]]function[arguments]`

White space may be inserted before the first *address* and the *function* portions of the command.

Normally, **sed** cyclically copies a line of input, not including its terminating newline character, into a "pattern space", (unless there is something left after a **D** function), applies all of the commands with addresses that select that pattern space, copies the pattern space to the standard output, appending a newline, and deletes the pattern space.

Some of the functions use a "hold space" to save all or part of the pattern space for subsequent retrieval.

Extended Description

sed Addresses

An address is not required, but if specified must be:

- a number that counts input lines cumulatively across input files,
- a dollar (\$) character that addresses the last line of input, or
- a context address which consists of a regular expression preceded and followed by a delimiter.

A command line with no addresses selects every pattern space.

A command line with one address selects all of the pattern spaces that match the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. If the second address is a number less than or equal to the line number first selected, only that line is selected. Starting at the first line following the selected range, **sed** starts looking again for the first address.

Editing commands can be applied to non-selected pattern spaces by use of the exclamation character (!) function.

sed Regular Expressions

sed regular expressions are basic regular expressions. In addition, **sed** has the following two additions to basic regular expressions:

- In a context address, any character other than a backslash (\) or newline character may be used to delimit the regular expression. Also, putting a backslash character before the delimiting character causes the character to be treated literally. For example, in the context address `\xabc\xdefx`, the regular expression delimiter is an *x* and the second *x* stands for itself, so that the regular expression is `abcxdef`.
- The escape sequence `\n` matches a newline character embedded in the pattern space. You can't, however, use a literal newline character in an address or in the substitute command.

One special feature of **sed** regular expressions is that they can default to the last regular expression used. If a regular expression is empty, that is, just the delimiter characters are specified, the last regular expression encountered is used instead. The last regular expression is defined as the last regular expression used as part of an address or substitute command, and at run-time, not compile-time. For example, the command:

```
/abc/s//XXX/
```

will substitute XXX for the pattern abc.

sed Functions

In the following list of commands, the maximum number of permissible addresses for each command is indicated by *[0addr]*, *[1addr]*, or *[2addr]*, representing zero, one, or two addresses.

The argument *text* consists of one or more lines. To embed a newline in the text, precede it with a backslash. Other backslashes in *text* are deleted and the following character taken literally.

The **r** and **w** functions take an optional *file* parameter, which should be separated from the function letter by white space. Each file given as an argument to **sed** is created (or its contents truncated) before any input processing begins.

The **b**, **r,s**, **t,w,y,!,** and **&** functions all accept additional arguments. The following synopses indicate which arguments have to be separated from the function letters by white space characters.

Two of the functions take a *function-list*. This is a list of **sed** functions separated by newlines, as follows:

```
{ function
  function
  ...
  function
}
```

The { can be preceded by white space and can be followed by white space. The function can be preceded by white space. The terminating } must be preceded by a newline or optional white space.

[2addr] *function-list*

Execute *function-list* only when the pattern space is selected.

[1addr]**a** \ *text*

Write *text* to standard output immediately before each attempt to read a line of input, whether by executing the **N** function or by beginning a new cycle.

[2addr]**b**[*label*]

Branch to the **&** function with the specified *label*. If the *label* is not specified, branch to the end of the script.

[2addr]**c** \ *text*

Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, *text* is written to the standard output.

[2addr]**d**

Delete the pattern space and start the next cycle.

[2addr]**D**

Delete the initial segment of the pattern space through the first newline character and start the next cycle.

[2addr]**g**

Replace the contents of the pattern space with the contents of the hold space.

[2addr]**G**

Append a newline character followed by the contents of the hold space to the pattern space.

[2addr]**h**

Replace the contents of the hold space with the contents of the pattern space.

[2addr]**H**

Append a newline character followed by the contents of the pattern space to the hold space.

[1addr]**i** \ *text*

Write *text* to the standard output.

[2addr]l

(The letter ell.) Write the pattern space to the standard output in a visually unambiguous form. This form is as follows:

- backslash (\)
- alert (\a)
- form-feed (\f)
- newline (\n)
- carriage-return (\r)
- tab (\t)
- vertical tab (\v)

Nonprintable characters are written as three-digit octal numbers (with a preceding backslash) for each byte in the character (most significant byte first). Long lines are folded, with the point of folding indicated by displaying a backslash followed by a newline. The end of each line is marked with a dollar sign (\$).

[2addr]n

Write the pattern space to the standard output if the default output has not been suppressed, and replace the pattern space with the next line of input.

[2addr]N

Append the next line of input to the pattern space, using an embedded newline character to separate the appended material from the original contents. Note that the current line number changes.

[2addr]p

Write the pattern space to standard output.

[2addr]P

Write the pattern space, up to the first newline character to the standard output.

[1addr]q

Branch to the end of the script and quit without starting a new cycle.

[1addr]r file

Copy the contents of *file* to the standard output immediately before the next attempt to read a line of input. If *file* cannot be read for any reason, it is silently ignored and no error condition is set.

[2addr]s/regular_expression/replacement/flags

Substitute the *replacement* string for the first instance of the *regular_expression* in the pattern space. Any character other than backslash or newline can be used instead of a slash to delimit the *regular_expression* and the *replacement*. Within the *regular_expression* and the *replacement*, the regular expression delimiter itself can be used as a literal character if it is preceded by a backslash.

An ampersand (&) appearing in the *replacement* is replaced by the string matching the regular expression. The special meaning of & in this context can be suppressed by preceding it by a backslash. The string *#*, where *#* is a digit, is replaced by the text matched by the corresponding backreference expression.

A line can be split by substituting a newline character into it. To specify a newline character in the replacement string, precede it with a backslash.

The value of *flags* in the substitute function is zero or more of the following:

- 0 ... 9** Make the substitution only for the N'th occurrence of the regular expression in the pattern space.
- g** Make the substitution for all non-overlapping matches of the regular expression, not just the first one.

- p** Write the pattern space to standard output if a replacement was made. If the replacement string is identical to that which it replaces, it is still considered to have been a replacement.
- w file** Append the pattern space to *file* if a replacement was made. If the replacement string is identical to that which it replaces, it is still considered to have been a replacement.

[2addr]t [label]

Branch to the **:** function bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t** function. If no *label* is specified, branch to the end of the script.

[2addr]w file

Append the pattern space to the *file*.

[2addr]x

Swap the contents of the pattern and hold spaces.

[2addr]y/string1/string2/

Replace all occurrences of characters in *string1* in the pattern space with the corresponding characters from *string2*. Any character other than a backslash or newline can be used instead of a slash to delimit the strings. Within *string1* and *string2*, a backslash followed by any character other than a newline is that literal character, and a */n* is replaced by a newline character.

[2addr]!function

[2addr]!function-list

Apply the *function* or *function-list* only to the lines that are **not** selected by the address(es).

[0addr]:label

This function does nothing; it bears a *label* to which the **b** and **t** commands may branch.

[1addr]=

Write the line number to the standard output followed by a newline character.

[0addr]

Empty lines are ignored.

[0addr]#

The **#** and the remainder of the line are ignored (treated as a comment), with the single exception that if the first two characters in the file are **#n**, the default output is suppressed. This is the same as specifying the **-n** option on the command line.

Environment Variables

sed is affected by the following environment variables:

QIBM_CCSD

Any files created by **sed** are created with the CCSID specified by the value of the environment variable.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- “**cmp** - Compare two files” on page 64
- “**sort** - Sort, merge, or sequence check text files” on page 74
- “**split** - Split files into pieces” on page 75

- “uniqu - Report or filter out repeated lines in a file” on page 78

sort - Sort, merge, or sequence check text files

Synopsis

sort [-cmubdfinr] [-t *char*] [-T *char*] [-k *keydef* ...] [-o *output*] [*file*] ...

Description

The **sort** utility sorts text files by lines. Comparisons are based on one or more sort keys extracted from each line of input, and are performed lexicographically. By default, if keys are not given, **sort** regards each input line as a single field.

Options

- c Check that the single input file is sorted. If the file is not sorted, **sort** produces the appropriate error messages and exits with code 1. Otherwise, **sort** returns 0. This option produces no output.
- m Merge only; the input files are assumed to be presorted.
- o *output*
The *output* argument is the name of an output file to be used instead of the standard output. This file can be the same as one of the input files.
- u Unique processing to suppress all but one in each set of lines having equal keys. If used with the -c option, check that there are no lines with duplicate keys.

The following options override the default ordering rules. When ordering options appear independent of key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key, the ordering options override all global ordering options for that key.

- d Only blank space and alphanumeric characters are used in making comparisons.
- f Considers all lowercase characters that have uppercase equivalents to be the same for purposes of comparison.
- i Ignore all non-printable characters.
- n An initial numeric string, consisting of optional blank space, optional minus sign, and zero or more digits (including decimal point) is sorted by arithmetic value.
- r Reverse the sense of comparisons.

The treatment of field separators can be altered using the options:

- b Ignores leading blank space when determining the start and end of a restricted sort key. A -b option specified before the first -k option applies globally to all -k options. Otherwise, the -b option can be attached independently to each field argument of the -k option (see below). Note that the -b option has no effect unless key fields are specified.
- t *char* The *char* argument is used as the field separator character. The initial *char* is not considered to be part of a field when determining key offsets (see below). Each occurrence of *char* is significant (for example, "char-char" delimits an empty field). If -t is not specified, blank space characters are used as default field separators.
- T *char*
The *char* argument is used as the record separator character. This option should be used with discretion. The -T option with an alphanumeric *char* typically produces undesirable results. The default line separator is newline.
- k *keydef*
Select the key fields to use for sorting. *keydef* as the format:

field_start[*type*][,*field_end*[*type*]]

where *field_start* is the starting position and *field_end* is the optional ending position of a key field. If *field_end* is not specified, the ending position is the end of the line. The *type* is a character from the set of characters b, d, f, i, n, r. The *type* behaves the same as the corresponding option but only to the specified key field. If no **-k** option is specified, a default sort key is used. A maximum of nine **-k** options can be specified.

Operands

The path name of a file to be sorted, merged, or checked. If no *file* operands are specified, the standard input is used.

Extended Description

A field is defined as a minimal sequence of characters followed by a field separator or a newline character. By default, the first blank space of a sequence of blank spaces acts as the field separator. All blank spaces in a sequence of blank spaces are considered as part of the next field. For example, all blank spaces at the beginning of a line are considered to be part of the first field.

Fields are specified by the **-k** *field_start*[*type*][,*field_end*[*type*]] option.

The *field_start* portion of the option argument has the form:

field_number[,*first_character*]

Fields and characters within fields are numbered starting with 1. The *field_number* and *first_character* are positive decimal integers and specify the first character to be used as part of a sort key. If *first_character* is not specified, it refers to the first character of the field.

The *field_end* portion of the option argument has the form:

field_number[.*last_character*]

The *field_number* is a positive decimal integer and *last_character* is a non-negative decimal integer. If *last_character* is not specified or is zero, it refers to the last character of the field.

If the **-b** option or the **b** type modifier is in effect, characters in fields are counted from the first non-blank character.

Exit Status

- 0 normal behavior.
- 1 on disorder (or non-uniqueness) with the **-c** option
- 2 an error occurred

Related information

- “**cmp** - Compare two files” on page 64
- “**sed** - Stream editor” on page 69
- “**split** - Split files into pieces”
- “**uniq** - Report or filter out repeated lines in a file” on page 78

split - Split files into pieces

Synopsis

split [**-b** *byte_count*[**k**|**m**]] [**-l** *line_count*] [*file* [*prefix*]]

Description

The **split** utility reads the given *file* (or standard input if no file is specified) and breaks it up into files of 1000 lines each.

Options

- b** Create files that are *byte_count* bytes in length. If **k** is appended to the number, the file is split into *byte_count* kilobyte pieces. If **m** is appended to the number, the file is split into *byte_count* megabyte pieces.
- l** Create files that are *line_count* lines in length.

Operands

If additional arguments are specified, the first is used as the name of the input file which is to be split. If a second additional argument is specified, it is used as a prefix for the names of the files into which the file is split. In this case, each file into which the *file* is split is named by the prefix followed by a lexicographically ordered suffix in the range of "aa-zz". If the *prefix* argument is not specified, the default prefix is "x". The maximum number of possible output file names is 676.

Exit Status

- 0 if successful
- >0 if an error occurs

Related information

- "cmp - Compare two files" on page 64
- "sed - Stream editor" on page 69
- "sort - Sort, merge, or sequence check text files" on page 74
- "uniq - Report or filter out repeated lines in a file" on page 78

Examples

1. Split the file `jdk_v11.jar` into files that are 1.44MB in size and use the prefix `"jdk_v11."` for the output files.

```
split -b1440k jdk_v11.jar jdk_v11.
```
2. Split the file `myapp.java` into files of 100 lines each.

```
split -l 100 myapp.java
```

tr - Translate characters

Synopsis

```
tr [-cs] string1 string2
```

```
tr [-c] -d string1
```

```
tr [-c] -s string1
```

```
tr [-c] -ds string1 string2
```

Description

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters.

In the first synopsis form, the characters in *string1* are translated into the characters in *string2* where the first character in *string1* is translated into the first character in *string2* and so on. If *string1* is longer than *string2*, the last character found in *string2* is duplicated until *string1* is exhausted.

In the second synopsis form, the characters in *string1* are deleted from the input.

In the third synopsis form, the characters in *string1* are compressed as described for the **-s** option below.

In the fourth synopsis form, the characters in *string1* are deleted from the input, and the characters in *string2* are compressed as described for the **-s** option below.

The following conventions can be used in *string1* and *string2* to specify sets of characters. Any character not described by one of the following conventions represents itself.

nnn A backslash (\) followed by 1, 2 or 3 octal digits represents a character with that encoded value.

char To follow an octal sequence with a digit as a character, left zero-pad the octal sequence to the full 3 octal digits. A backslash (\) followed by certain special characters maps to special values. The special characters and their values are:

- a - alert character
- b - backspace
- f - form-feed
- n - newline
- r - carriage return
- t - tab
- v - vertical tab
- A backslash (\) followed by any other character maps to that character.

c-c Represents the range of characters between the range endpoints, inclusively.

[:class:]

Represents all characters belonging to the defined character class. These are the class names:

- alnum - alphanumeric characters
- alpha - alphabetic characters
- cntrl - control characters
- digit - numeric characters
- graph - graphic characters
- lower - lower-case alphabetic characters
- print - printable characters
- punct - punctuation characters
- space - space characters
- upper - upper-case characters
- xdigit - hexadecimal characters

Note:

With the exception of the upper and lower classes, characters in the classes are in unspecified order. In the upper and lower classes, characters are entered in ascending order.

Options

-c Complement the set of characters in *string1*, that is **-c ab** includes every character except for "a" and "b".

- d Delete characters from the input.
- s Squeeze multiple occurrences of the characters listed in the last operand (either *string1* or *string2*) in the input into a single instance of the character. This occurs after all deletion and translation is completed.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- “cut - Cut out selected fields of each line of a file” on page 65
- “grep - Search a file for a pattern” on page 66
- “tr - Translate characters” on page 76
- “wc - Word, line and byte/character count” on page 79

Examples

1. Create a list of the words in file1, one per line, where a word is taken to be a maximal string of letters.

```
tr -cs '[:alpha:]' 'n' < file1
```

2. Translate the contents of file1 to upper-case.

```
tr '[:lower:]' '[:upper:]' < file1
tr 'a-z' 'A-Z' < file1
```

3. Remove the non-printable characters from file1.

```
tr -cd '[:print:]' < file1
```

uniq - Report or filter out repeated lines in a file

Synopsis

```
uniq [-c | -du] [-f fields] [-s chars] [input_file [output_file]]
```

Description

The **uniq** utility reads the standard input comparing adjacent lines, and writes a copy of each unique input line to the standard output. The second and succeeding copies of identical adjacent input lines are not written. Repeated lines in the input will not be detected if they are not adjacent, so it may be necessary to sort the files first.

Options

- c Precede each output line with the count of the number of times the line occurred in the input, followed by a single space.
- d Suppress the writing of lines that are not repeated in the input.
- f *fields* Ignore the first *fields* fields in each input line when doing comparisons. A field is a string of non-blank characters separated from adjacent fields by blanks. Field numbers are one based, so the first field is field one.
- s *chars* Ignore the first *chars* characters in each input line when doing comparisons. If specified in conjunction with the -f option, the first *chars* characters after the first *fields* fields will be ignored. Character numbers are one based, so the first character is character one.
- u Suppress the writing of lines that are repeated in the input.

Operands

If additional arguments are specified on the command line, the first such argument is used as the name of an input file, the second is used as the name of an output file.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- “`cmp` - Compare two files” on page 64
- “`sed` - Stream editor” on page 69
- “`split` - Split files into pieces” on page 75
- “`sort` - Sort, merge, or sequence check text files” on page 74

Examples

In the following examples, the contents of example file are:

```
There are 5 apples
There are 9 oranges
There are 9 oranges
There are 2 pears
```

1. Display the unique lines in the file “fruit”.

```
uniq fruit
```

```
There are 5 apples
There are 9 oranges
There are 2 pears
```

2. Display the lines that repeat in the file “fruit”.

```
uniq -d fruit
```

```
There are 9 oranges
```

3. Display a list of how many times a line is repeated in the file “fruit”.

```
uniq -c fruit
```

```
1 There are 5 apples
2 There are 9 oranges
1 There are 2 pears
```

wc - Word, line and byte/character count

Synopsis

```
wc [-c | -m] [-lw] [file ...]
```

Description

The `wc` utility displays the number of lines, words, and bytes contained in each input *file* (or standard input, by default) to standard output. A line is defined as a string of characters delimited by a newline character. A word is defined as a string of characters delimited by white space characters. If more than one input file is specified, a line of cumulative counts for all the files is displayed on a separate line after the output for the last file.

Options

- c** Write to standard output the number of bytes in each input file.

- l** Write to standard output the number of lines in each input file.
- m** Write to standard output the number of characters in each input file.
- w** Write to standard output the number of words in each input file.

Operands

When an option is specified, **wc** only reports the information requested by that option. The default action is equivalent to specifying all of the flags.

If no *files* are specified, the standard input is used and no file name is displayed.

Exit Status

- 0 when successful
- >0 when an error occurred

Related information

- “cut - Cut out selected fields of each line of a file” on page 65
- “grep - Search a file for a pattern” on page 66
- “tr - Translate characters” on page 76

Utilities for DB2 Universal Database

The following are Qshell utilities for working with the DB2 Universal Database^(TM):

- db2profc - DB2^(R) SQLJ profile customizer
- db2profp - Print DB2 customized version of SQLJ profile
- profconv - Convert SQLJ serialized profile instance to Java^(TM) class
- profdb - SQLJ profile auditor installer
- profp - Print SQLJ profile
- sqlj - Structured query language for Java (SQLJ) translator

Utilities for working with files and directories

The following are Qshell utilities for working with files and directories:

- “attr - Get or set attributes for files” on page 81
- “basename - Return non-directory portion of path name” on page 86
- “cat - Concatenate and print files” on page 86
- “cat splf - Concatenate and print spool files” on page 87
- “cd - Change working directory” on page 88
- “chgrp - Change file group ownership” on page 89
- “chmod - Change file modes” on page 90
- “chown - Change file ownership” on page 93
- “compress - Compress data” on page 94
- “cp - Copy files” on page 95
- “dirname - Return directory portion of path name” on page 97
- “file - Determine file type” on page 97
- “find - Find files” on page 98
- “gencat - Generate a formatted message catalog” on page 101
- “getconf - Get configuration values” on page 102
- “head - Copy the first part of files” on page 104

- “ln - Link files” on page 104
- “ls - List directory contents” on page 105
- “mkdir - Make directories” on page 108
- “mkfifo - Make FIFO special files” on page 109
- “mv - Move files” on page 110
- “od - Dump files in various formats” on page 110
- “pax - Portable archive interchange” on page 112
- “pr - Print files” on page 119
- “pwd - Return working directory name” on page 121
- “pwdx - Print working directory expanded” on page 121
- “Rfile - Read or write record files” on page 122
- “rm - Remove directory entries” on page 123
- “rmdir - Remove directories” on page 124
- “setccsid - Set CCSID attribute for file” on page 125
- “tail - Display the last part of a file” on page 125
- “tar - File archiver” on page 126
- “touch - Change file access and modification times” on page 128
- “umask - Get or set the file mode creation mask” on page 129
- “uncompress - Expand compressed data” on page 130
- “zcat - Expand and concatenate data” on page 131

attr - Get or set attributes for files

Synopsis

attr [**-hp**] *file* [*attribute* [*=value*] ...]

Description

The **attr** utility gets or sets attributes for the object specified by *file*. When no *attributes* are specified, **attr** displays all of the attributes for the object in a re-entrable format to standard output. When an *attribute* is specified, **attr** displays the value of the attribute to standard output. When an *attribute* and *value* are specified, **attr** sets the attribute to the value. Note that all attributes can be displayed, but only some attributes can be set.

For date and time attributes, the value displayed by default is formatted with the `asctime()` function. If the `LC_TIME` environment variable is set, the value displayed is formatted with the format specified by the `d_t_fmt` keyword in the `LC_TIME` category of the specified locale.

See the `Qp0IGetAttr()` API for detailed information about the attributes and their values.

Options

- h** Display or set the attributes of a symbolic link instead of the object pointed to by the symbolic link.
- p** Display the *attribute* in an re-entrable format.

Operands

The *file* operand specifies a path name to an object. The *attribute* operand can have the following values:

`ACCESS_TIME`

The date and time the object was last accessed. This attribute can only be displayed.

ALLOC_SIZE

The number of bytes allocated for the object displayed as a 32-bit number. This attribute can only be displayed.

ALLOC_SIZE_64

The number of bytes allocated for the object displayed as a 64-bit number. This attribute can only be displayed.

ALWCKPWRT

An indicator if a stream file can be shared with readers and writers during the save-while-active checkpoint processing. This attribute can be displayed or set.

ALWSAV

An indicator of whether the object can be saved or not. This attribute can be displayed or set.

ASP The auxiliary storage pool in which the object is stored. This attribute can only be displayed.

AUDIT

The auditing value associated with the object. This attribute can only be displayed.

AUTH_GROUP

The name of the user profile that is the primary group for the object. This attribute can only be displayed.

AUTH_LIST_NAME

The name of the authorization list used to secure the object. This attribute can only be displayed.

AUTH_OWNER

The name of the user profile that is the owner of the object. This attribute can only be displayed.

AUTH_USERS

The list of user profiles that are authorized to use the object. This attribute can only be displayed.

CCSID The coded character set identifier (CCSID) of the object. This attribute can be displayed or set.

CHANGE_TIME

The date and time the object's data or attributes were last changed. This attribute can only be displayed.

CHECKED_OUT

An indicator if the object is checked out. This attribute can only be displayed.

CHECKED_OUT_USER

The user profile that has the object checked out. This attribute can only be displayed.

CHECKED_OUT_TIME

The date and time that the object was checked out. This attribute can only be displayed.

CODEPAGE

The code page derived from the coded character set identifier (CCSID) of the object. This attribute can be displayed or set.

CREATE_TIME

The date and time the object was created. This attribute can only be displayed.

» *CRTOBJAUD*

The create object auditing value associated with a directory. The auditing value is given to any objects created in the directory. This attribute can be displayed or set. «

CRTOBJSCAN

An indicator of whether the objects created in a directory will be scanned when exit programs are registered with any of the integrated file system scan-related exit points. This attribute can be displayed or set.

DATA_SIZE

The size in bytes of the data in the object displayed as a 32-bit number. This attribute can only be displayed.

DATA_SIZE_64

The size in bytes of the data in the object displayed as a 64-bit number. This attribute can only be displayed.

DIR_FORMAT

An indicator of the format of a directory object. This attribute can only be displayed.

DISK_STG_OPT

An indicator of how auxiliary storage is allocated by the system for the object. This attribute can be displayed or set.

EXTENDED_ATTR_SIZE

The number of bytes used for extended attributes for the object. This attribute can only be displayed.

FILE_FORMAT

The format of the stream file. This attribute can only be displayed.

FILE_ID

The file identifier of the object if the object is stored in the "root" (/), the QOpenSys, or a user-defined file system. This attribute can only be displayed.

JOURNAL_APPLY_CHANGES

An indicator of whether the object was restored with partial transactions which requires an Apply Journalled Changes (APYJRNCHG) command to complete the transaction. This attribute can only be displayed.

JOURNAL_ID

The journal identifier that can be used on journal-related commands and APIs. This attribute can only be displayed.

JOURNAL_LIBRARY

If the object is journalled, the library containing the currently used journal. If the object is not journalled, the library containing the last used journal. This attribute can only be displayed.

JOURNAL_NAME

If the object is journalled, the name of the currently used journal. If the object is not journalled, the name of the last used journal. This attribute can only be displayed.

JOURNAL_OPTIONS

The current journaling options. This attribute can only be displayed.

JOURNAL_RCVR_ASP

The name of the ASP for the library that contains the journal receiver. This attribute can only be displayed.

JOURNAL_RCVR_LIBRARY

The name of the library that contains the journal receiver. This attribute can only be displayed.

JOURNAL_RCVR_NAME

The name of the oldest journal receiver needed to successfully Apply Journalled Changes (APYJRNCHG). This attribute can only be displayed.

JOURNAL_ROLLBACK_ENDED

An indicator of whether the object had rollback ended before completion of a request to roll back a transaction. This attribute can only be displayed.

JOURNAL_START_TIME

The date and time that journaling was last started for the object. This attribute can only be displayed.

JOURNAL_STATUS

An indicator if the object is currently journaled. This attribute can only be displayed.

LOCAL_REMOTE

An indicator if the object is on the local system or a remote system. This attribute can only be displayed.

MAIN_STG_OPT

An indicator of how main storage is allocated and used by the system for the object. This attribute can be displayed or set.

MODIFY_TIME

The date and time that the object's data was last modified. This attribute can only be displayed.

MULT_SIGS

An indicator if the object has more than one i5/OS^(TM) digital signature. This attribute can only be displayed.

OBJTYPE

A text string describing the type of the object. This attribute can only be displayed.

PC_ARCHIVE

An indicator if the object has changed since the last time the file was examined. This attribute can be displayed or set.

PC_HIDDEN

An indicator if the object is hidden. This attribute can be displayed or set.

PC_READ_ONLY

An indicator if the object is read-only. This attribute can be displayed or set.

PC_SYSTEM

An indicator if the object is a system object. This attribute can be displayed or set.

RSTDRNMUNL

An indicator of whether renames and unlinks are restricted for objects within a directory. Objects can be linked into a directory that has this attribute set on, but cannot be renamed or unlinked from it unless the user has the appropriate authority. This attribute can be displayed or set.

SCAN An indicator of whether the object will be scanned when exit programs are registered with any of the integrated file system scan-related exit points. This attribute can be displayed or set.

SCAN_BINARY

An indicator of whether the object has been scanned in binary mode when it was previously scanned. This attribute can only be displayed.

SCAN_CCSID1

If an object has been scanned in text mode, the first CCSID used when it was previously scanned. This attribute can only be displayed.

SCAN_CCSID2

If an object has been scanned in text mode, the second CCSID used when it was previously scanned. This attribute can only be displayed.

SCAN_SIGS_DIFF

An indicator of whether the scan signature for the object is different from the global scan signature. This attribute can only be displayed.

SCAN_STATUS

The scan status for the object. This attribute can only be displayed.

SGID An indicator if the effective group ID is set at run time. This attribute can be displayed or set.

SIGNED

An indicator if the object has an i5/OS^(TM) digital signature. This attribute can only be displayed.

STG_FREE

An indicator if the data is moved offline. This attribute can only be displayed.

SUID An indicator if the effective user ID is set at run time. This attribute can be displayed or set.

SYSTEM_ARCHIVE

An indicator if the object has changed and needs to be saved. This attribute can be displayed or set.

» *SYSTEM_USE*

An indicator if the object has a special use by the system. This attribute is valid only for stream files. This attribute can only be displayed. «

SYS_SIGNED

An indicator of whether the i5/OS^(TM) digital signature is from a source that is trusted by the system. This attribute can only be displayed.

UDFS_DEFAULT_FORMAT

The default file format of stream files created in the user-defined file system. This attribute can only be displayed.

USAGE_DAYS_USED

The number of days an object has been used. This attribute can only be displayed.

USAGE_LAST_USED_TIME

The date and time that the object was last used. This attribute can only be displayed.

USAGE_RESET_TIME

The date and time that the object's days used count was reset to zero. This attribute can only be displayed.

Environment Variables

attr is affected by the following environment variables:

LANG

Provides a default value for locale categories that are not specifically set with a variable starting with LC_.

LC_TIME

Defines the output format for date and time attributes.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “setccsid - Set CCSID attribute for file” on page 125
- “touch - Change file access and modification times” on page 128
- Qp0lGetAttr() - Get attributes
- Qp0lSetAttr() - Set attributes

Examples

1. Display all of the attributes for a file.
attr script.sh
2. Display the OBJTYPE and PC_READ_ONLY attributes for a file.
attr script.sh OBJTYPE PC_READ_ONLY
3. Display the DATA_SIZE_64 attribute in a re-entrable format for a file.

```
attr -p script.sh DATA_SIZE_64
```

4. Set the PC_HIDDEN attribute for a file.

```
attr script.sh PC_HIDDEN=1
```

basename - Return non-directory portion of path name

Synopsis

```
basename string [suffix]
```

Description

You can use **basename** to delete any prefix ending with the last slash (/) character present in *string*, and a *suffix*, if specified. The resulting filename is written to standard output. The *string* is processed using the following rules:

- If *string* consists entirely of slash characters, a single slash character is written to standard output and processing ends.
- If there are any trailing slash characters in *string*, they are removed.
- If there are any slash characters remaining in *string*, the prefix of *string* up to and including the last slash character is removed.
- If a *suffix* is specified, and is not identical to the characters remaining in *string*, and is identical to a suffix of the characters remaining in *string*, the suffix is removed. Otherwise *string* is not modified. It is not an error if *suffix* is not found in *string*.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- “dirname - Return directory portion of path name” on page 97

Examples

1. Set the shell variable FOO to “trail”.

```
FOO=$(basename /usr/bin/trail)
```
2. Return the last part of the path “/usr/bin/this_test” with the “test” suffix removed.

```
basename /usr/bin/this_test test
```

cat - Concatenate and print files

Synopsis

```
cat [-bcensStuv] [-] [file ...]
```

Description

The **cat** utility reads the specified *files* sequentially, writing them to standard output. The file operands are processed in command line order. A single dash represents standard input.

By default, **cat** reads *file* as text data so the data is translated from the CCSID of the file. When the **-c** option is specified, **cat** reads the file as binary data.

Note that because of the shell language mechanism used to perform output redirection, the command **cat file1 file2 > file2** will cause the original data in file2 to be destroyed. Also, the process will go into an infinite loop.

Options

- b** Number the output lines but do not number blank lines.
- c** Do not convert the data as it is read.
- e** Number the output lines and display a dollar sign (\$) at the end of each line as well.
- n** Number the output lines, starting at 1.
- s** Squeeze multiple adjacent empty lines, causing the output to be single spaced.
- S** Squeeze multiple adjacent empty lines, causing the output to be single spaced.
- t** Display non-printing characters so they are visible like the **-v** option and display tab characters as well.
- u** Guarantee that the output is unbuffered.
- v** Display non-printing characters so they are visible. A control character prints as "**^X**" (for control). The delete character prints as "**^?**". A non-display character prints as "**M-x**" (for meta). Note that in most locales, all of the characters are displayable.

Exit Status

- 0 when successful.
- >0 when an error occurred.

Related information

- "head - Copy the first part of files" on page 104
- "tail - Display the last part of a file" on page 125
- "zcat - Expand and concatenate data" on page 131

Examples

1. Display the contents of file, "myfile".

```
cat myfile
```
2. Display the contents of three different files at the same time and save their contents into a new file.

```
cat file1 file2 file3 > all.files
```

catsplf - Concatenate and print spool files

Synopsis

```
catsplf -j qualified-job [-aen] splfname splfnum
```

```
catsplf -p pid [-aen] splfname splfnum
```

Description

The **catsplf** utility reads the specified spool file and writes it to standard output.

In the first synopsis form, **catsplf** finds the spool files associated with the job specified by *qualified-job*.

In the second synopsis form, **catsplf** finds the spool files associated with the job specified by *pid*.

Options

- a** Print all of the spool files associated with the specified job.
- e** Number the output lines starting at 1 and display a dollar sign (\$) at the end of each line.

-j *qualified-job*

Find the spool files associated with the job identified by *qualified-job*, where *qualified-job* is a string in the form *number/user/name*. The *number* is a six-digit decimal number, *user* is the user profile under which the job was started, and *name* is the name of job.

-n Number the output lines starting at 1.

-p *pid* Find the spool files associated with the job identified by *pid*, where *pid* is the decimal process ID of the job.

Operands

The *splfname* operand specifies the name of the spool file and the *splfnum* operand specifies the number of the spool file. Both operands are required to uniquely identify a spool file.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “cat - Concatenate and print files” on page 86
- “Rfile - Read or write record files” on page 122
- “zcat - Expand and concatenate data” on page 131

Examples

1. Print the spool file named QPRINT and number 1 for a job using a qualified job name.

```
catsplf -j 386687/SHELLTST/QZSHCHLD QPRINT 1
```

2. Print the spool file named QPRINT and number 1 for a job using a pid.

```
catsplf -p 942 QPRINT 1
```

3. Print all of the spool files for a job.

```
catsplf -a -j 386687/SHELLTST/QZSHCHLD
```

cd - Change working directory

Synopsis

```
cd [ directory ]
```

Description

You can use **cd** to change the working directory. **qsh** sets the **PWD** variable to the new working directory and the **OLDPWD** variable to the previous working directory.

Options

None.

Operands

For *directory*, you can specify:

- (minus)

qsh changes the working directory to the previous directory and displays the new working directory name.

/name or *../name*

qsh changes the working directory to the specified *name*.

name (does not begin with a / or ../)

If the **CDPATH** variable is set, **qsh** prepends each directory in **CDPATH** to *name* to construct a directory name. **qsh** changes to the first directory that you have permission to. **qsh** displays the new working directory name.

If the **CDPATH** variable is not set, **qsh** changes the working directory to the specified *name*.

not specified

qsh changes the working directory to the value of the **HOME** variable.

You must have permission to the specified *directory*.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “hash - Remember or report utility locations” on page 145
- “pwd - Return working directory name” on page 121

chgrp - Change file group ownership

Synopsis

```
chgrp [-R [ -H | -L | -P ]] [ -h ] group file ...
```

Description

You can use **chgrp** to set the group of *file* to the group identifier or profile specified by *group*.

To change the group identifier, you must have one of the following authorities:

- The current user has *ALLOBJ special authority.
- The current user is the owner of *file* and either one of the following:
 - The primary group of the job is *group*.
 - One of the supplemental groups of the job is *group*.

In addition, the current user must have *USE authority to the group profile specified by *group*.

By default, **chgrp** follows symbolic links and changes the group of the file pointed to by the symbolic link.

The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command’s actions are determined by the last one specified.

» The group of a file cannot be the same as the owner of the file. «

Options

- H** If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed.
- L** If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P** If the **-R** option is specified, no symbolic links are followed.
- R** If *file* is a directory, **chgrp** recursively changes the group of each file in the entire subtree connected at that point.

-h Change the owner and group of a symbolic link instead of the file pointed to by the symbolic link.

Operands

The *group* operand specifies either a group identifier number or group profile name. The *file* operand specifies a path name to an object.

Exit Status

- 0 when successful and all requested changes were made.
- >0 when an error occurred.

Related information

- “*chmod - Change file modes*”
- “*chown - Change file ownership*” on page 93

Examples

1. Change the group to group profile “*abbey*” for the file “*newgui.java*”.

```
chgrp abbey newgui.java
```
2. Change the group to group profile “*managers*” for the subdirectory “*personal.dir*” and all files and subdirectories below this directory.

```
chgrp -R managers personal.dir
```
3. Change the group to group identifier “*442*” for the file “*memo.txt*”.

```
chgrp 442 memo.txt
```

chmod - Change file modes

Synopsis

```
chmod [ -R [-H | -L | -P] ] [ -h ] mode file ...
```

Description

The **chmod** utility modifies the file mode bits of *file* as specified by the *mode* operand.

To change the mode of a file, you must have one of the following authorities:

- The current user has *ALLOBJ special authority.
- The current user is the owner of the file.

By default, **chmod** follows symbolic links and changes the mode on the file pointed to by the symbolic link. Symbolic links do not have modes so using **chmod** on a symbolic link always succeeds and has no effect.

The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command’s actions are determined by the last one specified.

Note that **chmod** changes the i5/OS^(TM) data authorities for an object. Use the CHGAUT CL command to change the i5/OS^(TM) object authorities for an object.

Options

-H If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed. Since symbolic links do not have modes **chmod** has no effect on the symbolic links.

- L If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P If the **-R** option is specified, no symbolic links are followed. Since symbolic links do not have modes **chmod** has no effect on the symbolic links.
- R If *file* designates a directory, **chmod** changes the mode of each file in the entire subtree connected at that point.
- h Do not follow symbolic links. Since symbolic links do not have modes **chmod** has no effect on the symbolic links.

Operands

A *mode* may be absolute or symbolic. An absolute mode is a three or four digit octal number constructed by or-ing the following values:

- 4000 Set-user-id on execute bit
- 2000 Set-group-id on execute bit
- 1000 Restricted deletion bit for a directory
- 0400 Allow read by owner
- 0200 Allow write by owner
- 0100 Allow execute/search by owner
- 0040 Allow read by group
- 0020 Allow write by group
- 0010 Allow execute/search by group
- 0004 Allow read by other
- 0002 Allow write by other
- 0001 Allow execute/search by other

A symbolic mode is described by the following grammar:

- *mode* ::= clause [, clause ...]
- *clause* ::= [who ...] [action ...] last_action
- *action* ::= op [perm ...]
- *last_action* ::= op [perm ...]
- *who* ::= a | u | g | o
- *op* ::= + | - | =
- *perm* ::= r | w | x | X | s | t | u | g | o

The *who* symbols specify who is granted or denied the permissions as follows:

- u** The owner permission bits.
- g** The group permission bits.
- o** The other permission bits.
- a** The owner, group, and other permission bits. It is equivalent to specifying the **ugo** symbols together.

The *op* symbols represent the operation performed, as follows:

- +** Grant the specified permission. If no value is supplied for *perm*, the "+" operation has no effect. If

no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is set. Otherwise, the mode bits represented by the specified *who* and *perm* values are set.

- Deny the specified permission. If no value is supplied for *perm*, the "-" operation has no effect. If no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is cleared. Otherwise, the mode bits represented by the specified *who* and *perm* values are cleared.
- = Clear the selected permission field and set it to the specified permission. The mode bits specified by the *who* value are cleared, or, if no *who* value is specified, the owner, group and other mode bits are cleared. Then, if no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is set. Otherwise, the mode bits represented by the specified *who* and *perm* values are set.

The *perm* symbols represent the portions of the mode bits as follows:

- r** The read bits.
- w** The write bits.
- x** The execute/search bits.
- X** The execute/search bits if the file is a directory or if any of the execute/search bits are set in the original (unmodified) mode. Operations with this symbol are only meaningful in conjunction with the *op* symbol "+", and are ignored in all other cases.
- s** The set-user-id on execute bit when the owner permission bits are set or the set-group-id on execute bit when the group permission bits are set.
- t** The restricted deletion bit when the object is a directory. It can be used when the *who* symbol is **a** or there is no *who* symbol. It is ignored if the file is not a directory or the *who* symbol is **u**, **g**, or **o**.

Each *clause* specifies one or more operations to be performed on the mode bits, and each operation is applied to the mode bits in the order specified.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- "chgrp - Change file group ownership" on page 89
- "chown - Change file ownership" on page 93

Examples

1. Grant read and write permission to owner and read permission to group and other using an absolute mode.
`chmod 644 myfile`
2. Deny write permission to group and other.
`chmod go-w myfile`
3. Clear all permissions that are currently set and grant read and write permissions to owner, group, and other.
`chmod =rw myfile`
4. Grant search permission on a directory to owner, group, and other if search permission is set for one them.
`chmod +X mydir`

5. Grant read, write, and execute permission to owner and read and execute permission to group and other using an absolute mode.

```
chmod 755 myfile
```

6. Clear all permissions for group and other.

```
chmod go= myfile
```

7. Set the group permissions equal to the owner permission, but deny write permission to the group.

```
chmod g=u-w myfile
```

8. Set the set-user-id on execute bit and grant read, write, and execute permission to the owner and execute permission for other using an absolute mode.

```
chmod 4701 myfile
```

chown - Change file ownership

Synopsis

```
chown [-R [ -H | -L | -P ]] [ -h ] owner[:group] file ...
```

Description

You can use **chown** to set the owner of *file* to the user identifier or profile specified by *owner*. Optionally, **chown** can also set the group of the *file* to the group identifier or profile specified by *group*.

To change the owner of a file, you must have one of the following authorities:

- The current user has *ALLOBJ special authority.
- The current user is the owner of the file or directory.

To change the group of a file, you must have one of the following authorities:

- The current user has *ALLOBJ special authority.
- The current user is the owner of *file* and either one of the following:
 - The primary group of the job is *group*.
 - One of the supplemental groups of the job is *group*.

In addition, the current user must have *USE authority to the new user profile or group profile.

By default, **chown** follows symbolic links and changes the owner and group of the file pointed to by the symbolic link.

» The group of a file cannot be the same as the owner of the file. «

Options

- H** If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed.
- L** If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P** If the **-R** option is specified, no symbolic links are followed.
- R** If *file* designates a directory, **chown** recursively changes the owner and group of each file in the entire subtree connected at that point.
- h** Change the owner and group of a symbolic link instead of the file pointed to by the symbolic link.

Operands

The *owner* operand specifies either a user identifier number or a user profile name. The *group* operand specifies either a group identifier number or a group profile name. The *file* operand specifies a path name to an object.

Exit Status

- 0 when successful and all requested changes were made.
- >0 when an error occurred.

Related information

- “chgrp - Change file group ownership” on page 89
- “chmod - Change file modes” on page 90

Examples

1. Change the owner to user profile "sam" for the file "personal.file".
`chown sam personal.file`
2. Recursively change the owner to user profile "larry" for the sub-directory "moe.dir" and all files and sub-directories below this directory.
`chown -R larry moe.dir`
3. Change the owner to user identifier "500" for the file "your.file".
`chown 500 your.file`
4. Change the owner to user profile "sam" and the group to group profile "abbey" for the file "memo.txt".
`chown sam:abbey memo.txt`

compress - Compress data

Synopsis

`compress [-cfv] [-b bits] [file ...]`

Description

The **compress** utility reduces the size of the *files* using adaptive Lempel-Ziv coding. Each *file* is renamed to the same name plus the extension ".Z". As many of the modification time, access time, file flags, file mode, user ID, and group ID as allowed by permissions are retained in the new file. If compression would not reduce the size of a file, the *file* is ignored.

If renaming *file* would cause files to be overwritten and the standard input device is a terminal, the user is prompted (on standard error) for confirmation. If prompting is not possible or confirmation is not received, the files are not overwritten.

Options

- b *bits* Specify the bits code limit (see below for details).
- c Compressed output is written to the standard output. No files are modified.
- f Force compression of *file*, even if it is not actually reduced in size. Additionally, files are overwritten without prompting for confirmation.
- v Print the percentage of reduction for each *file*.

Operands

Each *file* is a pathname of a file to compress. If no *files* are specified, the standard input is compressed to the standard output. If either the input or output files are not regular files, the checks for reduction in size and file overwriting are not performed, the input file is not removed, and the attributes of the input file are not retained.

Extended Description

The **compress** utility uses a modified Lempel-Ziv algorithm. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (the default is 16). Bits must be between 9 and 16.

After the bits limit is reached, **compress** periodically checks the compression ratio. If it is increasing, **compress** continues to use the existing code dictionary. However, if the compression ratio decreases, **compress** discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

The amount of compression obtained depends on the size of the input, the number of bits per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50-60%.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- "pax - Portable archive interchange" on page 112
- "uncompress - Expand compressed data" on page 130
- "zcat - Expand and concatenate data" on page 131

cp - Copy files

Synopsis

```
cp [-r | -R [-H | -L | -P] ] [-fhipt] source_file target_file
```

```
cp [-r | -R [-H | -L | -P] ] [-fhipt] source_file ... target_directory
```

Description

In the first synopsis form, the **cp** utility copies the contents of the *source_file* to the *target_file*.

In the second synopsis form, the **cp** utility copies the contents of each named *source_file* to a file in the destination *target_directory*. The names of the files themselves are not changed. The *target_directory* must exist unless there is only one named *source_file* which is a directory and the **-R** flag is specified.

If **cp** detects an attempt to copy a file to itself, the copy will fail.

If *target_file* does not exist, the mode of the *source_file* is used, as modified by the file creation mask, when creating *target_file*. The S_ISUID and S_ISGID file permission bits are never set when creating a new file.

If *target_file* already exists and the **-t** option is not specified, its contents are overwritten as binary data and the CCSID attribute is changed to match the CCSID attribute of *source_file*. The file permission bits, owner, and group of *target_file* are unchanged. You can force the data to be copied as text data by using the **-t** option. You can force the file permission bits, owner, and group to be copied using the **-p** option.

Note that when copying to members in the QSYS.LIB file system, many attributes of *source_file* cannot be preserved because they are associated with the file object and not the member.

Symbolic links are always followed unless the **-h** option is specified or the **-R** option is specified with the **-H** or the **-L** options. The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command's actions are determined by the last one specified.

Options

- H** If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed and the symbolic link is copied instead of the file pointed to by the symbolic link.
- L** If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P** If the **-R** option is specified, no symbolic links are followed. A symbolic link encountered in the tree traversal is copied instead of the file pointed to by the symbolic link.
- R** If *source_file* designates a directory, **cp** copies the directory and the entire subtree connected at that point. This option causes **cp** to create special files rather than copying them as normal files. Created directories have the same mode as the corresponding source directory, unmodified by the file creation mask.
- f** Remove *target_file* if it cannot be opened for write operations. A new file is created before the data is copied.
- h** Copy symbolic links instead of the file pointed to by the symbolic link.
- i** Write a prompt to standard error before copying a file that would overwrite an existing file. If the response from the standard input begins with the first character for the YES response in the current locale, the file copy is attempted.
- p** Preserve in the copy as many of the modification time, access time, file permission bits, owner, and group as allowed by permissions.

If the owner and group cannot be preserved, no error message is displayed and the exit value is not altered.

The **S_ISUID** and **S_ISGID** file permission bits are only copied when both the owner and group of the file are successfully copied.

Note: This option has no effect when copying to the QSYS.LIB file system.

- r** Same as **-R** except this option copies special files in the same manner as regular files. The **-R** flag is preferred to the **-r** flag.
- t** When the target file exists, treat the data in *source_file* as text data and translate the data to the CCSID associated with *target_file* as it is copied. The CCSID attribute of *target_file* is not changed.

Exit Status

- 0 on success
- >0 if an error occurred.

Related information

- “ln - Link files” on page 104
- “ls - List directory contents” on page 105
- “mv - Move files” on page 110
- “rm - Remove directory entries” on page 123

- “rmdir - Remove directories” on page 124
- “umask - Get or set the file mode creation mask” on page 129

Examples

1. Copy the file, “file1”, into the subdirectory, “data.dir”.

```
cp file1 data.dir
```
2. Copy all the files with the .java extension from the “code” subdirectory into the subdirectory, “code/old_code.dir” and prompt the user for overwrite verification only if the file already exists in the subdirectory, “code/old_code.dir”.

```
cp -i code/*.java code/old_code.dir
```

dirname - Return directory portion of path name

Synopsis

dirname *string*

Description

You can use **dirname** to delete the filename portion, beginning with the last slash character (/) to the end of *string*, and write the result to standard output. The *string* is processed using the following rules:

- If *string* consists entirely of slash characters, a single slash character is written to standard output and processing ends.
- If there are any trailing slash characters in *string*, they are removed.
- If there are no slash characters remaining in *string*, a period character is written to standard output and processing ends.
- If there are trailing non-slash characters in *string*, they are removed.
- If there are any trailing slash characters in *string*, they are removed.
- If the remaining string is empty, *string* is set to a single slash character.

Operands

The *string* operand is the pathname of which **dirname** will return the directory portion of.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- “basename - Return non-directory portion of path name” on page 86

Examples

1. Set the shell variable FOO to “/usr/bin”.

```
FOO=$(dirname /usr/bin/trail)
```

file - Determine file type

Synopsis

```
file [-m MagicFile] [-f ListFile] [ file ... ]
```

```
file [-c] [-m MagicFile]
```

Description

In the first synopsis form, the **file** utility determines the type of object for the specified *file*. The **file** utility will make a best guess determination of the type. The file type is then written to standard output. If the pathname is determined to be a regular file, **file** examines the first 1024 bytes to determine the type. By default, the **file** utility uses the `/etc/magic` file to help identify files that have defined patterns at specified byte offsets within the object.

In the second synopsis form, the **file** utility checks the specified *MagicFile* for format errors.

Options

-c Checks a specified magic file for format errors.

-f *ListFile*

Specifies a file containing a list of file names to be tested. This *ListFile* must have only one file per line and not contain leading or trailing spaces.

-m *MagicFile*

Specifies the name of the magic file to use. The default magic file is `/etc/magic`.

Operands

Each *file* is a pathname of a file to be tested.

Exit Status

- 0 when successful
- >0 when an error occurred

Related information

- “find - Find files”

find - Find files

Synopsis

```
find [-H | -L | -P] [-Xdx] [-f file] file ... [expression]
```

Description

The **find** utility recursively descends the directory tree for each *file* listed, evaluating an *expression* (composed of the “primaries” and “operands” listed below) in terms of each file in the tree.

Options

- H** Symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed. The file information and file type returned for each symbolic link specified on the command line is for the file referenced by the link. If the referenced file does not exist, the file information and type will be for the link itself.
- L** Both symbolic links on the command line and symbolic links encountered in the tree traversal are followed. The file information and file type returned for each symbolic link is for the file referenced by the link. If the referenced file does not exist, the file information and type will be for the link itself.
- P** No symbolic links are followed. The file information and file type returned for each symbolic link are for the link itself.
- X** A modification to permit **find** to be safely used in conjunction with **xargs**. If a file name contains any of the delimiting characters used by **xargs**, a diagnostic message is displayed on standard error, and the file is skipped. The delimiting characters include single (') and double (") quotation marks, backslash (\), space, tab and newline characters.

- d **find** performs a depth-first traversal. The directories are visited in post-order and all entries in a directory will be acted on before the directory itself. By default, **find** visits directories in pre-order, or before their contents. Note, the default is not a breadth-first traversal.
- f Specify a file hierarchy for **find** to traverse. File hierarchies may also be specified as the operands immediately following the options.
- x Prevent **find** from descending into directories that have a device number different than that of the file from which the descent began.

Primaries

-atime *n*

True if the difference between the file last access time and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-ctime *n*

True if the difference between the time of last change of file status information and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-exec *utility* [*argument ...*] ;

True if the program named *utility* returns a zero value as its exit status. Optional arguments may be passed to the utility. The expression must be terminated by a semicolon (;). If the string "{}" appears anywhere in the utility name or the arguments it is replaced by the path name of the current file. The utility is run from the directory from which **find** was run. Since the semicolon is also a special character for the shell, you may need to escape the semicolon so it is passed as an argument to **find**.

-group *gname*

True if the file belongs to the group *gname*. If *gname* is numeric and there is no such group name, then *gname* is treated as a group identifier.

-inum *n*

True if the file has inode number *n*.

-links *n*

True if the file has *n* links.

-ls This primary always evaluates to true. The following information for the current file is written to standard output:

- inode number
- size in kilobytes
- file permissions
- number of hard links
- owner
- group
- size in bytes
- last modification time
- path name

If the file is a block or character special file, the major and minor numbers will be displayed instead of the size in bytes. If the file is a symbolic link, the path name of the linked-to file will be displayed preceded by `->'.

-mtime *n*

True if the difference between the file last modification time and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-ok *utility* [*argument...*] ;

The **-ok** primary is identical to the **-exec** primary with the exception that **find** requests user

affirmation for running the *utility* by printing a message to standard error and reading a response. If the response is other than the first character of the YES response in the current locale, the *utility* is not run and the value of the ok expression is false.

-name *pattern*

True if the last component of the path name being examined matches *pattern*. Special shell pattern matching characters ([,], *, and ?) may be used as part of *pattern*. These characters may be matched explicitly by escaping them with a backslash (\).

-newer *file*

True if the current file has a more recent last modification time than *file*.

-nouser

True if the file belongs to an unknown user.

-nogroup

True if the file belongs to an unknown group.

-path *pattern*

True if the path name being examined matches *pattern*. Special shell pattern matching characters ([,], *, and ?) may be used as part of *pattern*. These characters may be matched explicitly by escaping them with a backslash (\). Slashes (/) are treated as normal characters and do not need to be matched explicitly.

-perm [-]*mode*

The *mode* may be either symbolic or an octal number in the formats supported by the “chmod - Change file modes” on page 90 command. If the *mode* is symbolic, a starting value of zero is assumed and the mode sets or clears permissions without regard to the process file mode creation mask. If the *mode* is octal, only bits 00777 (S_IRWXU | S_IRWXG | S_IRWXO) of the file’s mode bits participate in the comparison. If the *mode* is preceded by a dash (-), this primary evaluates to true if at least all of the bits in the mode are set in the file’s mode bits. If the *mode* is not preceded by a dash, this primary evaluates to true if the bits in the *mode* exactly match the file’s mode bits. Note, the first character of a symbolic mode may not be a dash (-).

-print This primary always evaluates to true. It prints the path name of the current file to standard output. The expression is appended to the user specified expression if neither **-exec**, **-ls** nor **-ok** is specified.

-prune

This primary always evaluates to true. It causes **find** to not descend into the current file. Note, the **-prune** primary has no effect if the **-d** option was specified.

-size *n*[*c*]

True if the file’s size, rounded up, in 512-byte blocks is *n*. If *n* is followed by *c*, then the primary is true if the file’s size is *n* bytes.

-type *t*

True if the file is of the specified type. Possible file types are as follows:

- b for block special
- c for character special
- d for directory
- f for regular file
- l for symbolic link
- p for FIFO
- s for socket

-user *uname*

True if the file belongs to the user *uname*. If *uname* is numeric and there is no such user name, then *uname* is treated as a user identifier.

All primaries which take a numeric argument allow the number to be preceded by a plus sign (+) or a minus sign (-). A preceding plus sign means "more than n", a preceding minus sign means "less than n" and neither means "exactly n".

Operators

The primaries may be combined using the following operators. The operators are listed in order of decreasing precedence.

(expression)

This evaluates to true if the parenthesized expression evaluates to true.

!expression

This is the unary NOT operator. It evaluates to true if the expression is false.

expression -and expression

The -and operator is the logical AND operator. As it is implied by the juxtaposition of two expressions it does not need to be specified. The expression evaluates to true if both expressions are true. The second expression is not evaluated if the first expression is false.

expression -or expression

The -or operator is the logical OR operator. The expression evaluates to true if either the first or the second expression is true. The second expression is not evaluated if the first expression is true.

All operands and primaries must be separate arguments to the **find** utility. Primaries which themselves take arguments expect each argument to be a separate argument to **find**. **Notes**

1. The special characters used by **find** are also special characters to many shell programs. In particular, the characters *, [,], ?, (,), !, and ; may need to be escaped from the shell.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- "chmod - Change file modes" on page 90
- "file - Determine file type" on page 97
- "xargs - Construct argument lists and invoke utility" on page 62

Examples

1. Find all *.class files starting at the directory "/project/java/class".

```
find /project/java/class -name '*.class'
```
2. Find all the *.java files that have the "import java.awt;" string in them starting at the directory, "/project/java/code".

```
find /project/java/code -name '*.java' -exec grep 'import java.awt;' {} \;
```
3. Find all the *.class files starting at the directory "/project/java/class" and remove the files.

```
find /project/java/class -name '*.class' -exec rm {} \;
```
4. Find all the files that belong to the user "abbey" starting at the directory, "/project".

```
find /project -user abbey
```

gencat - Generate a formatted message catalog

Synopsis

```
gencat [-C ccsid] [-m mode] [-t text] catfile msgfile ...
```

Description

The **gencat** utility generates a formatted message catalog *catfile* from the message text source file *msgfile*. You can specify up to 300 message text source files. Message text source files are processed in the sequence specified. Each successive source file modifies the catalog. If a message number in the source file already exists in the message catalog, the new message text defined in the source file replaces the old message text in the message catalog file. If a message number in the source file does not already exist in the message catalog, the message information is added to the message catalog.

Options

-C *ccsid*

Create the message catalog and store the message text in the specified *ccsid*.

-m *mode*

Set the file permission bits of the message catalog to the specified *mode*. The mode argument can be in any of the formats supported by the “*chmod* - Change file modes” on page 90 command. If a symbolic mode is specified, the operation characters + and - are interpreted relative to an initial mode of “a=rw”.

-t *text* Assign the specified *text* to the message catalog object. Assigning text to objects is dependent on the support provided by the file system or object type used for the message catalog.

Operands

The *catfile* operand specifies the path to the message catalog to be changed or created. If the **-m** option is not specified, the message catalog is created using a default mode that allows read and write permission for the owner, group, and others (0666) as modified by the current file creation mask.

Each *msgfile* specifies the path to an input message text source file. There is a limit of 300 message text source files.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “*dspmsg* - Display message from message catalog” on page 131

Examples

1. Create a message catalog using one message text source file.

```
gencat product.cat msg.src
```
2. Create a message catalog using multiple message text source files.

```
gencat product.cat msg1.src msg2.src msg3.src
```
3. Create a message catalog and set the mode and *ccsid*.

```
gencat -C 37 -m a-w product.cat msg.src
```

getconf - Get configuration values

Synopsis

```
getconf [ name [ pathname ] ]
```

Description

The **getconf** utility displays the POSIX configuration variables. If you specify *name*, **getconf** displays the value of the configuration variable on standard output. When the configuration variable depends on a path name you must specify *pathname*.

When no arguments are specified, **getconf** displays a list of all the configuration variables and their values. For those configuration variables that depend on a path name, **getconf** uses */*.

Options

None.

Operands

If specified, *name* is one of these values:

CCSID

Represents the default coded character set identifier (CCSID) used internally for integrated file system path names.

CHOWN_RESTRICTED

Restrict the use of **chown** on the object represented by *pathname* to a job with appropriate privileges.

CLK_TCK

The number of clock ticks in a second.

LINK_MAX

Maximum number of links the object represented by *pathname* can have.

NAME_MAX

Maximum number of bytes in a file name (the last component of the path name).

NGROUPS_MAX

Maximum number of supplementary group IDs that can be associated with a job.

NO_TRUNC

Generate an error if a file name is longer than NAME_MAX.

OPEN_MAX

Maximum number of files a single job can have open at one time.

PAGE_SIZE

Represents the system hardware page size.

PAGESIZE

Represents the system hardware page size.

PATH_MAX

Maximum number of bytes in a complete path name.

PIPE_BUF

Maximum number of bytes that can be written atomically to a pipe.

STREAM_MAX

Maximum number of streams that a job can have open at one time.

THREAD_SAFE

The object represented by *pathname* resides in a thread-safe file system.

Exit Status

- 0 when successful.
- >0 when successful.

Examples

1. Determine if the directory `/home` is in a thread-safe file system:
`getconf THREAD_SAFE /home`
2. Display the maximum number of bytes in a file name:
`getconf NAME_MAX`
3. Display all of the configuration variables:
`getconf`

head - Copy the first part of files

Synopsis

`head [-n count] [file ...]`

Description

The **head** utility displays the first *count* lines of each of the specified files, or of standard input if no files are specified. If **-n** is not specified, then the first 10 lines of the file are displayed.

If more than one *file* is specified, each *file* is preceded by a header consisting of the string "`==> XXX <==`" where XXX is the name of the file.

Options

-n Display *count* number of lines.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- "cat - Concatenate and print files" on page 86
- "tail - Display the last part of a file" on page 125

Examples

1. To display the first 20 lines in the file "myfile".
`head -n 20 myfile`

In - Link files

Synopsis

`In [-fs] source_file [target_file]`

`In [-fs] source_file ... [target_dir]`

Description

The **In** utility creates a new directory entry (linked file) which has the same modes as the original file. It is useful for maintaining multiple copies of a file in many places at once without using up storage for the copies. Instead, a link "points to" the original copy. There are two types of links: hard links and symbolic links. How a link "points to" a file is one of the differences between a hard or symbolic link.

By default **In** makes hard links. A hard link to a file is indistinguishable from the original directory entry; any changes to a file are effective independent of the name used to reference the file. Hard links may not normally refer to directories and may not span file systems.

A symbolic link contains the name of the file to which it is linked. Symbolic links may span file systems and may refer to directories.

Given one or two arguments, **ln** creates a link to an existing file *source_file*. If *target_file* is given, the link has that name. *Target_file* may also be a directory in which to place the link. Otherwise it is placed in the current directory. If only the directory is specified, the link will be made to the last component of *source_file*.

Given more than two arguments, **ln** makes links in *target_dir* to all the named source files. The links made will have the same name as the files being linked to.

Options

- f Unlink any already existing file, permitting the link to occur.
- s Create a symbolic link.

Exit Status

- 0 when success
- >0 when an error occurs

Related information

- “cp - Copy files” on page 95
- “ls - List directory contents”
- “mv - Move files” on page 110
- “rm - Remove directory entries” on page 123
- “rmdir - Remove directories” on page 124

Examples

1. Create a symbolic link from the file, `"/usr/bin/perl5"` to the file `"/usr/bin/perl"`.

```
ln -s /usr/bin/perl5 /usr/bin/perl
```
2. Create a new link from the file `"/usr/bin/qsh"` to the file `"/bin/qsh"` and unlink the file `"/bin/qsh"` if it exists.

```
ln -f /usr/bin/qsh /bin/qsh
```

ls - List directory contents

Synopsis

```
ls [-ACFLRSTacdfileoqrstu1] [file ...]
```

Description

For each operand that names a *file* of a type other than directory, **ls** displays its name as well as any requested, associated information. For each operand that names a file of type directory, **ls** displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

Options

- A List all entries except for `."` and `.."`.
- C Force multi-column output; this is the default when output is to a terminal.

- F Display a slash (/) immediately after each path name that is a directory, an asterisk (*) after each that is executable, and an at sign (@) after each symbolic link.
- L If argument is a symbolic link, list the file or directory the link references rather than the link itself.
- R Recursively list subdirectories.
- S Display the CCSID attribute for the file.
- T Display complete time information for the file, including month, day, hour, minute, second, and year when the -l option is also specified.
- a Include directory entries whose names begin with a dot (.).
- c Use time when file status was last changed for sorting or printing.
- d Directories are listed as plain files (not searched recursively) and symbolic links in the argument list are not indirected through.
- f Output is not sorted.
- i For each file, print the file's file serial number (inode number).
- l (Lowercase letter `ell.')
- l List in long format. See Extended Description below for details. If the output is to a terminal, a total sum for all the file sizes is output on a line before the long listing.
- o Include the file flags in a long (-l) output.
- q Force printing of non-graphic characters in file names as the question mark (?) character. This is the default when output is to a terminal.
- r Reverse the order of the sort to get reverse lexicographical order or the oldest entries first.
- s Display the number of bytes actually allocated for each file, in units of 1024 bytes, where partial units are rounded up to the next integer value.
- t Sort by time modified (most recently modified first) before sorting the operands by lexicographical order.
- u Use time of last access, instead of last modification of the file for sorting (-t) or printing (-l).
- 1 (The numeric digit one) Force output to be one entry per line. This is the default when output is not to a terminal.

The -l, -C, and -l options all override each other. The last one specified determines the format used.

The -c, and -u options override each other. The last one specified determines the file time used.

By default, ls lists one entry per line to standard output; the exceptions are to terminals or when the -C option is specified.

File information is displayed with one or more blanks separating the information associated with the -i, -s, -l, and -S options.

Extended Description

If the -l option is specified, the following long format information is displayed for each file:

- file mode,
- number of links,
- owner name,
- group name,
- number of bytes in the file,

- time the file was last modified, and
- the path name.

If the file was modified within six months of the current date, the time is displayed as the abbreviated month, day-of-month, hour, and minute. Otherwise the time is displayed as the abbreviated month, day-of-month, and four-digit year.

In addition, for each directory whose contents are displayed, the total number of bytes used by the files in the directory is displayed on a line by itself immediately before the information for the files in the directory.

If the owner or group names are not a known user or group name the numeric identifiers are displayed.

If the file is a character special or block special file, the major and minor device numbers for the file are displayed in the size field. If the file is a symbolic link the pathname of the linked-to file is preceded by "->".

The file mode consists of the entry type, owner permissions, group permissions, and other permissions. The entry type character describes the type of file, as follows:

- b for a block special file.
- c for a character special file.
- d for a directory.
- l for a symbolic link.
- p for a pipe.
- s for a socket.
- - for a regular file.

The owner permissions, group permissions, and other permissions are each three characters. Each field has three character positions:

- For the first position, if the value is r, the file is readable. If the value is -, it is not readable.
- For the second position, if the value is w, the file is writable. If the value is -, it is not writable.
- For the third position,
 - If the value is S for the owner permissions, the set-user-ID mode is set. If the value is S for the group permissions, the set-group-ID mode is set.
 - If the value is s for the owner permissions, the file is executable and the set-user-ID mode is set. If the value is s for the group permissions, the file is executable and the set-group-ID mode is set.
 - If the value is x, the file is executable or the directory is searchable.
 - If the value is -, the object is not executable or searchable.

Environment Variables

ls is affected by the following environment variables:

COLUMNS

If this variable contains a string representing a decimal integer, it is used as the column position width for displaying multiple-text-column output. The **ls** utility calculates how many path name text columns to display based on the width provided. See the **-C** option.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- “chgrp - Change file group ownership” on page 89
- “chmod - Change file modes” on page 90
- “chown - Change file ownership” on page 93
- “cp - Copy files” on page 95
- “ln - Link files” on page 104
- “mkdir - Make directories”
- “mv - Move files” on page 110
- “rm - Remove directory entries” on page 123
- “rmdir - Remove directories” on page 124

Examples

1. Display the list of files in the current directory using the long format.

```
ls -l
```

2. Display all date and time details for the file “myfile”.

```
ls -lT myfile
-rwxrwxrwx 1 abbey 0      592 Sep 12 22:47:01 1998 myfile
```

mkdir - Make directories

Synopsis

```
mkdir [-p] [-m mode] directory ...
```

Description

The **mkdir** utility creates the directories named as operands, in the order specified, using mode `rwXrwxrwx` (0777) as modified by the current file creation mask.

The user must have write permission in the parent directory.

Options

- m** Set the file permission bits of the final created directory to the specified *mode*. The mode argument can be in any of the formats supported by the “chmod - Change file modes” on page 90 command. If a symbolic mode is specified, the operation characters + and - are interpreted relative to an initial mode of “a=rwx”.
- p** Create intermediate directories as required. If this option is not specified, the full path prefix of each operand must already exist. Intermediate directories are created with permission bits of `rwXrwxrwx` (0777) as modified by the current file creation mask, plus write and search permission for the owner.

Exit Status

- 0 if successful
- >0 if an error occurred.

Related information

- “chmod - Change file modes” on page 90
- “rmdir - Remove directories” on page 124
- “umask - Get or set the file mode creation mask” on page 129

Examples

1. Create the directories "new", "java", "test", "dir", "4" and "bob" and set the mode to read, write and execute for owner.

```
mkdir -p -m 700 /new/java/test/dir/4/bob
```

mkfifo - Make FIFO special files

Synopsis

```
mkfifo [-p] [-m mode] file ...
```

Description

The **mkfifo** utility creates the FIFO special files named as operands, in the order specified, using a default mode that allows read and write permission for the owner, group, and others (0666) as modified by the current file creation mask.

The user must have write permission in the parent directory.

Options

-m *mode*

Set the file permission bits of the FIFO special file to the specified *mode*. The mode argument can be in any of the formats supported by the "chmod - Change file modes" on page 90 command. If a symbolic mode is specified, the operation characters + and - are interpreted relative to an initial mode of "a=rw".

-p

Create intermediate directories as required. If this option is not specified, the full path prefix of each *file* must already exist. Intermediate directories are created with a default mode that allows read, write, and search permission for the owner, group, and others (0777) as modified by the current file creation mask.

Operands

Each *file* is the path name of FIFO special file.

Exit Status

- 0 if successful
- >0 if an error occurred.

Related information

- "chmod - Change file modes" on page 90
- "mkdir - Make directories" on page 108
- "umask - Get or set the file mode creation mask" on page 129

Examples

1. Create the FIFO special files "fifo1" and "fifo1":

```
mkfifo fifo1 fifo2
```

2. Create the the FIFO special file "fifo1" and set the permissions to read, write and execute for the owner:

```
mkfifo -m 700 myfifo
```

3. Create the the FIFO special file "/dir1/dir2/fifo1" and each directory in the path that does not exist:

```
mkfifo -p /dir1/dir2/fifo1
```

mv - Move files

Synopsis

```
mv [-f | -i] source_file target_file
```

```
mv [-f | -i] source_file ... target_dir
```

Description

In its first form, the **mv** utility renames the file named by the *source_file* operand to the destination path named by the *target_file* operand. This form is assumed when the last operand does not name an already existing directory.

In its second form, **mv** moves each file named by a *source_file* operand to a destination file in the existing directory named by the *target_dir* operand. The destination path for each *source_file* operand is the path name produced by the concatenation of *target_dir*, a slash, and the final path name component from *source_file*.

It is an error for either the *source_file* operand or the destination path to specify a directory except when both are directories.

If the destination path does not have a mode which permits writing, **mv** prompts the user for confirmation as specified for the **-i** option.

Options

- f** Do not prompt for confirmation before overwriting the destination path. The **-i** option is ignored if the **-f** option is specified.
- i** Write a prompt to standard error before moving a file that would overwrite an existing file. If the response from the standard input begins with the first character for the YES response in the current locale, the move is attempted.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- “cp - Copy files” on page 95
- “ln - Link files” on page 104
- “ls - List directory contents” on page 105
- “rm - Remove directory entries” on page 123

Examples

1. Move the file “perl5” into the directory “/usr/bin” and prompt the user to overwrite if the file exists.

```
mv -i perl5 /usr/bin
```

od - Dump files in various formats

Synopsis

```
od [-A address_base] [-j skip] [-N count] [-t type_string] [-Cbcdoxv] [file...]
```

Description

The **od** utility writes the contents of the specified *files* to standard output in a user-specified format. If the *file* parameter is not given, the **od** command reads standard input. The format is specified by the **-t** flag. If no format type is specified, **-t oS** is the default.

Options

-A *address_base*

Specifies the format for the output offset base. The *address_base* can be one of these values:

- **d** for decimal,
- **o** for octal,
- **x** for hexadecimal, or
- **n** for none.

In the case of **n**, the offset base is not displayed. If **-A** is not specified, **-A o** is the default.

-b Output bytes in octal. It is equivalent to **-t 01**.

-C Display the CCSID of the file to standard output before the rest of the output is written.

-c Output bytes as characters. It is equivalent to **-t c**.

-d Output bytes in unsigned decimal. It is equivalent to **-t u2**.

-j *skip* Specifies the number of bytes to skip before beginning to display output. If more than one file is specified, the number of bytes will be used on the concatenated input of all files specified. An error will occur if this number is larger than the size of the concatenated inputs. This value can be specified in hexadecimal (preceded by 0x or 0X), octal (preceded by 0), or decimal (default).

-N *count*

Specifies the number of bytes to be written. By default, the whole file will be written. This value can be specified in hexadecimal (preceded by 0x or 0X), octal (preceded by 0), or decimal (default).

-o Output bytes in octal. It is equivalent to **-t o2**.

-s Output bytes in signed decimal. It is equivalent to **-t d2**.

-t *type_string*

Specifies one or more output types. The type specified must be a string containing all of the formatting types that you want. The *type_string* can contain these values:

- **a** for character,
- **c** for character,
- **d** for signed decimal,
- **f** for floating point,
- **o** for octal,
- **u** for unsigned decimal, or
- **x** for hexadecimal.

The type specifications of **a** and **c** may give unexpected results since they depend on the CCSID on the data. The **a** type specifier displays non-printable characters as named characters. The **c** type specifier displays non-printable characters as three digit octal numbers.

The type specifications of **d**, **o**, **u** and **x** can also be followed by **1**, **2**, **4**, **C**, **S**, **I** or **L**. These specify the number of bytes to be transformed by each instance of the output type. The values **C**, **S**, **I** and **L** correspond to char, short, int and long.

The type specification of **f** can be followed by **4**, **8**, **F**, **D** or **L**. These specify the number of bytes to be transformed by each instance of the output type. The values **F**, **D** and **L** correspond to float, double, and long double. If **-t** is not specified, the default is **-t oS**.

- v Write all input data. Without this option, repeated output lines will not be written. When repeats occur, only an asterisk (*) will be written.
- x Output bytes in hexadecimal. It is equivalent to **-t x2**.

Operands

Each *file* is a path name of an object to be written to standard output. If no *file* operands are specified, standard input will be used.

Exit Status

- 0 when successful
- >0 when an error occurred.

Related information

- “cat - Concatenate and print files” on page 86
- “pr - Print files” on page 119

Examples

1. Dump a file in hexadecimal format.
od -tx output.txt
2. Dump the first 50 bytes of a file.
od -N50 output.txt
3. Skip the first 100 bytes and then dump the rest of a file.
od -j100 output.txt
4. Dump a file in both hexadecimal and character format.
od -tx1 -tc output.txt

pax - Portable archive interchange

Synopsis

```
pax [-cdnv] [-E limit] [-f archive] [-s replstr ...] [-U user ...] [-G group ...] [-T [from_date][,to_date] ...] [pattern ...]
```

```
pax -r [-cdiknuvDYZ] [-C ccsid ] [-E limit] [-f archive] [-o options ...] [-p string ...] [-s replstr ...] [-U user ...] [-G group ...] [-T [from_date][,to_date] ...] [pattern ...]
```

```
pax -w [-dituvHLPX] [-b blocksize] [[-a] [-f archive]] [-x format] [-B bytes] [-s replstr ...] [-o options ...] [-U user ...] [-G group ...] [-T [from_date][,to_date][/[c][m]] ...] [file ...]
```

```
pax -r -w [-diklntuvDHLXPYZ] [-p string ...] [-s replstr ...] [-U user ...] [-G group ...] [-T [from_date][,to_date][/[c][m]] ...] [file ...] directory
```

Description

The **pax** utility reads, writes, and lists the members of an archive file, and copies directory hierarchies. **pax** operation is independent of the specific archive format, and supports a wide variety of different archive formats. A list of supported archive formats can be found under the description of the **-x** option.

The presence of the **-r** and the **-w** options specifies which of the following functional modes **pax** will operate under: list, read, write, and copy.

<none>List

pax writes a table of contents of the members of the archive file read from whose path names match the specified patterns. The table of contents contains one file name per line and is written using single line buffering.

-r Read

pax extracts the members of the archive file read from the with path names matching the specified *patterns*. The archive format and blocking is automatically determined on input. When an extracted file is a directory, the entire file hierarchy rooted at that directory is extracted. All extracted files are created relative to the current file hierarchy. The setting of ownership, access and modification times, and file mode of the extracted files are discussed in more detail under the **-p** option.

-w Write

pax writes an archive containing the *file* operands to standard output using the specified archive *format*. When no *file* operands are specified, a list of files to copy with one per line is read from standard input. When a *file* operand is also a directory, the entire file hierarchy rooted at that directory will be included.

-r -w Copy

pax copies the *file* operands to the destination *directory*. When no *file* operands are specified, a list of files to copy with one per line is read from standard input. When a *file* operand is also a directory the entire file hierarchy rooted at that directory will be included. The effect of the copy is as if the copied files were written to an archive file and then subsequently extracted, except that there may be hard links between the original and the copied files (see the **-l** option below).

Warning:

The destination *directory* must not be one of the *file* operands or a member of a file hierarchy rooted at one of the file operands. The result of a copy under these conditions is unpredictable.

Note:

Archive files must be in CCSID 819 for portability with other platforms.

While processing a damaged archive during a read or list operation, **pax** will attempt to recover from media defects and will search through the archive to locate and process the largest number of archive members possible (see the **-E** option for more details on error handling).

Options

- r** Read an archive file from standard input and extract the specified *files*. If any intermediate directories are needed in order to extract an archive member, these directories will be created as if **mkdir** was called with the bitwise inclusive OR of **S_IRWXU**, **S_IRWXG**, and **S_IRWXO** as the mode argument. When the selected archive *format* supports the specification of linked files and these files cannot be linked while the archive is being extracted, **pax** will write a diagnostic message to standard error and exit with a nonzero exit status at the completion of operation.
- w** Write files to the standard output in the specified archive *format*. When no *file* operands are specified, standard input is read for a list of path names with one per line without any leading or trailing <blanks>.
- a** Append files to the end of an archive that was previously written. If an archive *format* is not specified with a **-x** option, the format currently being used in the archive will be selected. Any attempt to append to an archive in a format different from the format already used in the archive will cause **pax** to exit immediately with a non-zero exit status. The blocking size used in the archive volume where writing starts will continue to be used for the remainder of that archive volume.

- b** *blocksize*
When writing an archive, block the output at a positive decimal integer number of bytes per write to the archive file. The *blocksize* must be a multiple of 512 bytes with a maximum of 32256 bytes. A *blocksize* can end with k or b to specify multiplication by 1024 (1K) or 512. A pair of *blocksizes* can be separated by x to indicate a product. When blocking is not specified, the default blocksize is dependent on the specific archive format being used (see the **-x** option).
- c** Match all file or archive members except those specified by the *pattern* and *file* operands.
- d** Cause files of type directory being copied or archived, or archive members of type directory being extracted, to match only the directory file or archive member and not the file hierarchy rooted at the directory.
- f** *archive*
Specify *archive* as the path name of the input or output archive, overriding the default standard input (for list and read) or standard output (for write). A single archive may span multiple files and different archive devices. When required, **pax** will prompt for the path name of the file or device of the next volume in the archive.
- i** Interactively rename files or archive members. For each archive member matching a *pattern* operand or each file matching a *file* operand, **pax** will prompt to the terminal giving the name of the file, its file mode and its modification time. **pax** then reads a line from the terminal. If this line is blank, the file or archive member is skipped. If this line consists of a single period, the file or archive member is processed with no modification to its name. Otherwise, its name is replaced with the contents of the line. **pax** will immediately exit with a non-zero exit status if EOF is encountered when reading a response. If the LC_TIME environment variable is set, the modification time is formatted using the format specified by the d_t_fmt keyword in the LC_TIME category of the specified locale.
- k** Do not overwrite existing files.
- l** (The lowercase letter ell) Link files. In the copy mode (**-r -w**), hard links are made between the source and destination file hierarchies whenever possible.
- n** Select the first archive member that matches each *pattern* operand. No more than one archive member is matched for each *pattern*. When members of type directory are matched, the file hierarchy rooted at that directory is also matched (unless **-d** is also specified).
- o** Information to modify the algorithm for extracting or writing archive files which is specific to the archive format specified by **-x**. In general, options take the form: name=value.
- p** *string*
Specify one or more file characteristic options (privileges). The *string* is a string specifying file characteristics to be retained or discarded on extraction. The string consists of the specification characters a, e, m, o, and p. Multiple characteristics can be concatenated within the same string and multiple **-p** options can be specified. The meaning of the specification characters are as follows:
 - a** Do not preserve file access times. By default, file access times are preserved whenever possible.
 - e** Preserve everything, the user ID, group ID, file mode bits, file access time, and file modification time. This is intended to be used by someone with all the appropriate privileges in order to preserve all aspects of the files as they are recorded in the archive. The e flag is the sum of the o and p flags.
 - m** Do not preserve file modification times. By default, file modification times are preserved whenever possible.
 - o** Preserve the user ID and group ID.
 - p** Preserve the file mode bits. This intended to be used by a user with regular privileges

who wants to preserve all aspects of the file other than the ownership. The file times are preserved by default, but two other flags are offered to disable this and use the time of extraction instead.

In the preceding list, `preserve` indicates that an attribute stored in the archive is given to the extracted file, subject to the permissions of the invoking process. Otherwise the attribute of the extracted file is determined as part of the normal file creation action. If the preservation of any of these items fails for any reason, `pax` will write a diagnostic message to standard error. Failure to preserve these items affects the final exit status, but will not cause the extracted file to be deleted. If the file characteristic letters in any of the *strings* are duplicated or conflict with each other, the one given last will take precedence. For example, if `-p eme` is specified, file modification times are still preserved.

- s Modify the file or archive member names specified by the *pattern* or *file* operands according to the substitution expression *replstr*, using the syntax of the regular expressions. The format of these regular expressions are:

```
/old/new/[gp]
```

Old is a basic regular expression and new can contain an ampersand (&), n (where n is a digit) back-references, or subexpression matching. The old string may also contain <newline> characters. Any non-null character can be used as a delimiter (/ is shown here). Multiple `-s` expressions can be specified. The expressions are applied in the order they are specified on the command line, terminating with the first successful substitution. The optional trailing `g` continues to apply the substitution expression to the path name substring which starts with the first character following the end of the last successful substitution. The first unsuccessful substitution stops the operation of the `g` option. The optional trailing `p` will cause the final result of a successful substitution to be written to standard error in the following format:

```
<original path name> >> <new path name>
```

File or archive member names that substitute to the empty string are not selected and will be skipped.

- t Reset the access times of any file or directory read or accessed by `pax` to be the same as they were before being read or accessed by `pax`.
- u Ignore files that are older (having a less recent file modification time) than a pre-existing file or archive member with the same name. During read, an archive member with the same name as a file in the file system will be extracted if the archive member is newer than the file. During write, a file system member with the same name as an archive member will be written to the archive if it is newer than the archive member. During copy, the file in the destination hierarchy is replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.
- v During a list operation, produce a verbose table of contents using the format of the `ls` utility with the `-l` option. For path names representing a hard link to a previous member of the archive, the output has the format: <ls -l listing> == <link name> For path names representing a symbolic link, the output has the format: <ls -l listing> = >><link name> Where <ls -l listing> is the output format specified by the `ls` utility when used with the `-l` option. Otherwise for all the other operational modes (read, write, and copy), path names are written and flushed to standard error without a trailing newline as soon as processing begins on that file or archive member. The trailing newline is not buffered, and is written only after the file has been read or written. If the `LC_TIME` environment variable is set, the output time is formatted using the format specified by the `d_t_fmt` keyword in the `LC_TIME` category of the specified locale.
- x Specify the output archive format, with the default format being `ustar`. `pax` currently supports the following formats:

- cpio** The extended cpio interchange format specified in the 1003.2 standard. The default blocksize for this format is 5120 bytes.
- bcpio** The old binary cpio format. The default blocksize for this format is 5120 bytes. This format is not very portable and should not be used when other formats are available.
- sv4cpio**
The System V release 4 cpio. The default blocksize for this format is 5120 bytes.
- sv4crc** The System V release 4 cpio with file crc checksums. The default blocksize for this format is 5120 bytes.
- tar** The old BSD tar format as found in BSD4.3. The default blocksize for this format is 10240 bytes. Path names stored by this format must be 100 characters or less in length. Only regular files, hard links, soft links, and directories will be archived (other file system types are not supported). For backward compatibility with even older tar formats, a **-o** option can be used when writing an archive to omit the storage of directories. This option takes the form: **-o -Cm -write_opt=nodir**
- ustar** The extended tar interchange format specified in the 1003.2 standard. The default blocksize for this format is 10240 bytes. Path names stored by this format must be 250 characters or less in length.

pax will detect and report any file that it is unable to store or extract as the result of any specific archive format restrictions. The individual archive formats may impose additional restrictions on use. Typical archive format restrictions include (but are not limited to): file path name length, file size, link path name length and the type of the file.

- A** Run **pax** as old tar.
- B** Limit the number of bytes written to a single archive volume to bytes. The bytes limit can end with m, k, or b to specify multiplication by 1048576 (1M), 1024 (1K) or 512. A pair of bytes limits can be separated by x to indicate a product.
- C ccsid**
Create the files extracted from the archive in the specified *ccsid*. There must be a valid translation from CCSID 819 to the specified *ccsid*. This option overrides the value of the QIBM_CCSID environment variable.
- D** This option is the same as the **-u** option, except that the file inode change time is checked instead of the file modification time. The file inode change time can be used to select files whose inode information (for example, uid, gid, and so on) is newer than a copy of the file in the destination directory.
- E** Limit the number of consecutive read faults while trying to read a flawed archives. With a positive limit, **pax** will attempt to recover from an archive read error and will continue processing starting with the next file stored in the archive. A limit of 0 will cause **pax** to stop operation after the first read error is detected on an archive volume. A limit of NONE will cause **pax** to attempt to recover from read errors forever. The default limit is a small positive number of retries.

Warning: Using this option with NONE should be used with extreme caution as **pax** may get stuck in an infinite loop on a very badly flawed archive.

- G** Select a file based on its group name, or when starting with a #, a numeric gid. A " can be used to escape the #. Multiple **-G** options may be supplied and checking stops with the first match.
- H** Follow only command line symbolic links while performing a physical file system traversal.
- L** Follow all symbolic links to perform a logical file system traversal.
- P** Do not follow symbolic links, perform a physical file system traversal. This is the default mode.

-T Allow files to be selected based on a file modification or inode change time falling within a specified time range of *from_date* to *to_date* (the dates are inclusive). If only a *from_date* is supplied, all files with a modification or inode change time equal to or younger are selected. If only a *to_date* is supplied, all files with a modification or inode change time equal to or older will be selected. When the *from_date* is equal to the *to_date*, only files with a modification or inode change time of exactly that time will be selected.

When **pax** is in the write or copy mode, the optional trailing field **[c][m]** can be used to determine which file time (inode change, file modification or both) are used in the comparison. If neither is specified, the default is to use file modification time only. The **m** specifies the comparison of file modification time (the time when the file was last written). The **c** specifies the comparison of inode change time (the time when the file inode was last changed; for example, a change of owner, group, mode, and so on). When **c** and **m** are both specified, then the modification and inode change times are both compared. The inode change time comparison is useful in selecting files whose attributes were recently changed or selecting files which were recently created and had their modification time reset to an older time (as what happens when a file is extracted from an archive and the modification time is preserved). Time comparisons using both file times is useful when **pax** is used to create a time based incremental archive (only files that were changed during a specified time range will be archived).

A time range is made up of seven different fields and each field must contain two digits. The format is:

```
[cc[yy[mm[dd[hh]]]]]mm[.ss]
```

where **cc** is the century, **yy** is the last two digits of the year, the first **mm** is the month (from 01 to 12), **dd** is the day of the month (from 01 to 31), **hh** is the hour of the day (from 00 to 23), the second **mm** is the minute (from 00 to 59), and **ss** is the seconds (from 00 to 59). The minute field **mm** is required, while the other fields are optional and must be added in the following order: **hh**, **dd**, **mm**, **yy**, **cc**.

The **ss** field may be added independently of the other fields. Time ranges are relative to the current time, so **-T 1234/cm** would select all files with a modification or inode change time of 12:34 p.m. today or later. Multiple **-T** time range can be supplied and checking stops with the first match.

- U** Select a file based on its user name, or when starting with a **#**, a numeric uid. A **"** can be used to escape the **#**. Multiple **-U** options may be supplied and checking stops with the first match.
- X** When traversing the file hierarchy specified by a path name, do not descend into directories that have a different device ID.
- Y** This option is the same as the **-D** option, except that the inode change time is checked using the path name created after all the file name modifications have completed.
- Z** This option is the same as the **-u** option, except that the modification time is checked using the path name created after all the file name modifications have completed.

The options that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u**, **-v**, **-D**, **-G**, **-T**, **-U**, **-Y**, and **-Z**) interact as follows.

- When extracting files during a read operation, archive members are selected based only on the user specified *pattern* operands as modified by the **-c**, **-n**, **-u**, **-D**, **-G**, **-T**, **-U** options. Then any **-s** and **-i** options will modify in that order, the names of these selected files. Then the **-Y** and **-Z** options will be applied based on the final path name. Finally the **-v** option will write the names resulting from these modifications.
- When archiving files during a write operation, or copying files during a copy operation, archive members are selected based only on the user specified path names as modified by the **-n**, **-u**, **-D**, **-G**, **-T**, and **-U** options (the **-D** option only applies during a copy operation). Then any **-s** and **-i** options

will modify in that order, the names of these selected files. Then during a copy operation the **-Y** and the **-Z** options will be applied based on the final path name. Finally the **-v** option will write the names resulting from these modifications.

- When one or both of the **-u** or **-D** options are specified along with the **-n** option, a file is not considered selected unless it is newer than the file to which it is compared.

Operands

The *directory* operand specifies a destination directory path name. If the *directory* operand does not exist, or it is not writable by the user, or it is not of type directory, **pax** will exit with a non-zero exit status.

The *pattern* operand is used to select one or more path names of archive members. When the *pattern* operand is not supplied, all members of the archive will be selected. When a pattern matches a directory, the entire file hierarchy rooted at that directory will be selected. When a *pattern* operand does not select at least one archive member, **pax** will write these *pattern* operands in a diagnostic message to standard error and then exit with a non-zero exit status.

The *file* operand specifies the path name of a file to be copied or archived. When a *file* operand does not select at least one archive member, **pax** will write these *file* operand path names in a diagnostic message to standard error and then exit with a non-zero exit status.

Environment Variables

pax is affected by the following environment variables:

LANG

Provides a default value for locale categories that are not specifically set with a variable starting with LC_.

LC_TIME

Defines the date and time format used in displaying file times.

QIBM_CCSID

pax creates the file extracted from the archive in the CCSID specified by the value of the environment variable.

Exit Status

- 0 All files were processed successfully
- 1 An error occurred

Related information

- “compress - Compress data” on page 94
- “tar - File archiver” on page 126

Examples

1. Copy the contents of the current directory to an archive file:

```
pax -w -f saved.ar
```

2. Display the verbose table of contents for an archive file:

```
pax -r -v -f saved.ar
```

3. The following commands copy the entire directory tree anchored at /home/abbey/olddir to /home/abbey/newdir:

```
mkdir /home/abbey/newdir  
cd /home/abbey/olddir  
pax -rw . /home/abbey/newdir
```

4. Interactively select the files to copy from the current directory to the directory destination:

```
pax -rw -i . destination
```

5. Extract all files from an archive file that are owned by user root and group bin and preserve all file permissions:

```
pax -r -pe -U root -G bin -f saved.ar
```

6. List and update only those files in the destination directory /backup which are older than files with the same name found in the source directory /sourcecode:

```
pax -r -w -v -Y -Z /sourcecode /backup
```

pr - Print files

Synopsis

```
pr [+page] [-column] [-adFmrt] [-e [char][gap]] [-h header] [-i[char][gap]] [-l line] [-n[char][width]] [-o offset] [-s[char]] [-w width] [-] [file ...]
```

Description

The **pr** utility is a printing and pagination filter for text files. When multiple input files are specified, each is read, formatted, and written to standard output. By default, the input is separated into 66-line pages, each with a 5-line header with the page number, date, time, and the path name of the file and a 5-line trailer consisting of blank lines. If the LC_TIME environment variable is set, the date and time in the header is formatted using the format specified by the d_t_fmt keyword in the LC_TIME category of the specified locale.

When multiple column output is specified, text columns are of equal width. By default text columns are separated by at least one <space>. Input lines that do not fit into a text column are truncated. Lines are not truncated under single column output.

Error messages are written to standard error during the printing process (if output is redirected) or after all successful file printing is complete (when printing to a terminal).

If **pr** receives an interrupt while printing to a terminal, it flushes all accumulated error messages to the screen before terminating.

Options

Notes:

1. In the following option descriptions, *column*, *lines*, *offset*, *page*, and *width* are positive decimal integers and *gap* is a nonnegative decimal integer.
2. The **-s** option does not allow the option letter to be separated from its argument.
3. The **-e**, **-i**, and **-n** options require that both arguments, if present, not be separated from the option letter.

+page Begin output at page number *page* of the formatted input.

-column

Produce output that is *columns* wide (default is 1) that is written vertically down each column in the order in which the text is received from the input file. The options **-e** and **-i** are assumed. This option should not be used with the **-m** option. When used with the **-t** option the minimum number of lines is used to display the output.

-a Modify the effect of the **column** option so that the columns are filled across the page in a round-robin order (for example, when column is 2, the first input line heads column 1, the second heads column 2, the third is the second line in column 1, and so on). This option requires the use of the **column** option.

-d Produce output that is double spaced. An extra <newline> character is output following every <newline> found in the input.

- e** [*char*][*gap*]
Expand each input <tab> to the next greater column position specified by the formula $n*gap+1$, where n is an integer > 0. If *gap* is zero or is omitted the default is 8. All <tab> characters in the input are expanded into the appropriate number of <space>s . If any nondigit character, *char*, is specified, it is used as the input tab character.
- F** Use a <form-feed> character for new pages, instead of the default behavior that uses a sequence of <newline> characters.
- h** *header*
Use the string *header* to replace the file name in the header line.
- i** [*char*][*gap*]
In output, replace multiple <space>s with <tab>s whenever two or more adjacent <space>s reach column positions $gap+1$, $2*gap+1$, and so on. If *gap* is zero or omitted, default <tab> settings at every eighth column position is used. If any nondigit character, *char*, is specified, it is used as the output <tab> character.
- l** *lines* Override the 66 line default and reset the page length to *lines*. If *lines* is not greater than the sum of both the header and trailer depths (in lines), the **pr** utility suppresses output of both the header and trailer, as if the **-t** option were in effect.
- m** Merge the contents of multiple files. One line from each file specified by a file operand is written side by side into text columns of equal fixed widths, in terms of the number of column positions. The number of text columns depends on the number of file operands successfully opened. The maximum number of files merged depends on page width and the per process open file limit. The options **-e** and **i** are assumed.
- n** [*char*][*width*]
Provide *width* digit line numbering. The default for *width*, if not specified, is 5. The number occupies the first *width* column positions of each text column or each line of **-m** output. If *char* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows. The default for *char* is a <tab>. Line numbers longer than *width* columns are truncated.
- o** *offset*
Each line of output is preceded by *offset* <spaces>s. If this option is not specified, the default is zero. The space taken is in addition to the output line width.
- r** Write no diagnostic reports on failure to open a file.
- s** *char* Separate text columns by the single character *char* instead of by the appropriate number of <space>s (default for *char* is the <tab> character).
- t** Print neither the five-line identifying header nor the five-line trailer typically supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.
- w** *width*
Set the width of the line to *width* column positions for multiple text-column output only. If this option is not specified and the **-s** option is not specified, the default width is 72. If this option is not specified and the **-s** option is specified, the default width is 512.

Operands

Each *file* is a path name of a file to be printed. If no *file* operands are specified, or if a *file* operand is -, the standard input is used.

Environment Variables

pr is affected by the following environment variables:

LANG

Provides a default value for locale categories that are not specifically set with a variable starting with LC_.

LC_TIME

Defines the format of the date and time used in writing header lines.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- “cat - Concatenate and print files” on page 86
- “od - Dump files in various formats” on page 110

Examples

1. Print a file starting at page 3:

```
pr +3 source.java
```
2. Print every *.java file and change the header message:

```
pr -h 'JDK source files and examples' code/*.java
```

pwd - Return working directory name

Synopsis

pwd

Description

You can use **pwd** to display the working directory on standard output.

Options

None.

Operands

None.

Exit Status

- 0 when successful.

Related information

- “cd - Change working directory” on page 88
- “pwdx - Print working directory expanded”

pwdx - Print working directory expanded

Synopsis

pwdx

Description

You can use **pwdx** to display the working directory with symbolic links expanded on standard output.

Exit Status

- 0 when successful

Related information

- “cd - Change working directory” on page 88
- “pwd - Return working directory name” on page 121

Rfile - Read or write record files

Synopsis

Rfile -r | -w | -h [-abKlqQs] [-c CL-command] [-C CL-command] file ...

Description

The **Rfile** utility reads i5/OS^(TM) record files (database or device files) and writes the data to standard output, or reads standard input and writes the data to record files.

Note: This utility is unique to i5/OS^(TM).

Options

- a** Append the contents of standard input to the record file. This option only applies when **-w** is specified. If **-w** is specified without **-a**, any physical file member is cleared before writing the contents of the stream.
- b** Process binary data. This option prevents normal processing for newline characters in the input or output stream. When **-b** is omitted, newline characters are removed from standard input lines written to a record file, and newline characters are inserted at the end of records written to standard output.
- c CL-command**
Run a CL command in the utility process before processing any record file. This option can be used to run a CL override command that specifies device-dependent parameters for a record file. If more than one **-c** option is specified, the CL commands are processed in sequence before processing any record file.
- C CL-command**
Run a CL command in the utility process after processing all record files. If more than one **-C** option is specified, the CL commands are processed in sequence after processing all record files.
- h** Write a brief description of command syntax to standard error.
- K** Keep the job log at job termination. The system normally deletes the job log after running a QShell utility. This option forces the system to produce a job log listing (which may assist with problem determination) when the job that runs **Rfile** ends.
- l** Do not truncate long text lines. This option only applies to text data. When **-l** is specified, any standard input line longer than one output record is folded onto as many records as necessary, and no trailing blanks are removed from records written to standard output.
- q** Suppress warning messages. This option suppresses messages normally written to standard error when long text lines are truncated or folded in the output file.
- Q** Use i5/OS^(TM) qualified name syntax for file names. When this option is specified, the file names specified as command operands are i5/OS^(TM) qualified names (instead of Integrated File System path names).
- r** Read the specified record files and write their contents to standard output. Either **-r** or **-w**, but not both, must be specified.

- s Process source sequence number and date fields as text. This option only applies to text processing of FILETYPE(*SRC) record files. When -s is specified, the entire contents of every record is processed as a text line. If -s is omitted, the first 12 bytes is stripped from every source record read, and the first 12 bytes of every source record written is filled with a sequence number and zeros for the date field.
- w Read standard input and write its contents to the specified record file. The output file must already exist, or an error is reported (and no file is created). Either -r or -w, but not both, must be specified.

Operands

At least one i5/OS^(TM) record file name must be specified. If more than one file is specified, they are processed in sequence as end of file is reached on each input source. When option -Q is omitted, files are identified by path names in the Integrated File System. If option -Q is specified, file names are specified in any of these forms:

```
file
library/file
'file(member) '
'library/file(member) '
```

If the library name is omitted or *LIBL is specified for the library name, the file is located using the job library list. If the member name is omitted or *FIRST is specified as the member name, the first member of a database file is opened. Specifying *LAST for the member name opens the last member of a database file. Member name *ALL can be used with option -r to read all members of a database file (from first to last). Member names are ignored for device files (when specified in i5/OS^(TM) qualified name form).

Examples

1. Read the contents of source database member QSYSINC/H(SQLCLI), and write it to standard output. Trailing blanks are removed from each line, as are the first 12 characters of each line (containing sequence number and date information):

```
Rfile -rQ 'qsysinc/h(sqlcli)'
```

2. Write the contents of stream file mydoc.ps to spooled printer device file QPRINT as unconverted ASCII data, and then use the CL LPR command to send the spool file to another system:

```
before='ovrprtq qprint devtype(*userascii) spool(*yes) '
after="lpr file(qprint) system(usrchprt01) prtq('rhdps') transform(*no) "
cat -c mydoc.ps | Rfile -wbQ -c "$before" -C "$after" qprint
```

3. Copy the contents of save file INSAVF in library QGPL to another save file named OUTSAVF located using the job library list. Note that the data is read and written in binary mode to avoid ASCII/EBCDIC conversion and newline processing:

```
Rfile -rb /qsys.lib/qgpl.lib/insavf.file | Rfile -wbQ outsavf
```

rm - Remove directory entries

Synopsis

```
rm [-f | -i] [-dPRr] file ...
```

Description

The **rm** utility attempts to remove the non-directory type *files* specified on the command line. If the permissions of the *file* do not permit writing, and the standard input device is a terminal, the user is prompted (on standard error) for confirmation.

The **rm** utility removes symbolic links, not the files referenced by the links.

It is an error to attempt to remove the files "." and "..".

Options

- d** Attempt to remove directories as well as other types of files.
- f** Attempt to remove the *files* without prompting for confirmation, regardless of the file's permissions. If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error. The **-f** option overrides any previous **-i** options.
- i** Request confirmation before attempting to remove each *file*, regardless of the file's permissions, or whether the standard input device is a terminal. If the response from the standard input begins with the first character for the YES response in the current locale, the *file* is removed. The **-i** option overrides any previous **-f** options.
- P** Overwrite regular files before deleting them. Files are overwritten three times, first with the byte pattern 0xff, then 0x00, and then 0xff again, before they are deleted.
- R** Attempt to remove the file hierarchy rooted in each *file* argument. The **-R** option implies the **-d** option. If the **-i** option is specified, the user is prompted for confirmation before each directory's contents are processed (as well as before the attempt is made to remove the directory). If the user does not respond affirmatively, the file hierarchy rooted in that directory is skipped.
- r** Equivalent to **-R**.

Exit Status

- 0 if all of the named files or file hierarchies were removed, or if the **-f** option was specified and all of the existing files or file hierarchies were removed.
- >0 if an error occurs.

Related information

- "cp - Copy files" on page 95
- "ln - Link files" on page 104
- "ls - List directory contents" on page 105
- "mv - Move files" on page 110
- "rmdir - Remove directories"

Examples

1. Remove all the files and the directory "java", as well as any subdirectories or files, or both, and do not prompt for conformation.

```
rm -r -f /home/bob/examples/code/java
```
2. Remove the files "file1", "file2" and "file3".

```
rm file1 file2 file3
```

rmdir - Remove directories

Synopsis

rmdir *directory* ...

Description

The **rmdir** utility removes the directory entry specified by each *directory* argument, provided it is empty.

Arguments are processed in the order given. In order to remove both a parent directory and a subdirectory of that parent, the subdirectory must be specified first so the parent directory is empty when **rmdir** tries to remove it.

Exit Status

- 0 if each directory entry specified referred to an empty directory and was removed successfully.
- >0 An error occurred.

Related information

- “mkdir - Make directories” on page 108
- “rm - Remove directory entries” on page 123

setccsid - Set CCSID attribute for file

Synopsis

```
setccsid [-R [-H | -L | -P]] [-h] ccsid file ...
```

Description

The **setccsid** utility sets the CCSID attribute for the specified *files* to the specified *ccsid*. The data contained in *file* is not changed.

Options

- H** If the **-R** option is specified, symbolic links on the command line are followed. Symbolic links encountered in the tree traversal are not followed.
- L** If the **-R** option is specified, both symbolic links on the command line and symbolic links encountered in the tree traversal are followed.
- P** If the **-R** option is specified, no symbolic links are followed.
- R** If *file* designates a directory, **setccsid** sets the CCSID of each file in the entire subtree connected at that point.
- h** Set the CCSID of a symbolic link instead of the file pointed to by the symbolic link.

Operands

The *ccsid* is an integer number identifying the coded character set id. Each *file* is a pathname of a file to set the CCSID.

Related information

- “attr - Get or set attributes for files” on page 81
- “chmod - Change file modes” on page 90
- “chown - Change file ownership” on page 93
- “iconv - Convert characters from one CCSID to another CCSID” on page 68
- “touch - Change file access and modification times” on page 128

Examples

1. Set the CCSID of the files “file1” and “file2” to 819:

```
setccsid 819 file1 file2
```

tail - Display the last part of a file

Synopsis

```
tail [-f | -r] [-b number | -c number | -k number | -n number] [file ...]
```

Description

The **tail** utility displays the contents of *file* or, by default, standard input, to the standard output.

The display begins at a byte, line, 512-byte block, or kilobyte location in the input. *Numbers* having a leading plus sign (+) are relative to the beginning of the input, for example, "-c +2" starts the display at the second byte of the input. *Numbers* having a leading minus sign (-) or no explicit sign are relative to the end of the input, for example, "-n 2" displays the last two lines of the input. The default starting location is "-n 10", or the last 10 lines of the input.

If more than one *file* is specified, each file is preceded by a header consisting of the string "==> XXX <==" where XXX is the name of the file.

Note: **tail** does not support large files (files greater than 2GB in size).

Options

-b *number*

The location is *number* 512-byte blocks.

-c *number*

The location is *number* bytes.

-f

Causes **tail** to not stop when end of file is reached, but rather to wait for additional data to be appended to the input. The **-f** option is ignored if the standard input is a pipe, but not if it is a FIFO.

-k *number*

The location is *number* kilobytes.

-n *number*

The location is *number* lines.

-r

Causes the input to be displayed in reverse order, by line. Additionally, this option changes the meaning of the **-b**, **-c** and **-n** options. When the **-r** option is specified, these options specify the number of bytes, lines or 512-byte blocks to display, instead of the bytes, lines or blocks from the beginning or end of the input from which to begin the display. The default for the **-r** option is to display all of the input.

Exit Status

- 0 on success
- >0 if an error occurs

Related information

- "cat - Concatenate and print files" on page 86
- "head - Copy the first part of files" on page 104

Examples

1. Display the last 100 lines from the file "donkeys". If the file "donkeys" is less than 100 lines, then **tail** displays the entire file.

```
tail -n 100 donkeys
```

tar - File archiver

Synopsis

```
tar -crtux[befmopvwHLPX] [archive] [blocksize] file ...
```

Description

The **tar** utility reads, writes, and lists files from an archive file.

Options

The following options select the function **tar** performs. One of these options must be specified.

- c Create a new archive.
- r Add the specified *file* to end of the archive.
- t List the names of the files in the archive to standard output.
- u Update the specified *file* in the archive if it has been modified since last written to the archive or add *file* to the archive if it is not in the archive.
- x Extract the specified *files* from the archive. If no *files* are specified, all files are extracted from the archive.

The following options affect the operation of **tar**.

- b Use the first operand (or the second, if **f** has already been specified) as the block size for the archive.
- e Exit after the first error is found.
- f Use the first operand (or the second, if **b** has already been specified) as the name of the archive instead of the default name. If the name of the file is -, **tar** writes to the standard output or reads from the standard input depending on the function.
- m Do not restore the modification times. The modification time of the file is the time of extraction.
- o Set the owner and group of extracted files to the user running **tar** instead of to the user and group saved with the archive.
- p Preserve the owner, group, file mode, access time, and modification time of files extracted from the archive.
- v Verbose mode. Write to standard error the name of each file being processed. When the **t** function is specified, the output also includes the mode, number of links, owner, group, size, and modification date of each file.
- w Write the action to be taken, followed by the name of the file, and then wait for the user's confirmation. If an affirmative response is given, the action is performed. Any other input suppresses the action.
- H Follow only command line symbolic links while performing a physical file system traversal.
- L Follow all symbolic links to perform a logical file system traversal.
- P Do not follow symbolic links, perform a physical file system traversal. This is the default mode.
- X When traversing the file hierarchy specified by a path name, do not descend into directories that have a different device ID.

Operands

Each *file* is an object that is either added to the archive or extracted from the archive depending on the function.

Environment Variables

tar is affected by the following environment variables:

QIBM_CCSID

The value of the environment variable is the CCSID used to create files extracted from the archive. There must be a valid translation from CCSID 819 to the specified CCSID.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “pax - Portable archive interchange” on page 112

touch - Change file access and modification times

Synopsis

```
touch [-acfm] [-r ref_file] [-t [[CC]YY]MMDDhhmm[.SS] ] [-C ccsid] file ...
```

Description

The **touch** utility sets the modification and access times of files to the current time of day. If the *file* doesn't exist, it is created with default permissions.

Options

- a** Change the access time of *file*. The modification time of the file is not changed unless the **-m** flag is also specified.
- C *ccsid***
If *file* does not exist, create the file with the specified *ccsid*. This option overrides the value of the QIBM_CCSD environment variable.
- c** Do not create *file* if it does not exist. The **touch** utility does not treat this as an error. No error messages are displayed and the exit value is not affected.
- f** Attempt to force the update, even if the file permissions do not currently permit it.
- m** Change the modification time of *file*. The access time of the file is not changed unless the **-a** flag is also specified.
- r *ref_file***
Use the access and modifications times from the specified *ref_file* instead of the current time of day.
- t** Change the access and modification times to the specified time. The argument should be in the form:
[[CC]YY]MMDDhhmm[.SS]

where each pair of letters represents the following:

- CC** The first two digits of the year (the century).
- YY** The second two digits of the year. If YY is specified, but CC is not, a value for CC between 69 and 99 results in a YY value of 19. Otherwise, a CC value of 20 is used.
- MM** The month of the year, from 1 to 12.
- DD** The day of the month, from 1 to 31.
- hh** The hour of the day, from 0 to 23.
- mm** The minute of the hour, from 0 to 59.
- SS** The second of the minute, from 0 to 59.

If the CC and YY letter pairs are not specified, the values default to the current year. If the SS letter pair is not specified, the value defaults to 0.

Environment Variables

touch is affected by the following environment variables:

QIBM_CCSID

If *file* does not exist, **touch** creates the file with the CCSID specified by the value of the environment variable.

Exit Status

- 0 on success
- >0 if an error occurs

Examples

1. Change the time-date stamp of the file *myfile* to match the time-date stamp of the file *yourfile*.

```
touch -r yourfile myfile
```
2. Change the time-date stamp of the file *myfile* to a specific time-date stamp.

```
touch -t 200001010000.00 myfile
```

umask - Get or set the file mode creation mask

Synopsis

```
umask [ -S ] [ mask ]
```

Description

You can use **umask** to set or display the file creation mask. The mask allows you to control the file permission bits that are set when creating a file or directory.

If you specify *mask*, **qsh** sets the file creation mask to *mask*. If you do not specify *mask*, **qsh** displays the current file creation mask on standard output.

Options

- **-S** Use symbolic permissions.

Operands

When using symbolic permissions, *mask* is an expression that defines which permissions should not be removed. A symbolic permission is an expression with the format [*who*] *op* [*permission*] where:

- *who* is a combination of the letters:
 - **u** for owner permissions.
 - **g** for group permissions
 - **o** for other (or public) permissions
 - **a** for all permissions (the default value).
- *op* is one of the following:
 - **-** (minus) to delete the permission.
 - **+** (plus) to add the permission.
- *permission* is one or more of the following:
 - **r** for read permission.
 - **w** for write permission.
 - **x** for execute or search permission.

Exit Status

- 0 when successful.
- >0 when *mask* is invalid.

Related information

- “chmod - Change file modes” on page 90
- “touch - Change file access and modification times” on page 128

Examples

1. Display the current file creation mask in symbolic form: **umask -S**
2. Display the current file creation mask: **umask**
3. Set the file creation mask to remove read permission for others: **umask 004**
4. Set the file creation mask to remove write permission for group: **umask -S g-w**

uncompress - Expand compressed data

Synopsis

uncompress [-cv] [-b *bits*] [*file* ...]

Description

The **uncompress** utility restores the compressed *files* to their original form, renaming the files by deleting the “.Z” extension.

If renaming *file* would cause files to be overwritten and the standard input device is a terminal, the user is prompted (on standard error) for confirmation. If prompting is not possible or confirmation is not received, the files are not overwritten.

Options

- b *bits*** Specify the bits code limit (see below for details).
- c** Uncompressed output is written to the standard output. No files are modified.
- v** Print the percentage of expansion for each file.

Operands

Each *file* is a pathname of a file to uncompress. If no *files* are specified, the standard input is uncompressed to the standard output. If either the input and output files are not regular files, the checks for reduction in size and file overwriting are not performed, the input file is not removed, and the attributes of the input file are not retained.

Extended Description

The **uncompress** utility uses a modified Lempel-Ziv algorithm. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (the default is 16). Bits must be between 9 and 16.

The amount of compression obtained depends on the size of the input, the number of bits per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50-60%.

Exit Status

- 0 on success

- >0 if an error occurs.

Related information

- “compress - Compress data” on page 94
- “zcat - Expand and concatenate data”

zcat - Expand and concatenate data

Synopsis

`zcat [file ...]`

Description

The `zcat` utility expands the compressed data from the specified *files* and the uncompressed output is written to standard output.

Operands

Each *file* is a pathname of a file that contains compressed data.

Exit Status

- 0 on success
- >0 if an error occurs.

Related information

- “compress - Compress data” on page 94
- “uncompress - Expand compressed data” on page 130

Utilities for reading and writing input and output

The following are Qshell utilities for reading and writing input and output:

- “dspmsg - Display message from message catalog”
- “echo - Write arguments to standard output” on page 132
- “print - Write output” on page 133
- “printf - Write formatted output” on page 133
- “read - Read a line from standard input” on page 135

dspmsg - Display message from message catalog

Synopsis

`dspmsg [-n] [-s set] catalog msgid [defaultMsg [arguments ...]]`

Description

The `dspmsg` utility displays a message from a message catalog created by the GENCAT CL command. The message is written to standard output. The `dspmsg` utility can be used as a replacement for `echo` or `print` when a script needs to display messages that are translated to multiple languages.

Options

- n Display the specified message with no substitution.
- s set Retrieve the message from the specified *set* in the message catalog. The default value for *set* is 1.

Operands

The *catalog* operand specifies the path name to a message catalog. If the catalog is specified using a relative path name, the NLSPATH variable and the LC_MESSAGES locale category are used to find the catalog.

The *msgid* operand specifies the message identifier to retrieve from the message catalog.

When the specified *catalog* or *msgid* is not found, the optional *defaultMsg* is displayed instead. If the *defaultMsg* operand is not specified, a system generated message is displayed.

The optional *arguments* are substituted into the output message if it contains the %, %n\$, %ld, or %n\$ld printf() conversion specifications. Any other conversion specifications are not valid. Also, the normal control character escapes (for example, \n) are supported.

Exit Status

- 0 if successful
- >0 if an error occurred.

Related information

- “echo - Write arguments to standard output”
- “gencat - Generate a formatted message catalog” on page 101
- “print - Write output” on page 133
- “printf - Write formatted output” on page 133

Examples

1. Display message 5 from catalog mycat.
dspmmsg mycat 5 "Message not found" hello

echo - Write arguments to standard output

Synopsis

echo [*arg* ...]

Description

You can use **echo** to display each *arg* on standard output separated by a space character and followed by a newline character.

Operands

Each *arg* is echoed on standard output.

Exit Status

- 0 when successful
- >0 when an error occurs

Related information

- “dspmmsg - Display message from message catalog” on page 131
- “print - Write output” on page 133
- “printf - Write formatted output” on page 133
- “tee - Duplicate standard input” on page 182

print - Write output

Synopsis

```
print [ -nrR ] [ -u [ n ] ] [ argument ... ]
```

Description

You can use **print** to display each *argument* on standard output separated by a <space> character and followed by a <newline> character.

Unless you specify **-r** or **-R**, **print** formats the output using the following conventions:

- **\a** Bell.
- **\b** Backspace.
- **\c** Print without adding newline character. The remaining *arguments* are ignored.
- **\f** Formfeed.
- **\n** Newline.
- **\r** Return.
- **\t** Tab.
- **\v** Vertical tab.
- **** Backslash.
- **\0x** The character whose EBCDIC code is the 1, 2, or 3-digit octal number *x*.

Options

- n** Do not add a trailing newline character to the output.
- r** Do not use the conventions listed above.
- R** Do not use the conventions listed above.
- u *n*** Write output to descriptor *n* if specified or descriptor 1 by default. The descriptor must be 1, 2, or one you opened with **exec**.

Operands

Each *argument* is printed on standard output.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “**dspmsg** - Display message from message catalog” on page 131
- “**exec** - Run commands and open, close, or copy descriptors” on page 53
- “**echo** - Write arguments to standard output” on page 132
- “**printf** - Write formatted output”

printf - Write formatted output

Synopsis

```
printf format [ argument ... ]
```

Description

You can use **printf** to format and display output on standard output. The syntax is similar to the ILE C printf() function. **printf** formats using the following conversion control string syntax:

`%[flags][width].[precision]conversion`

conversion specifies how the corresponding *argument* is displayed. You must specify one of the following conversion characters:

- c** Unsigned character.
- d** Signed decimal number.
- e,E** Scientific notation.
- f** Floating point number.
- g,G** Scientific notation with significant digits.
- i** Signed decimal number.
- o** Unsigned octal number.
- s** String.
- u** Unsigned decimal number.
- x** Unsigned hexadecimal number with digits 0123456789abcdef.
- X** Unsigned hexadecimal number with digits 0123456789ABCDEF.

flags control how the *argument* is displayed in the following ways:

- (minus)**
Left align *argument* within the field.
- + (plus)**
Prefix all numbers with a + or -.
- space** Prefix positive numbers with <space> and negative numbers with -.
- 0** Pad field width with leading zeros for **d**, **e**, **E**, **f**, **g**, or **G**.
- #** Use an alternate output form depending on conversion character. For **o**, prefix octal numbers with "0". For **x**, prefix hexadecimal numbers with "0x". For **X**, prefix hexadecimal numbers with "0X". For **e**, **E**, **f**, **g**, or **G**, display decimal point. For **g** or **G**, display trailing zeros.

width is the minimum number of character positions displayed. Using an asterisk (*) character for the width means the value of the next *argument* is the field width.

The meaning of *precision* depends on the conversion character.

- For **d**, **i**, **o**, **u**, **x**, or **X** *precision* specifies the minimum number of digits to be displayed.
- For **e**, **E**, or **f** *precision* specifies the number of digits to be displayed after the decimal point.
- For **g**, or **G** *precision* specifies the maximum number of significant digits.
- For **s** *precision* specifies the maximum number of characters to be displayed.

Options

None.

Operands

Each *argument* is converted and displayed as specified by the *format*.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “dspmmsg - Display message from message catalog” on page 131
- “echo - Write arguments to standard output” on page 132
- “print - Write output” on page 133

read - Read a line from standard input

Synopsis

```
read [ -r ] [ -p prompt ] [ -u [ n ] ] [ name ... ]
```

Description

You can use **read** to read a line and split it into fields using the characters from the **IFS** variable as delimiters. By default, a backslash (\) at the end of a line causes the line to be continued on the next line. **qsh** removes both the backslash and the <newline>.

Options

-p *prompt*

When the interactive option is set, display *prompt* on stderr.

-r A backslash at the end of a line does not mean continue the line.

-u *n* Read from descriptor *n* if specified or descriptor 0 by default. The descriptor must be 0 or one that you opened with **exec**.

Operands

Each *name* is assigned to the corresponding field from the input line. Any leftover fields are assigned to the last *name*. The default name is the **REPLY** variable.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “exec - Run commands and open, close, or copy descriptors” on page 53
- “print - Write output” on page 133

Examples

1. Read a line from stdin after displaying a prompt: **read -p** `Enter a name: ` *firstname lastname*
2. Read a line from descriptor 5: **read -u5**

Utilities for developing Java™ programs

The following are Qshell utilities for developing Java™ programs:

- “ajar - Alternative Java™ archive” on page 136
- “appletviewer - View Java™ applet” on page 139
- “extcheck - A utility to detect JAR conflicts” on page 139
- “jar - Archive Java™ files” on page 139

- “jarsigner - JAR signing and verification” on page 140
- “java - Run Java™ interpreter” on page 140
- “javac - Compile a Java™ program” on page 141
- “javadoc - Generate Java™ documentation” on page 141
- “javah - Generate C header or stub file” on page 141
- “javakey - Manage Java™ security keys and certificates” on page 142
- “javap - Disassemble a compiled Java™ program” on page 142
- “keytool - Key and certificate management tool” on page 142
- “native2ascii - Convert native characters to ASCII” on page 143
- “policytool - Policy file creation and management tool” on page 143
- “rmic - Compile Java™ RMI stubs” on page 143
- “rmid - The Java™ RMI activation system” on page 143
- “rmiregistry - Start a remote object registry” on page 143
- “serialver - Return serial version” on page 143
- “tnameserv - Naming service” on page 144

ajar - Alternative Java™ archive

Synopsis

ajar {-h | —help}

ajar {-l | —list} [-v | —verbose] [-q | —quiet] *jarfile* [{*file* | *pattern*} ...] [{-x | -i} {*file* | *pattern*} ...] ...

ajar {-x | —extract} [-v | —verbose] [-q | —quiet] [-N | —neverWrite] [-p | —pipe] *jarfile* [{*file* | *pattern*} ...]
 [{-x | -i} {*file* | *pattern*} ...] ...

ajar {-c | —create} [-0 | —store-only] [-v | —verbose] [-r | —recurse] [-@ | —stdin] [-D | —nodirs] [-q | —quiet]
 [{-m | —manifest} *mffile*] [-M | —no-manifest] [{-n | —no-deflate} *suffix..*] *jarfile file ...* [{-x | -i} {*file* | *pattern*} ...] ...

ajar {-a | —add} [-0 | —store-only] [-v | —verbose] [-r | —recurse] [-@ | —stdin] [-D | —nodirs] [-q | —quiet]
 [{-m | —manifest} *mffile*] [-M | —no-manifest] [{-n | —no-deflate} *suffix..*] *jarfile file ...* [{-x | -i} {*file* | *pattern*} ...] ...

ajar {-d | —delete} [-v | —verbose] [-q | —quiet] [{-m | —manifest} *mffile*] [-M | —no-manifest] *jarfile*
 {*file* | *pattern*} ... [{-x | -i} {*file* | *pattern*} ...] ...

Description

ajar may be used as an alternative interface for creating and manipulating Java™ Archive (JAR) files. The **ajar** utility combines several of the features found in zip/unzip tools with those of the IBM^(R) Developer Kit for Java **jar** tool. Use **ajar** instead of the **jar** command when you need a zip or unzip like interface.

Like the **jar** tool, **ajar** lists the contents of jar files, extracts from jar files, creates new jar files and supports many of the zip formats.. Additionally, **ajar** supports adding and deleting files in existing jars.

Actions

-h | —help

Writes command syntax to stdout.

- l | —list**
Writes table of contents to stdout.
- x | —extract**
Extracts files to the current directory.
- c | —create**
Creates a new archive.
- a | —add**
Adds new files to the archive and replaces existing files.
- d | —delete**
Deletes files from the archive.

Options

- @ | —stdin**
Read file list from stdin. The file list consists of parameters that would normally follow the *jarfile* parameter on the command line. The file list may consist of multiple lines with one item per line and no leading blanks. Comments begin with '#' and extend to the end of the line.
- 0 | —store-only**
Store only. Do not compress/deflate files. Used when adding files and creating jars.
- m | —manifest**
Include manifest information from the specified file.
- n | —no-deflate**
Do not deflate files with the specified suffixes. The list of suffixes must be terminated by another option or "—". See examples below.
- p | —pipe**
Extract to stdout.
- q | —quiet**
Quiet mode. Do not write informational and warning messages.
- r | —recurse**
Recurse into directories. Used when adding files and creating jars.
- v | —verbose**
Verbose mode. Write diagnostic information to stderr.
- D | —nodirs**
Suppress directory entries. Used when adding files and creating jars.
- M | —no-manifest**
Do not create a manifest.
- N | —neverWrite**
Never overwrite any files when extracting.

Operands

The *jarfile* operand specifies the pathname of the jar file being operated on. *jarfile* must be an Integrated File System (IFS) name.

The *file* operand specifies the pathname of a file or directory. *file* must be an IFS name.

The *pattern* operand specifies a pattern to match pathnames of files and directories. *pattern* will match to IFS names. A *pattern* is a sequence of characters containing one or more of the following meta characters:

- * matches 0 or more characters

- ? matches any single character
- [...] matches any single character found within the brackets where "... " represents a set of characters. Ranges are specified by a beginning character, a hyphen, and an ending character. An exclamation (!) or caret (^) following the left bracket means the characters within the brackets are complemented (match anything except the characters within the brackets).

Patterns must be contained within quotation marks or the meta characters must be preceded by a back slash ('\') to prevent Qshell from interpreting them.

The *file* and *pattern* operands are used to select the files to be acted upon. Selected files are determined using three sets of files, a candidate set, an exclusion set, and an inclusion set.

candidate set

The candidate set is determined using the operands listed after *jarfile* and before any *-x* or *-i*. For the list and extract actions the candidate set defaults to all files contained in the jar file. For all other actions there is no default value for the candidate set.

exclusion set

The exclusion set is determined using all lists of *file* and *pattern* operands preceded by a *-x* and followed by another *-x*, a *-i* or the end of the command string. The exclusion set defaults to the empty set.

inclusion set

The inclusion set is determined using all lists of *file* and *pattern* operands preceded by a *-i* and followed by another *-i*, a *-x* or the end of the command string. The inclusion set defaults to all files in the candidate set.

All candidate files are selected that are in the inclusion set and not in the exclusion set.

Exit Status

- 0 when all files were processed successfully
- >0 when an error occurred

Examples

1. To list all files in a jar file named myjar which is located in the current directory: `ajar -l myjar`
2. To list all .java files in myjar: `ajar -l myjar *.java`
3. To extract all files from myjar into the current directory: `ajar -x myjar`
4. To create a jar named myjar containing all directories and files in the file system hierarchy rooted in the current directory (Note in this example Qshell interprets the '*' and expands it so that the list of candidate files contains all files and directories in the current directory.): `ajar -c -r myjar *`
5. To create a jar named myjar containing entries for only the files in the current directory: `ajar -c -D myjar *`
6. To create the same jar file without a manifest (which is a zip file for all practical purposes): `ajar -c -D -M myjar *`
7. To create a jar named myjar containing all files except .java files in the file system hierarchy rooted in the current directory: `ajar -c -r myjar * -x *.java`
8. To create a jar named myjar containing only the .class files in a file system hierarchy rooted in the current directory: `ajar -c -r myjar * -i *.class`
9. To create a jar named myjar without deflating the .java files: `ajar -c -r -n java — myjar *`
10. To create a jar named myjar while reading the file list from stdin: `ajar -@ -c -r myjar`

Sample stdin data:

```
docs
source
classes
-x
docs/foo/*
```

11. To add a file named bar to a jar named myjar: `ajar -a myjar bar`
12. To delete a file named foo/bar from a jar named myjar: `ajar -d myjar foo/bar`


Notes

1. Short options can be clustered (for example, `-c -v -D` is the same as `-cvD`). Long options (`—create`, `—verbose`, `—nodirs`, ..., and so on.) can be abbreviated as long as the abbreviations are unique.
2. File names can be changed when creating a jar or adding a file to a jar. For example, `"ajar -c x.jar bin/foo : bin/bar"` creates the jar file `x.jar` from the file `bin/foo` with a single entry, `bin/bar`. This can also be done using `stdin`, `"ajar -c@ x.jar"`, where `stdin` contains:
`bin/foo : bin/bar`
3. Use of **ajar** requires the `QIBM_MULTI_THREADED` environment variable must be set to 'Y'.

appletviewer - View Java™ applet

The `appletviewer` tool allows you to run applets without a web browser. It is compatible with the `appletviewer` tool that is supplied by Sun Microsystems, Inc.


The `appletviewer` tool is available using the Qshell Interpreter.

For more information about the `appletviewer` tool, see the `appletviewer` tool  by Sun Microsystems, Inc.

extcheck - A utility to detect JAR conflicts

In Java™ 2 SDK (J2SDK), Standard Edition, version 1.2, the `extcheck` tool detects version conflicts between a target JAR file and currently installed extension JAR files. It is compatible with the `keytool` that is supplied by Sun Microsystems, Inc.

The `extcheck` tool is available using the Qshell Interpreter.


For more information about the `extcheck` tool, see the `extcheck` tool  by Sun Microsystems, Inc.

jar - Archive Java™ files

The `jar` tool combines multiple files into a single Java™ ARchive (JAR) file. It is compatible with the `jar` tool that is supplied by Sun Microsystems, Inc.

The `jar` tool is available using the Qshell Interpreter.

For more information about file systems, see Integrated File System Information on IBM^(R) Toolbox for Java or "Files in the integrated file system."

For more information about the `jar` tool, see the `jar` tool  by Sun Microsystems, Inc.

Files in the integrated file system: The integrated file system stores Java™-related class files, source files, ZIP files, and JAR files in a hierarchical file structure. You can also store source files in the integrated file system. You may store the files in these integrated file systems:

- "root" (/) file system
- open systems file system (QOpenSys)
- user-defined file system
- library file system (QSYS.LIB)


- OS/2 Warp Server for iSeries^(TM) file system (QLANSrv)
- optical file system (QOPT)

Note: Other integrated file systems are not supported, because they are not thread safe.

jarsigner - JAR signing and verification

In Java[™] 2 SDK (J2SDK), Standard Edition, version 1.2, the jarsigner tool signs JAR files and verifies signatures on signed JAR files. The jarsigner tool accesses the keystore, which the keytool creates and manages, when it needs to find the private key for signing a JAR file. In J2SDK, the jarsigner and keytool tools replace the javakey tool. It is compatible with the jarsigner tool that is supplied by Sun Microsystems, Inc.

The jarsigner tool is available using the Qshell Interpreter.

For more information about the jarsigner tool, see the jarsigner tool  by Sun Microsystems, Inc.

java - Run Java[™] interpreter

The java Qshell command runs Java[™] programs. It is compatible with the java tool that is supplied by Sun Microsystems, Inc. with a few exceptions.

The IBM Developer Kit for Java ignores the following options of the java Qshell command:

Ignored option	Description
-cs	Not supported.
-checksource	Not supported.
-debug	Supported by the iSeries internal debugger.
-noasyncgc	Garbage collection is always running with the IBM Developer Kit for Java.
-noclassgc	Garbage collection is always running with the IBM Developer Kit for Java.
-prof	iSeries has its own performance tools.
-ss	Not applicable.
-oss	Not applicable.
-t	iSeries uses its own trace function.
-verify	Always verify on iSeries.
-verifyremote	Always verify on iSeries.
-noverify	Always verify on iSeries.

The java Qshell command supports new options. These are the new supported options:

Supported option	Description
-secure	Checks for public write access to directories in the CLASSPATH.
-gcfrq	Specifies the garbage collection frequency.
-gcpty	Specifies the garbage collection priority.
-opt	Specifies the optimization level.
-verbosegc	A message is displayed for each garbage collection sweep.

The Run Java (RUNJVA) command in the CL command reference documentation describes these new options in detail. The CL command reference documentation for the Create Java Program (CRTJVAPGM)

command, Delete Java Program (DLTJVAPGM) command, and Display Java Program (DSPJVAPGM) command contains information about managing Java programs.


The java Qshell command is available using the Qshell Interpreter.

For more information about the java Qshell command, see the java tool  by Sun Microsystems, Inc.

javac - Compile a Java™ program

The javac tool compiles Java™ programs. It is compatible with the javac tool that is supplied by Sun Microsystems, Inc.


The javac tool is available using the Qshell Interpreter.

For more information about the javac tool, see the javac tool  by Sun Microsystems, Inc.

javadoc - Generate Java™ documentation

The javadoc tool generates API documentation. It is compatible with the javadoc tool that is supplied by Sun Microsystems, Inc.

The javadoc tool is available using the Qshell Interpreter.

For more information about the javadoc tool, see the javadoc tool  by Sun Microsystems, Inc.

javah - Generate C header or stub file

The javah tool facilitates the implementation of Java™ native methods. It is compatible with the javah tool that is supplied by Sun Microsystems, Inc. with a few exceptions.

Note:

Writing native methods means that your application is not 100% pure Java. It also means that your application is not directly portable across platforms. Native methods are, by nature, platform or system specific. Using native methods may increase your development and maintenance costs for your applications.


The javah tool is available using the Qshell Interpreter. It reads a Java class file and creates a C-language header file in the current working directory. The header file that is written is a Stream File (STMF). It must be copied to a file member before it can be included in a C program on i5/OS.

The javah tool is compatible with the tool that is provided by Sun Microsystems, Inc. If the following options are specified; however, iSeries 400 ignores them:

Ignored option	Description
-td	The javah tool does not require a temporary directory.
-stubs	Java on iSeries only supports the Java Native Interface (JNI) form of native methods. Stubs were only required for the pre-JNI form of native methods.
-trace	Relates to the .c stub file output, which Java on iSeries does not support.
-v	Not supported.

Note:

The -jni option must always be specified. The iSeries server does not support native method implementations before JNI.

For more information about the javah tool, see the javah tool  by Sun Microsystems, Inc.

javakey - Manage Java™ security keys and certificates


Use the javakey tool for encryption key, and certificate generation and management, including generation of digital signatures for applets. It is compatible with the javakey tool that is supplied by Sun Microsystems, Inc.

Applet packaging and applet signing is dependent on your browser. Check your browser documentation to ensure that your browser is compatible with the Java™ JAR file format and javakey applet signing.

Note: The files that are created by the javakey tool contain sensitive information. Appropriate Integrated File System security measures protect the public and private key files.

The javakey tool is available using the Qshell Interpreter.

For more information about file systems, see the information about Integrated File System. or “Files in the integrated file system” on page 139.

For more information about the javakey tool, see the javakey tool  by Sun Microsystems, Inc.

javap - Disassemble a compiled Java™ program


The javap tool disassembles compiled Java™ files and prints out a representation of the Java program. This may be helpful when the original source code is no longer available on a system.

It is compatible with the javap tool that is supplied by Sun Microsystems, Inc. with a few exceptions:

Ignored option	Description
-b	This option is ignored. Backward compatibility is not required, because Java on iSeries only supports Java Developer Kit (JDK) 1.1.4 and later.
-p	On iSeries -p is not a valid option. You must spell out -private .
-verify	This option is ignored. The javap tool does not perform verification.

The javap tool is available using the Qshell Interpreter.


Note: The use of the javap tool to disassemble classes may violate the license agreement for those classes. Consult the license agreement for the classes before using the javap tool.

For more information about the javap tool, see the javap tool  by Sun Microsystems, Inc.

keytool - Key and certificate management tool

In Java™ 2 SDK (J2SDK), Standard Edition, version 1.2, the keytool creates public and private key pairs, self-signed certificates, and manages keystores. In J2SDK, the jarsigner and keytool tools replace the javakey tool. It is compatible with the keytool that is supplied by Sun Microsystems, Inc.


The `keytool` is available using the Qshell Interpreter.

For more information about the `keytool`, see the `keytool`  by Sun Microsystems, Inc.

native2ascii - Convert native characters to ASCII

The `native2ascii` tool converts a file with native-encoded characters (characters which are non-Latin 1 and non-Unicode) to one with Unicode-encoded characters. It is compatible with the `native2ascii` tool that is supplied by Sun Microsystems, Inc.


The `native2ascii` tool is available using the Qshell Interpreter.

For more information about the `native2ascii` tool, see the `native2ascii` tool  by Sun Microsystems, Inc.

policytool - Policy file creation and management tool

In Java™ 2 SDK, Standard Edition, version 1.2, the `policytool` creates and changes the external policy configuration files that define the Java security policy of your installation. It is compatible with the `policytool` that is supplied by Sun Microsystems, Inc.


The `policytool` is a graphical user interface (GUI) tool that is available using the Qshell Interpreter and the Native Abstract Windowing Toolkit. See IBM Developer Kit for Java Native Abstract Windowing Toolkit for more information.

For more information about the `policytool`, see the `policytool`  by Sun Microsystems, Inc.

rmic - Compile Java™ RMI stubs

The `rmic` tool generates stub files and class files for Java™ objects. It is compatible with the `rmic` tool that is supplied by Sun Microsystems, Inc.

The `rmic` tool is available using the Qshell Interpreter.

For more information about the `rmic` tool, see the `rmic` tool  by Sun Microsystems, Inc.

rmid - The Java™ RMI activation system

In Java™ 2 SDK (J2SDK), Standard Edition, version 1.2, the `rmid` tool starts the activation system daemon, so objects can be registered and activated in a Java virtual machine. It is compatible with the `rmid` tool that is supplied by Sun Microsystems, Inc.


The `rmid` tool is available using the Qshell Interpreter.

For more information about the `rmid` tool, see the `rmid` tool  by Sun Microsystems, Inc.

rmiregistry - Start a remote object registry

The `rmiregistry` tool starts a remote object registry on a specified port. It is compatible with the `rmiregistry` tool that is supplied by Sun Microsystems, Inc.


The `rmiregistry` tool is available using the Qshell Interpreter.

For more information about the `rmiregistry` tool, see the `rmiregistry` tool  by Sun Microsystems, Inc.

serialver - Return serial version

The `serialver` tool returns the version number or serialization-unique identifier for one or more classes. It is compatible with the `serialver` tool that is supplied by Sun Microsystems, Inc.


The `serialver` tool is available using the Qshell Interpreter.

For more information about the `serialver` tool, see the `serialver` tool  by Sun Microsystems, Inc.

tnameserv - Naming service

In Java™ 2 SDK (J2SDK), Standard Edition, version 1.2, the `tnameserv` tool provides access to the naming service. It is compatible with the `tnameserv` tool that is supplied by Sun Microsystems, Inc.

The `tnameserv` tool is available using the Qshell Interpreter.

For more information about the `tnameserv` tool, see the `tnameserv` tool  by Sun Microsystems, Inc.

Utilities for managing jobs

The following are Qshell utilities for managing jobs:

- “`getjobid` - Display job information”
- “`hash` - Remember or report utility locations” on page 145
- “`jobs` - Display status of jobs in current session” on page 146
- “`kill` - Terminate or signal processes” on page 147
- “`liblist` - Manage library list” on page 148
- “`ps` - Display process status” on page 149
- “`sleep` - Suspend invocation for an interval” on page 151
- “`trap` - Trap signals” on page 151
- “`wait` - Wait for process completion” on page 152

getjobid - Display job information

Synopsis

`getjobid [-csv] [pid ...]`

`getjobid -j [-csv] [qualified-job ...]`

Description

The `getjobid` utility writes the qualified job name and process ID for the specified process to standard output. The qualified job name is a string in the form *number/user/name*. The *number* is a six-digit decimal number, *user* is the user profile under which the job was started, and *name* is the name of job.

In the first synopsis form, the process is identified using the process ID. In the second synopsis form, the process is identified using the qualified job name.

When the `-v` option is specified, `getjobid` displays the process ID, parent’s process ID, process group, current status, and qualified job name for the specified process.

Note: This utility is unique to i5/OS™.

Options

- `-c` Display information about all of the current child processes of the specified process.
- `-j` Processes are identified using the qualified job name.
- `-s` Display a short form with just the qualified job name.

- v Display detailed information about the specified process, including the process ID, the parent's process ID, process group, current status, and qualified job name.

Operands

Each *pid* is the decimal process ID of an active process on the system. When *pid* is not specified, **getjobid** displays information for the current process.

Each *qualified-job* is the qualified job name of an active process on the system. The qualified job name is a string in the form *number/user/name*. The *number* is a six-digit decimal number, *user* is the user profile under which the job was started, and *name* is the name of job.

Exit Status

- 0 when successful.
- >0 when an error occurred. The exit status is the number of processes for which information could not be obtained.

Related information

- “jobs - Display status of jobs in current session” on page 146
- “ps - Display process status” on page 149

Examples

1. Display the qualified job name of the current process.
getjobid
2. Display detailed information for three processes.
getjobid -v 318 942 1130
3. Display the short form of the qualified job name for one process.
getjobid -s 325
4. Display detailed information for a process identified with the qualified job name.
getjobid -jv 325411/SHELLTST/QZSHCHLD

hash - Remember or report utility locations

hash [-p *filename*] [*utility* ...]

hash -r

Description

The **hash** utility adds *utility* to the list of remembered utility locations or removes all remembered utilities from the list. By default, **hash** uses a path search to find *utility*.

When no arguments are specified, **hash** reports the contents of the list. An entry that has not been looked at since the last “cd - Change working directory” on page 88 command is marked with an asterisk; it is possible for the entry to be invalid.

Options

- p *filename*
Do not use a path search to find *utility*. Use the specified *filename* as the location of *utility*.
- r
Remove all previously remembered utility locations.

Operands

Each *utility* is added to the list of remembered utility locations.

Exit Status

- 0 when successful.

Related information

- “cd - Change working directory” on page 88

jobs - Display status of jobs in current session

Synopsis

```
jobs [ -ln ] [ job ... ]
```

Description

You can use **jobs** to display information about active jobs started by **qsh**. For each job, **qsh** displays:

- Job number in brackets ([]).
- Status (Running, Done, Terminated, and so on).
- Return value of the job in parenthesis () when the return value is greater than zero and the job status is Done.
- Command line for the job.

Options

- l Display status for each process in the specified job.
- n Display status only for those jobs whose status has changed but has not been reported yet.

Operands

Each *job* specifies an active job. The job can be specified as a:

- Number to refer to a process id.
 - %number to refer to a job number.
 - %string to refer to a job whose name begins with string.
- If *job* is not specified, **qsh** displays status for all active jobs.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “kill - Terminate or signal processes” on page 147
- “wait - Wait for process completion” on page 152

Examples

1. Display status for job number 1: **jobs %1**
2. Display status for process id 16107: **jobs 16107**
3. Display status for a job running the ls utility: **jobs %ls**
4. Display status for all active jobs: **jobs**

kill - Terminate or signal processes

Synopsis

kill [**-s** *signame*] *job* ...

kill [**-n** *signum*] *job* ...

kill [**-sig**] *job* ...

kill -l [*signal* ...]

Description

You can use **kill** to send a signal to the specified *jobs*. You can specify a signal using:

- *signame* - A signal name.
- *signum* - A signal number.
- *sig* - Either a signal name or signal number with no space after the minus (-).

Note:

The valid signal numbers on i5/OS^(TM) may be different than the signal numbers on other systems. You can list the valid signal names by specifying the **-l** option. For portability, you should always specify the signal name.

Options

- l** List signal names. If there are no arguments, **qsh** displays all of the signal names. If *signal* is a name, **qsh** displays the corresponding signal number. If *signal* is a number, **qsh** displays the corresponding signal name.
- n** A signal number.
- s** A signal name in either uppercase or lowercase.

Operands

Each *job* specifies an active job. The job can be specified as a:

- Number to refer to a process id.
- %number to refer to a job number.
- %string to refer to a job whose name begins with string.

Exit Status

- 0 when successful.
- >0 when unsuccessful. If the **-l** option was not specified, the exit status is the number of jobs to which **qsh** could not send the signal.

Related information

- “jobs - Display status of jobs in current session” on page 146
- “trap - Trap signals” on page 151
- “wait - Wait for process completion” on page 152

Examples

1. Send the USR1 signal to process id 16711: **kill -s USR1 16711**
2. Send the USR1 signal to job 1: **kill -n 7 %1**

3. List the valid signal names: **kill -l**

liblist - Manage library list

Synopsis

liblist [**-acdf**] [*library ...*]

Description

You can use **liblist** to add or delete a library from the user portion of the library list, set the current library, or display the library list for the current job.

You can add libraries to the user portion of the library list by specifying the **-a** option and a list of libraries. By default, the libraries are added to user portion of the beginning of the library list.

You can remove libraries from the user portion of the library list by specifying the **-d** option and a list of libraries.

The current library is set to *library* when the **-c** option is specified. The current library can be unset by specifying both the **-c** and **-d** options.

When no arguments are specified, **qsh** displays the current library list. Each line in the output includes the library name and the type of the library. A library can be one of the following types:

- SYS for a library in the system portion of the library list.
- PRD for a library in the product portion of the library list.
- CUR for the current library.
- USR for a library in the user portion of the library list.

Options

- a** Add *library* to the user portion of the library list.
- c** Set the current library to *library*.
- d** Remove *library* from the user portion of the library list or unset the current library if the **-c** option is also specified.
- f** When the **-a** option is specified, add *library* to the beginning of the user portion of the library list.
- l** When the **-a** option is specified, add *library* to the end of the user portion of the library list.

Operands

Each *library* is a library to either add or delete from the user portion of the library list depending on the options specified.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Examples

1. Add the library MYLIB to the library list: **liblist -a MYLIB**
2. Remove the library MYLIB from the library list: **liblist -d MYLIB**
3. Set the current library to MYLIB: **liblist -c MYLIB**
4. Unset the current library: **liblist -cd**
5. Display the library list: **liblist**

ps - Display process status

Synopsis

ps [-Aaefjlt] [-o *format*] [-p *pidlist*] [-s *sbslist*] [-u *userlist*]

Description

The **ps** utility displays information about processes. The output from **ps** can include the following fields:

CGROUP

The current primary group profile of the process.

CMD Program, menu, or command most recently run by the process.

CUSER

The current user profile of the process.

DEVICE

Name of the device description object that is associated with the process.

ETIME

The elapsed time since the process started. The time is displayed in the format `[[dd-]hh:]mm:ss` where *dd* is the number of days, *hh* is the number of hours, *mm* is the number of minutes, and *ss* is the number of seconds.

FUNCTION

Program, menu, or command most recently run by the process.

JOBID Qualified job name of the process. The qualified job name is a string in the form *number/user/name*. The *number* is a six-digit decimal number, *user* is the user profile under which the job was started, and *name* is the name of job.

JOBNAME

Job name component of the qualified job name.

JOBNUM

Job number component of the qualified job name.

NTHREADS

The number of threads currently running in the process as a decimal number.

PCPU The ratio of CPU time used recently to CPU time available, expressed as a percentage.

PGID Process group ID number as a decimal number.

PID Process ID number as a decimal number.

PPID Parent process ID number as a decimal number.

PRI Current priority of the process as a decimal number. Lower numbers mean a higher priority.

SBS Subsystem in which the process is running.

STATUS

Current status of the process.

STIME

Date and time the process was started. By default, the date and time is displayed in the format *mm-dd-yyyy hh:nn:ss* where *mm* is the month, *dd* is the day, *yyyy* is the year, *hh* is the hour, *nn* is the minute, and *ss* is the second. If the `LC_TIME` environment variable is set, the date and time is displayed with the format specified by the `d_t_fmt` keyword in the `LC_TIME` category of the specified locale.

THCOUNT

The number of threads currently running in the process as a decimal number.

TIME CPU time used by the process in seconds. The time is displayed in the format `[[dd-]hh:]mm:ss` where *dd* is the number of days, *hh* is the number of hours, *mm* is the number of minutes, and *ss* is the number of seconds.

TMPSZ

The amount of temporary storage used by the process in megabytes as a decimal number.

TYPE The type of the process.

USER User profile component of the qualified job name.

UID User id number corresponding to the user profile component of the qualified job name.

By default, **ps** displays the PID, DEVICE, TIME, FUNCTION, STATUS, and JOBID fields about processes owned by the current user. Use the **-o** option to select the fields displayed by **ps**.

To display information about other processes, you must have *JOBCTL special authority.

Options

- a** Display information for all processes associated with a 5250 terminal.
- A** Display information for all processes. This includes processes that are active, on a job queue, or on an output queue.
- e** Include active processes in the output.
- f** Display a full listing. The output includes the USER, PID, PPID, STIME, DEVICE, TIME and FUNCTION fields.
- j** Include processes on a job queue in the output.
- l** Display a long listing. The output includes the USER, PID, PPID, PRI, STATUS, JOBID, STIME, DEVICE, TIME and FUNCTION fields.
- o *format***
Display information according to the format specification given in *format*. Multiple **-o** options can be specified.
- p *pidlist***
Write information for processes whose process ID numbers are specified in *pidlist*. The *pidlist* must be a single argument in the form of a blank- or comma-separated list.
- s *sbslist***
Write information for processes running in the subsystems specified in *sbslist*. The *sbslist* must be a single argument in the form of a blank- or comma-separated list.
- t** Include processes on an out queue in the output.
- u *userlist***
Write information for processes whose user ID numbers or user names are specified in *userlist*. The *userlist* must be a single argument in the form of a blank- or comma-separated list.

Environment Variables

ps is affected by the following environment variables:

LANG

Provides a default value for locale categories that are not specifically set with a variable starting with LC_.

LC_TIME

Defines the output format for date and time attributes.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “getjobid - Display job information” on page 144
- “jobs - Display status of jobs in current session” on page 146

sleep - Suspend invocation for an interval

Synopsis

sleep *time*

Description

You can use **sleep** to suspend a process from running for *time* seconds.

Options

None.

Operands

The value of *time* must be a positive integer.

Exit Status

- 0 when successful.
- >0 when *time* is invalid.

trap - Trap signals

Synopsis

trap [*action condition ...*]

trap -p [*condition ...*]

trap -l

Description

The **trap** utility sets the *action* for **qsh** to take when a *condition* arises. **qsh** expands *action* once when running **trap** and again when *condition* arises.

When the **-p** option is specified, **trap** displays the current *action* for the specified *conditions*.

When the **-l** option is specified, **trap** displays a list of all of the signal names and their corresponding numbers.

When no arguments are specified, **trap** displays a list of the currently defined traps.

Options

- l** Display a list of all of the signal names and their corresponding numbers.
- p** Display each trap in a re-enterable format.

Operands

For *action*, you can specify:

- Null to ignore *condition* when it arises
- Minus (-) to reset *condition* to its original value.
- A command to be run each time *condition* arises.

For *condition*, you can specify:

- Name or number of a signal. You can use **trap -l** to display a list of valid signals. For portability, you should always specify the signal name.
- **0** or **EXIT**. **qsh** runs *action* when the shell exits.
- **ERR**. **qsh** runs *action* when a command has a non-zero exit status.
- **DEBUG**. **qsh** runs *action* after each simple command.

If more than one condition arises at the same time, **qsh** runs the traps in this order:

1. **DEBUG**, if it is specified, then
2. **ERR**, if it is specified and applicable, then
3. Any other specified traps in signal number order, then
4. **EXIT**.

Exit Status

- 0 when successful.
- >0 when an invalid *condition* is specified.

Related information

- “kill - Terminate or signal processes” on page 147
- “wait - Wait for process completion”

Examples

1. Set a trap for the ERR condition:

```
trap `print Command failed` ERR
```
2. Ignore the ERR condition:

```
trap "" ERR
```
3. Reset the ERR condition to its original value:

```
trap - ERR
```
4. Display the current action for the ERR condition:

```
trap -p ERR
```
5. Display all of the currently defined traps:

```
trap
```

wait - Wait for process completion

Synopsis

```
wait [ job ... ]
```

Description

You can use **wait** to wait for the specified *jobs* to end. If *job* is not specified, **qsh** waits for all child processes to end.

Options

None.

Operands

Each *job* specifies an active job. The job can be specified as a:

- Number to refer to a process id. **qsh** waits for the given process to end.
- %number to refer to a job number. **qsh** waits for all processes in the job to end.
- %string to refer to a job whose name begins with string. **qsh** waits for all processes in the job to end.

Exit Status

When no *job* was specified, the exit status is:

- 0 when all running jobs have ended.
- >0 when unsuccessful.

When at least one *job* was specified, the exit status is the exit status of the last *job*.

Related information

- “jobs - Display status of jobs in current session” on page 146
- “kill - Terminate or signal processes” on page 147
- “trap - Trap signals” on page 151

Examples

1. Wait for process id 16825 to end: **wait** 16825
2. Wait for job number 5 to end: **wait** %5

Utilities for Kerberos credentials and key tables

The following are Qshell utilities for working with Kerberos credentials and key tables:

- **kdestroy** - Destroy a Kerberos credentials cache
- **keytab** - Manage a Kerberos key table file
- **kinit** - Obtain or renew a Kerberos ticket-granting ticket
- **klist** - Display the contents of a Kerberos credentials cache or key table file
- **ksetup** - Manage Kerberos service entries in the LDAP directory for a Kerberos realm

Utilities for LDAP directory server

The following are Qshell utilities for working with the LDAP directory server:

- **ldapadd** - Add LDAP entry tool
- **ldapmodify** - Change LDAP entry tool
- **ldapchangepwd** - Change LDAP password tool
- **ldapmodrdn** - Change LDAP Relative Distinguished Name (RDN^(TM)) tool
- **ldapdiff** - Compare LDAP replication synchronization tool
- **ldapdelete** - Delete LDAP entry tool
- **ldapexop** - Extend LDAP operation tool
- **ldapsearch** - Search LDAP server tool

Utilities for working with parameters and variables

The following are Qshell utilities for working with parameters and variables:

- “declare - Declare variables and set attributes”
- “export - Set export attribute for variables” on page 155
- “local - Assign a local variable in a function” on page 156
- “printenv - Display values of environment variables” on page 157
- “readonly - Set read-only attribute for variables” on page 157
- “set - Set or unset options and positional parameters” on page 158
- “shift - Shift positional parameters” on page 160
- “typeset - Declare variables and set attributes” on page 160
- “unset - Unset values of variables and functions” on page 161

declare - Declare variables and set attributes

Synopsis

```
declare [ -Eilrux ] name [=value] ...
```

```
declare [ +Eilrux ] name [=value] ...
```

```
declare -fF [ name ... ]
```

```
declare -p name ...
```

```
declare
```

Description

The **declare** utility declares variables, assigns values to variables, sets or unsets attributes for variables, and displays the definitions for shell functions. If used in a shell function, **declare** makes the variable *name* local to the function.

In the first synopsis form, **declare** declares a variable *name* and optionally assigns it the specified *value*. If an option is specified, the corresponding attribute is turned on for the variable.

In the second synopsis form, **declare** declares a variable *name* and optionally assigns it the specified *value*. If an option is specified, the corresponding attribute is turned off for the variable.

In the third synopsis form, **declare** displays the names and definitions for all shell functions if no *names* are specified or the shell functions specified by *name*.

In the fourth synopsis form, **declare** displays the attributes and value of the variables specified by *name* in a re-enterable format.

In the fifth synopsis form, **declare** displays the names and values of all variables.

Options

- E Set the floating point attribute for the variable. On assignments to the variable the value is evaluated as a floating point number.
- f Display the names and definitions of shell functions.
- F Display the names of shell functions.

- i Set the integer attribute for the variable. On assignments to the variable the value is evaluated as an integer number.
- l Set the lowercase attribute for the variable. On assignments to the variable the value is set to lowercase characters.
- p Display each variable in a re-enterable format.
- r Set the read-only attribute for the variable. The variable cannot have its value changed by a subsequent assignment and cannot be unset. If a *value* is also specified, the value of the variable is updated before setting the read-only attribute.
- u Set the uppercase attribute for the variable. On assignments to the variable the value is set to uppercase characters.
- x Set the export attribute for the variable. The variable is automatically placed in the environment of subsequently executed commands.

Operands

Each *name* must be a valid “Variables” on page 10 name.

Exit Status

- 0 when successful
- >0 when unsuccessful

Related information

- “export - Set export attribute for variables”
- “let - Evaluate arithmetic expression” on page 164
- “local - Assign a local variable in a function” on page 156
- “readonly - Set read-only attribute for variables” on page 157
- “set - Set or unset options and positional parameters” on page 158
- “unset - Unset values of variables and functions” on page 161

export - Set export attribute for variables

Synopsis

```
export [ -ps ] [ name [ =value ] ... ]
```

Description

You can use **export** to set the export attribute for the variables specified by *name*. A variable with the export attribute is automatically placed in the environment of subsequently executed commands.

When no arguments are specified, **qsh** displays a list of all the variables with the export attribute and their values.

Options

- p Precede each line of the output with the word “export ” so it is displayed in a re-enterable format.
- s Also set the variable as an environment variable in the current process.

Operands

Each *name* specifies a variable in the current environment. If a *value* is also specified, the value of the variable is updated.

Exit Status

- 0 when successful.

Related information

- “declare - Declare variables and set attributes” on page 154
- “local - Assign a local variable in a function”
- “readonly - Set read-only attribute for variables” on page 157
- “set - Set or unset options and positional parameters” on page 158
- “unset - Unset values of variables and functions” on page 161

Examples

1. Set the export attribute for an existing variable:

```
export ALPHA
```
2. Set the value and export attribute of a new variable:

```
export ALPHA=one
```
3. List all variables with the export attribute:

```
export
```

local - Assign a local variable in a function

Synopsis

```
local [ name [ =value ] ... ]
```

Description

You can use **local** to make a variable local to a function. When a variable is made local, it inherits the initial value and exported and read-only attributes from the variable with the same name in the surrounding scope, if there is one. Otherwise, the variable is initially unset.

qsh uses dynamic scoping, so that if you make the variable *alpha* local to function *foo*, which then calls function *bar*, references to the variable *alpha* made inside *bar* will refer to the variable declared inside *foo*, not to the global variable named *alpha*.

The special parameter `-` is the only special parameter that can be made local. By making `-` local, any shell options that are changed with **set** inside the function are restored to their original values when the function returns.

Options

None.

Operands

Each *name* specifies a variable in the current environment. If a *value* is also specified, the value of the variable is updated.

Exit Status

- 0 when successful.
- >0 when called from outside of a function.

Related information

- “export - Set export attribute for variables” on page 155

- “readonly - Set read-only attribute for variables”
- “set - Set or unset options and positional parameters” on page 158
- “unset - Unset values of variables and functions” on page 161

printenv - Display values of environment variables

Synopsis

```
printenv [ -s ] [ name ]
```

Description

The **printenv** utility displays the value of the environment variable *name*. If no *name* is specified, **printenv** displays all of the current environment variables, one per line, in the format “name=value”. By default, **printenv** displays job environment variables.

Options

-s Display system environment variables.

Operands

The *name* is the name of an environment variable in the current environment or a system environment variable.

Exit Status

- 0 when successful
- >0 if *name* is not currently defined

Related information

- “env - Set environment for command invocation” on page 52
- “export - Set export attribute for variables” on page 155

readonly - Set read-only attribute for variables

Synopsis

```
readonly [ -p ] [ name [ =value ] ... ]
```

Description

You can use **readonly** to set the read-only attribute for the variables specified by *name*. A variable with the read-only attribute cannot have its value changed by a subsequent assignment and cannot be unset.

Note that **qsh** can change the value of a variable with the read-only attribute. For example, if **PWD** has the read-only attribute, it’s value will be changed when you change the current working directory.

When no arguments are specified, **qsh** displays a list of the variables with the read-only attribute and their values.

Options

-p Precede each line of the output with the word “readonly ” so it is displayed in a re-enterable format.

Operands

Each *name* specifies a variable in the current environment. If a *value* is also specified, the value of the variable is updated before setting the read-only attribute.

Exit Status

- 0 when successful.
- >0 when unsuccessful.

Related information

- “declare - Declare variables and set attributes” on page 154
- “export - Set export attribute for variables” on page 155
- “local - Assign a local variable in a function” on page 156
- “set - Set or unset options and positional parameters”
- “unset - Unset values of variables and functions” on page 161

Examples

1. Set the read-only attribute for an existing variable:
`readonly ALPHA`
2. Set the value and read-only attribute of a new variable:
`readonly ALPHA=one`
3. List all variables with the read-only attribute:
`readonly`

set - Set or unset options and positional parameters

Synopsis

```
set [ -abCefJlmntuvx- ] [ -o option ] [ argument ... ]
```

```
set [ +abCefJlmntuvx- ] [ +o option ] [ argument ... ]
```

Description

The `set` utility can:

- Display the names and values of all shell variables by specifying no options or arguments.
- Display the option settings by specifying the `-o option` but no *option*.
- Set an option by specifying a - (minus) followed by the option letter or by specifying `-o option`.
- Unset an option by specifying a + (plus) followed by the option letter or by specifying `+o option`.
- Set positional parameters by specifying *arguments*.
- Unset positional parameters by specifying `—` but no *argument*.

Options

All of the single letter options have a corresponding `-o option`. The option value is listed in parenthesis following the letter option below. `qsh` supports the following options:

-a (allexport)

Set the export attribute to each variable that is assigned a value.

-b (notify)

Enable asynchronous notification of background job completion.

-C (noclobber)

Do not overwrite existing files with the `>` redirection operator.

- e (errexit)**
If the interactive option is not set, exit immediately if any untested command fails. The exit status of a command is considered to be explicitly tested if the command is used to control an **if**, **elif**, **while**, or **until**; or if the command is the left hand operand of an **&&** or **||** operator.
- f (noglob)**
Disable path name expansion.
- F (float)**
Enable floating point arithmetic in “Arithmetic expansions” on page 18
- j (jobtrace)**
Enable job tracing. Each time **qsh** starts a i5/OS^(TM) job, it displays a message to standard error with the fully-qualified job name and process id.
- l (logcmds)**
Enable command logging. Write each command to a message in the job log before it is run.
- m (monitor)**
Display a message when a job completes. **qsh** implicitly turns on this option when the interactive option is set.
- n (noexec)**
If the interactive option is not set, read commands but do not run them. This is useful for checking the syntax of shell scripts.
- t (trace)**
Enable internal tracing. **qsh** traces internal information to the file specified by **TRACEFILE** variable or the **qsh_trace** file in the user’s home directory.
- u (nounset)**
Write a message to standard error when attempting to expand a variable that is not set, and if the interactive option is not set exit immediately.
- v (verbose)**
Write input to standard error as it is read.
- x (xtrace)**
Write each command to standard error before it is run, preceded by the expansion of the **PS4** variable.

Operands

Each *argument* is assigned in order to the positional parameters.

Exit Status

- 0 when successful.

Related information

- “export - Set export attribute for variables” on page 155
- “qsh - Qshell command language interpreter” on page 56
- “readonly - Set read-only attribute for variables” on page 157
- “shift - Shift positional parameters” on page 160
- “unset - Unset values of variables and functions” on page 161

Examples

1. List all variables and their values:
set
2. List all option settings:

- ```
set -o
```
3. Set positional parameters \$1, \$2, \$3:  
set alpha beta gamma
  4. Set the allexport and notify options:  
set -o allexport -o notify
  5. Set the verbose and xtrace options:  
set -xv
  6. Unset the xtrace option:  
set +x
  7. Unset the notify option:  
set +o notify
  8. Unset all positional parameters:  
set --

## shift - Shift positional parameters

### Synopsis

```
shift [n]
```

### Description

You can use **shift** to shift the positional parameters to the left by *n*. Positional parameter 1 is assigned the value of positional parameter (1+*n*), positional parameter 2 is assigned the value of positional parameter (2+*n*), and so forth. The special parameter # is updated with the new number of positional parameters.

### Options

None.

### Operands

The value of *n* must be an unsigned integer less than or equal to the special parameter #. If *n* is not specified, the default value is 1. If *n* is 0, there are no changes to the positional parameters.

### Exit Status

- 0 when successful.
- >0 when *n* is invalid.

### Related information

- “set - Set or unset options and positional parameters” on page 158

### Examples

1. Shift the positional parameters by two: **shift 2**

## typeset - Declare variables and set attributes

### Synopsis

```
typeset [-Eilrux] name [=value] ...
```

```
typeset [+Eilrux] name [=value] ...
```

```
typeset -fF [name ...]
```



**typeset -p** *name* ...

**typeset**

### Description

The **typeset** utility declares variables, assigns values to variables, sets attributes for variables, and displays the definitions for shell functions. It is a synonym for the “declare - Declare variables and set attributes” on page 154 utility.

### Related information

- “declare - Declare variables and set attributes” on page 154
- “export - Set export attribute for variables” on page 155
- “local - Assign a local variable in a function” on page 156
- “readonly - Set read-only attribute for variables” on page 157
- “set - Set or unset options and positional parameters” on page 158
- “unset - Unset values of variables and functions”

## unset - Unset values of variables and functions

### Synopsis

**unset** [ **-fv** ] [ *name* ... ]

### Description

You can use **unset** to unset each variable or function specified by name. If no option is specified, *name* refers to a variable. Variables with the read-only attribute cannot be unset.

### Options

- f *name* refers to a function.
- v *name* refers to a variable.

### Operands

Each *name* is a variable or function.

### Exit Status

- 0 when successful.
- >0 when at least one *name* could not be found. The value is the number of *names* that are not found.

### Related information

- “export - Set export attribute for variables” on page 155
- “local - Assign a local variable in a function” on page 156
- “readonly - Set read-only attribute for variables” on page 157
- “set - Set or unset options and positional parameters” on page 158

### Examples

1. Unset the variable alpha: `unset alpha`
2. Unset the function foo: `unset -f foo`

## Utilities for writing scripts

The following are Qshell utilities for writing scripts:

- “break - Exit from for, while, or until loop”
- “colon (:) - Null utility”
- “continue - Continue for, while, or until loop” on page 163
- “false - Return false value” on page 163
- “getopts - Parse utility options” on page 163
- “let - Evaluate arithmetic expression” on page 164
- “return - Return from a function” on page 164
- “test - Evaluate expression” on page 165
- “true - Return true value” on page 167

### break - Exit from for, while, or until loop

#### Synopsis

**break**[ *n* ]

#### Description

You can use **break** to exit from the smallest enclosing **for**, **while**, or **until** loop or from the *n*th enclosing loop. Processing resumes with the command immediately following the loop.

#### Options

None.

#### Operands

The value of *n* must be greater than or equal to 1.

#### Exit Status

- 0 when successful.

#### Related information

- “continue - Continue for, while, or until loop” on page 163

### colon (:) - Null utility

#### Synopsis

: [ *argument ...* ]

#### Description

You can use **colon** where you must have a command, but you do not want the command to do anything. For example, in the **then** condition of an **if** command.

#### Options

None.

#### Operands

Each *argument* is expanded.

### Exit Status

- 0 when successful.

## **continue - Continue for, while, or until loop**

### Synopsis

`continue [ n ]`

### Description

You can use **continue** to go to the top of the smallest enclosing **for**, **while**, or **until** loop or to the *n*th enclosing loop. Processing resumes with the first command at the top of the loop.

### Options

None.

### Operands

The value of *n* must be greater than or equal to 1.

### Exit Status

- 0 when successful.

### Related information

- “break - Exit from for, while, or until loop” on page 162

## **false - Return false value**

### Synopsis

`false`

### Description

`false` returns with an exit code that is non-zero.

### Options

None.

### Operands

None.

### Exit Status

- >0 when successful.

### Related information

- “true - Return true value” on page 167

## **getopts - Parse utility options**

### Synopsis

`getopts optstring varname`

## Description

You can use **getopts** to check the positional parameters for legal options. An option argument begins with a minus (-). The end of the the options is marked by the first argument that does not begin with a minus or an argument of `—`.

Each time you call **getopts**, it places the next option letter it finds in *varname*. **qsh** stores the index of the next parameter to be processed in the variable **OPTIND**. When an option requires an argument, **qsh** stores the argument in the variable **OPTARG**.

## Options

None.

## Operands

The option letters recognized by **getopts** are identified in *optstring*. If a letter is followed by a colon (:), that option is expected to have an argument. The argument can be separated from the option letter by `<space>s`.

With each call to **getopts**, *varname* is updated with the option letter.

## Exit Status

- 0 when successful.
- >0 when unsuccessful.

## let - Evaluate arithmetic expression

### Synopsis

```
let arg ...
```

## Description

You can use **let** to evaluate each *arg* as an “Arithmetic expansions” on page 18. You may need to quote each *arg* since many arithmetic operators have a special meaning to **qsh**.

## Operands

Each *arg* is evaluated as an “Arithmetic expansions” on page 18.

## Exit Status

- 0 when the value of the last expression is non-zero
- 1 when the value of the last expression is zero

## Examples

1. Add one to the variable x.

```
let x=x+1
```

## return - Return from a function

### Synopsis

```
return [n]
```

## Description

You can use **return** to cause a function or dot script to return to the invoking shell script. If **return** is called outside a function or dot script, it is equivalent to **exit**.

### Options

None.

### Operands

The value of *n* is an integer that is greater than or equal to 0 and less than or equal to 255.

### Exit Status

*n* if specified. Otherwise, the exit status of the preceding command.

### Related information

- “exit - Exit from the shell” on page 54

## test - Evaluate expression

### Synopsis

*test expression*

[ *expression* ]

### Description

The **test** utility checks the type of a file, checks permissions on files, compares two strings, or compares two arithmetic expressions.

The **test** utility tests conditions for files using the following primaries:

- b** *file* True if *file* exists and is a block special file.
- c** *file* True if *file* exists and is a character special file.
- d** *file* True if *file* exists and is a directory.
- e** *file* True if *file* exists regardless of type.
- f** *file* True if *file* exists and is a regular file.
- g** *file* True if *file* exists and its set group id flag is set.
- G** *file* True if *file* exists and is owned by the effective group id.
- h** *file* True if *file* exists and is a symbolic link.
- k** *file* True if *file* exists and its restricted deletion flag is set.
- L** *file* True if *file* exists and is a symbolic link.
- N** *file* True if *file* exists and is a native object.
- O** *file* True if *file* exists and is owned by the effective user id.
- p** *file* True if *file* exists and is a pipe.
- r** *file* True if *file* exists and is readable.
- s** *file* True if *file* exists and has a size greater than zero.
- S** *file* True if *file* exists and is a socket.

- u file** True if *file* exists and its set user id flag is set.
- w file** True if *file* exists and is writable.
- x file** True if *file* exists and is executable. This only means that the execute bit is on. If *file* is a directory, the directory can be searched.

**file1 -ef file2**  
True if *file1* and *file2* are different names for the same file (they have the same device and inode numbers).

**file1 -nt file2**  
True if *file1* is newer than *file2* or *file2* does not exist.

**file1 -ot file2**  
True if *file1* is older than *file2* or *file2* does not exist.

The **test** utility tests conditions for checking status using the following primaries:

- o optname** True if shell option *optname* is enabled.
- t fd** True if file descriptor *fd* is open and associated with a terminal.

The **test** utility tests conditions for comparing strings using the following primaries:

- n string** True if the length of *string* is non-zero.
- z string** True if the length of *string* is zero.
- string** True if *string* is not the null string.
- string1 = string2** True if the strings are identical.
- string1 == string2** True if the strings are identical.
- string1 != string2** True if the strings are not identical.
- string1 < string2** True if *string1* sorts before *string2* in the collation sequence of the current locale.
- string1 > string2** True if *string1* sorts after *string2* in the collation sequence of the current locale.

The **test** utility tests conditions for comparing “Arithmetic expansions” on page 18 using the following primaries:

- exp1 -eq exp2** True if the arithmetic expressions are equal.
- exp1 -ne exp2** True if the arithmetic expressions are not equal.
- exp1 -gt exp2** True if the first arithmetic expression is greater than the second arithmetic expression.
- exp1 -ge exp2** True if the first arithmetic expression is greater than or equal to the second arithmetic expression.
- exp1 -lt exp2** True if the first arithmetic expression is less than the second arithmetic expression.

*exp1 -le exp2*

True if the first arithmetic expression is less than or equal to the second arithmetic expression.

The above primaries can be combined to form complex expressions using the following operators:

- *! expr* True if *expr* is false.
- *expr1 -a expr2* True if both expressions are true.
- *expr1 & expr2* True if both expressions are true.
- *expr1 && expr2* True if both expressions are true.
- *expr1 -o expr2* True if either expression is true.
- *expr1 | expr2* True if either expression is true.
- *expr1 || expr2* True if either expression is true.
- (*expr*) Parentheses are for grouping.

The **-a**, **&**, and **&&** operators have higher precedence than the **-o**, **|** operators, and **||** operators.

### Options

See above.

### Operands

All operators and flags are separate arguments.

### Exit Status

- 0 when *expression* is true.
- 1 when *expression* is false.
- >1 when there is an error.

### Examples

1. See if /home is a directory:  

```
test -d /home
```
2. See if one integer is less than or equal to another:  

```
test "$index" -le "$count"
```
3. See if two strings are equal:  

```
test "$REPLY" = "Yes"
```

## true - Return true value

### Synopsis

**true**

### Description

**true** returns with an exit code of zero.

### Options

None.

### Operands

None.

## Exit Status

Zero.

## Related information

- “false - Return false value” on page 163

## Miscellaneous utilities

The following are miscellaneous Qshell utilities:

- “clrtmp - Clear the /tmp directory”
- “dataq - Send or receive messages from i5/OS<sup>(TM)</sup> data queue” on page 169
- “datarea - Read or write i5/OS<sup>(TM)</sup> data area” on page 170
- “date - Write the date and time” on page 170
- “expr - Evaluate arguments as an expression” on page 172
- “hostname - Display the name of the current host system” on page 173
- “id - Return user identity” on page 173
- “ipcrm - Remove interprocess communication identifier” on page 174
- “ipcs - Report interprocess communication status” on page 175
- “locale - Get locale specific information” on page 180
- “logger - Log messages” on page 181
- “logname - Display user’s login name” on page 182
- “sysval - Retrieve system values or network attributes” on page 182
- “tee - Duplicate standard input” on page 182
- “ulimit - Set or display resource limits” on page 183
- “uname - Return system name” on page 184

## clrtmp - Clear the /tmp directory

### Synopsis

**clrtmp** [-c]

### Description

The **clrtmp** utility clears the /tmp directory by removing all of the objects from it. On other systems, the /tmp directory is cleared each time the system is started. On i5/OS<sup>(TM)</sup>, the /tmp directory is not cleared when the system is started. You can include a call to the **clrtmp** utility from the startup program specified by the QSTRUPPGM system value to have the /tmp directory cleared when i5/OS<sup>(TM)</sup> is started.

To remove objects from the /tmp directory the caller of **clrtmp** must have \*WX authority to each subdirectory contained in /tmp and \*OBJEXIST authority to each object. If the caller does not have the required authority those objects are not removed from the /tmp directory.

Unpredictable results may occur if **clrtmp** is called while the system is running. For example, if another program is writing to a file in the /tmp directory, the path to the file is removed and you will not be able use the file.

**Note:** This utility is unique to i5/OS<sup>(TM)</sup>.

### Options

**-c** Create /tmp if it does not exist.



### Exit Status

- 0 when successful
- >0 when an error occurs or at least one object could not be removed from the /tmp directory

## **dataq - Send or receive messages from i5/OS<sup>(TM)</sup> data queue**

### Synopsis

**dataq -c** [-l] *queue*

**dataq -r** [-lp] [-n *number*] [-t *seconds*] *queue*

**dataq -w** [-l] [-n *number*] *queue* [ *data ...* ]

### Description

The **dataq** utility clears messages from a data queue, reads messages from a data queue, or writes messages to a data queue.

In the first synopsis form, **dataq** clears all of the messages from the *queue*.

In the second synopsis form, **dataq** reads messages from the *queue* and writes them to standard output. By default, it reads one message from the *queue*. If no messages are available from the *queue*, **dataq** waits for a message.

In the third synopsis form, **dataq** writes messages to the *queue*. If *data* is specified, it is written as one message to the *queue*. Otherwise, each line read from standard input is written as a message to the *queue*.

### Options

**-c** Clear all of the messages from the *queue*.

**-l** When a relative path name is specified, use the library list to find the *queue*.

**-n** *number*

If the **-r** option is specified, read *number* messages from the *queue*. If the **-w** option is specified, write *number* messages to the *queue*.

**-p** Peek mode. When reading messages, the messages are left on the *queue*.

**-r** Read messages from the *queue*.

**-t** *seconds*

When reading messages, exit if no messages have been received after *seconds* seconds of waiting.

**-w** Write messages from the *queue*.

### Operands

The *queue* is the path name to a data queue. A data queue can only exist in the QSYS.LIB file system.

### Exit Status

- 0 when successful
- >0 when unsuccessful

### Related information

- "dataarea - Read or write i5/OS<sup>(TM)</sup> data area" on page 170
- "Rfile - Read or write record files" on page 122

## **dataarea - Read or write i5/OS<sup>™</sup> data area**

### **Synopsis**

**dataarea -r** [-l] [-s *substring*] *data-area*

**dataarea -w** [-l] [-s *substring*] *data-area* [ *data ...* ]

### **Description**

The **dataarea** utility reads or writes a data area.

In the first synopsis form, **dataarea** reads the contents of the *data-area* and writes it to standard output. By default, it reads the entire data area.

In the second synopsis form, **dataarea** writes to the *data-area*. If *data* is specified, it is written to the *data-area*. Otherwise, one line is read from standard input and written to the *data-area*.

### **Options**

**-l** When a relative path name is specified, use the library list to find the *data-area*.

**-r** Read from the *data-area*.

**-s** *substring*

For a character type data area, read or write the character positions specified by *substring*. The *substring* is specified as a number range that consists of a number, a dash (-), and a second number to select the character positions from the first number to the second number, inclusive. If the first number is omitted, character positions from 1 to the second number are selected. If the second number is omitted, character positions from the first number to the end of the data area are selected.

**-w** Write to the *data-area*.

### **Operands**

The *data-area* is the path name to a data area. A data area can only exist in the QSYS.LIB file system.

### **Exit Status**

- 0 when successful
- >0 when unsuccessful

### **Related information**

- “dataq - Send or receive messages from i5/OS<sup>™</sup> data queue” on page 169
- “Rfile - Read or write record files” on page 122

## **date - Write the date and time**

### **Synopsis**

**date** [-u] [+*format*]

### **Description**

The **date** utility writes the date and time to standard output. By default, the current date and time are written.

### **Options**

- u Give time in universal coordinated time (UTC). The QUTCOFFSET system value must be set correctly for **date** to return the correct time.

## Operands

The *+format* operand specifies the format of the output from the **date** command. Each field descriptor is replaced in the standard output by its corresponding value. All other characters are copied to the output without change. The output is always terminated with a newline character.

You can use these field descriptors:

- %a Insert abbreviated weekday name from locale.
- %A Insert full weekday name from locale.
- %b Insert abbreviated month name from locale.
- %B Insert full month name from locale.
- %c Insert date and time from locale.
- %d Insert day of the month (01-31).
- %H Insert hour (24-hour clock) as a decimal number (00-23).
- %I Insert hour (12-hour clock) as a decimal number (01-12).
- %j Insert day of the year (001-366).
- %m Insert month (01-12).
- %M Insert minute (00-59).
- %p Insert equivalent of either AM or PM from locale.
- %S Insert second (00-61).
- %U Insert week number of the year (00-53) where Sunday is the first day of the week.
- %w Insert weekday (0-6) where Sunday is 0. first day of the week.
- %W Insert week number of the year (00-53) where Monday is the first day of the week
- %x Insert date representation from locale.
- %X Insert time representation from locale.
- %y Insert year without the century (00-99).
- %Y Insert year.
- %Z Insert name of time zone, or no characters if time zone is not available.
- %% Insert %.

## Exit Status

- 0 when successful
- >0 when an error occurred

## Examples

1. Print the full weekday name, the full month name, the day and the full year.  
date +%@(#) 89 1.41@(#), 0 %d%, %Y  
Friday, August 14, 1998
2. Print the day, the abbreviated month name, and the abbreviated year.  
date +%d%.%b%.%y  
14.Aug.98

3. Print the numeric month, day, and abbreviated year.

```
date +%m%/%d%/%y
08/14/98
```

## expr - Evaluate arguments as an expression

### Synopsis

**expr** *operand* ...

### Description

The **expr** utility evaluates an expression formed by the *operands* and writes the result to standard output.

### Operands

The format of the expression to evaluate is shown as follows. *expr*, *expr1*, and *expr2* can be decimal integers or strings.

#### Note:

The six relational expressions return the result of a decimal integer comparison if both arguments are integers. Otherwise, they return the result of a string comparison. The result of each comparison is 1 if the specified relationship is true, or 0 if the relationship is false.

#### Expression

*expr1* | *expr2*

*expr1* & *expr2*

*expr1* = *expr2*

*expr1* > *expr2*

*expr1* >= *expr2*

*expr1* < *expr2*

*expr1* <= *expr2*

*expr1* != *expr2*

*expr1* + *expr2*

*expr1* - *expr2*

*expr1* \* *expr2*

*expr1* / *expr2*

*expr1* % *expr2*

*expr1* : *expr2*

( *expr* )

#### Description

Returns the evaluation of *expr1* if it is neither null nor zero; otherwise, returns the evaluation of *expr2*.

Returns the evaluation of *expr1* if neither expression evaluates to null or zero; otherwise, returns zero.

Equal.

Greater than.

Greater than or equal.

Less than.

Less than or equal.

Not equal.

Addition of decimal integers.

Subtraction of decimal integers.

Multiplication of decimal integers.

Division of decimal integers.

Remainder of decimal integer division.

Matching expression.

Grouping symbols.

### Exit Status

- 0 when the expression evaluates to neither null nor zero.
- 1 when the expression evaluates to null or zero.
- 2 when the expression is invalid.
- >2 when an error occurred.

### Examples

1. Evaluate an arithmetic expression.

```
expr 10+10*10/10-10
```

2. Evaluate a true or false condition.

expr 10 = 10

## hostname - Display the name of the current host system

### Synopsis

hostname [-is]

### Description

The **hostname** utility writes the name of the current host system to standard output.

### Options

- i Also display the IP address of the host system.
- s Display the short name of the host system without the domain information.

### Exit Status

- 0 when successful
- >0 when an error occurs

## id - Return user identity

### Synopsis

id [*user*]

id -G [-n] [*user*]

id -g [-nr] [*user*]

id -p [*user*]

id -u [-nr] [*user*]

### Description

The **id** utility displays the user and group names and numeric identifiers, of the calling process, to standard output. If the real and effective identifiers are different, both are displayed, otherwise only the real identifier is displayed.

If a *user* (login name or user identifier) is specified, the user and group identifiers of that user are displayed. In this case, the real and effective identifiers are assumed to be the same.

### Options

- G Display the different group identifiers (effective, real and supplementary) as white-space separated numbers, in no particular order.
- g Display the effective group identifier as a number.
- n Display the name of the user or group identifier for the -G, -g and -u options instead of the number. If any of the identifier numbers cannot be mapped into names, the number will be displayed as usual.
- p Make the output human-readable. The user identifier as a name is displayed, preceded by the keyword "uid". If the effective user identifier is different from the real user identifier, the real user identifier is displayed as a name, preceded by the keyword "euid". If the effective group identifier is different from the real group identifier, the real group identifier is displayed as a name,

preceded by the keyword "rgid". The list of groups to which the user belongs is then displayed as names, preceded by the keyword "groups". Each display is on a separate line.

- r** Display the real identifier for the **-g** and **-u** options instead of the effective identifier.
- u** Display the effective user identifier as a number.

#### Exit Status

- 0 on success
- >0 if an error occurs.

#### Related information

- "logname - Display user's login name" on page 182

#### Examples

1. Display all user and groups identifiers that belong to the user "SAM".

```
id -p SAM
uid SAM
groups 500, 1
```

## ipcrm - Remove interprocess communication identifier

### Synopsis

```
ipcrm [-m shmid] [-M shmkey] [-q msgid] [-Q msgkey] [-s semid] [-S semkey]
```

### Description

The **ipcrm** utility removes an interprocess communication (IPC) entry if the caller has the necessary authority to the IPC entry. The caller can specify an entry either by the key or by the identifier. The caller can remove multiple entries at once.

### Options

- M *shmkey***  
Remove the shared memory segment with the specified key.
- m *shmid***  
Remove the shared memory segment with the specified id.
- Q *msgkey***  
Remove the message queue with the specified key.
- q *msgid***  
Remove the message queue with the specified id.
- S *semKey***  
Remove the semaphore set with the specified key.
- s *semid***  
Remove the semaphore set with the specified id.

### Operands

There are no operands.

### Exit Status

- 0 on success
- >0 if an error occurs

## Related Information

- “ipcs - Report interprocess communication status”

## Examples

- Remove a semaphore with key 1283 and a message queue with id 10:  

```
ipcrm -S 1283 -q 10
```

## ipcs - Report interprocess communication status

### Synopsis

» ipcs [-ETabcjmnopqstu] «

### Description

The **ipcs** utility reports information about existing interprocess communication (IPC) entries on the system and displays the output on standard output. The **ipcs** utility is shipped with public authority set to \*EXCLUDE. The user must have \*SERVICE special authority to run **ipcs**.

**ipcs** automatically reports some information for all entries that match the IPC mechanism specified. Additional information is reported based on the specified options.

If no IPC mechanism is specified, all five mechanisms are reported. An IPC mechanism is specified by using the **-m** option for shared memory, **-n** option of named semaphores, **-s** option for semaphores sets, **-q** option for message queues, or **-u** option for unnamed semaphores.

The following information is reported for every shared memory, semaphore set, and message queue entry:

- The type of the mechanism (column T).
- The id of the entry in decimal form (column ID).
- The key of the entry in hexadecimal form (column KEY).
- The entry's access modes and flags (column MODE).
- The user profile of the owner of the entry (column OWNER).
- The group profile of the owner of the entry (column GROUP).

The following information is reported for every named semaphore entry:

- The type of the mechanism (column T).
- The title for the semaphore (column TITLE).
- The entry's access modes and flags (column MODE).

The following information is reported for every unnamed semaphore entry:

- The type of the mechanism (column T).
- The title for the semaphore (column TITLE).

**Warning:** Running **ipcs** locks system-scoped resources that can affect the performance of other IPC operations.

### Options

The following options are used to select the IPC mechanism to report on.

- m** Show the shared memory entries on the system.
- n** Show the named semaphore entries on the system.

- q Show the message queue entries on the system.
- s Show the semaphore set entries on the system.
- u Show the unnamed semaphore entries on the system.

The following options select the additional information that is reported for the IPC mechanism.

- a Report all information as if the **-b**, **-c**, **-o**, **-p**, and **-t** options were specified.
- b Display the maximum allowable size. If message queues are specified, the report includes the QBYTES column. If shared memory is specified, the report includes the SEGSZ column. If semaphore sets are specified, the report includes the NSEMS column. If named semaphores or unnamed semaphores are specified, the report includes the VALUE and NWAITERS columns.
- c Display the user profile and group profile of the creator of the entry. For all mechanisms, the report includes the CREATOR and CGROUP columns.
- E Display extended information. If message queues are specified, the report includes the WPID, WTID, MSGTYPE, and SIZE columns. If shared memory is specified, the report includes the APID and NUMATT columns. If semaphore sets are specified, the report includes the SEMNUM, SEMVAL, LOPID, WAITZ, WAITP, and WAITVAL columns. If named semaphores are specified, the report includes the NAME, LPOST, LWAIT, WAITER, JOB, and THREAD columns. If unnamed semaphores are specified, the report includes the LPOST, LWAIT, WAITER, JOB, and THREAD columns.

Since this level of detail is not available on other systems, this option is kept separate from the **-a** option. When this option is specified, at least one row is added for each entry.

- »-j Display the qualified job name instead of the process ID when the **-E** option is also specified. If message queues are specified, the report includes the WJOBID column instead of WPID. If shared memory is specified, the report includes the AJOBID column instead of APID. If semaphore sets are specified, the report includes the LOJOBID column instead of LOPID, the WAITZJID column instead of WAITZ, and the WAITPJID column instead of WAITP. «
- o Display information about outstanding usage. If message queues are specified, the report includes the CBYTES and QNUM columns. If shared memory is specified, the report includes the NATTC column.
- p Display process ID information. If message queues are specified, the report includes the LSPID and LRPID columns. If shared memory is specified, the report includes the CPID and LPID columns.
- t Display time information. If message queues are specified, the report includes the CTIME, RTIME, and STIME columns. If shared memory is specified, the report includes the CTIME, ATIME, and DTIME columns. If semaphore sets are specified, the report includes the CTIME and OTIME columns.
- T Display thread information. If message queues are specified, the report includes the LSTID and LRTID columns. If shared memory is specified, the report includes the CTID and LTID columns. If semaphore sets are specified and the **-E** option is specified, the report includes the LOTID, WAITZTID, and WAITPTID columns.

## Operands

There are no operands.

## Extended Description

Listed below are descriptions for all of the columns that can be reported in the output. After the column name, the options that display the column are shown. A value of "default" means that the column is always displayed, no matter what option is specified.



» **AJOBID (-Ej)**

The qualified job name of the jobs attached to the shared memory segment. «

**ATIME (-t, -a)**

The last time a job attached to the shared memory segment.

**APID (-E)**

The process ID of the job or jobs attached to the shared memory segment.

**CBYTES (-o, -a)**

The total number of bytes in the messages currently on the message queue.

**CGROUP (-c, -a)**

The group profile of the creator of the entry.

**CPID (-p, -a)**

The process ID of the job that created the shared memory segment.

**CTID (-T)**

The thread ID of the thread that created the shared memory segment.

**CREATOR (-c, -a)**

The user profile of the creator of the entry.

**CTIME (-t, -a)**

The last time the entry was either created or the owner or permissions, or both, were changed.

**DTIME (-t, -a)**

The last time a job detached from the shared memory segment.

**GROUP (default)**

The group profile of the owner of the entry.

**ID (default)**

The id of the entry in decimal.

**JOB (-E)**

The fully-qualified job name of the job waiting on the named semaphore or unnamed semaphore.

**KEY (default)**

The key of the entry in hexadecimal.

» **LOJOBID (-Ej)**

The qualified job name of the last job to change the value of the semaphore using semop(). «

**LOPID (-E)**

The process ID of the last job to change the value of the semaphore using semop().

**LOTID (-TE)**

The thread ID of the last thread to change the value of the semaphore using semop().

**LPID (-p, -a)**

The process ID of the last job to attach or detach from the shared memory segment or change the semaphore value.

**LPOST (-E)**

The fully-qualified job name and thread id of the last thread to post the named semaphore or unnamed semaphore.

**LRPID (-p, -a)**

The process ID of the last job to receive a message from the message queue using msgrcv().

**LRTID (-T)**

The thread ID of the last thread to receive a message from the message queue using msgrcv().

**LSPID (-p, -a)**

The process ID of the last job to send a message to the message queue using msgsnd().

**LSTID (-T)**

The thread ID of the last thread to send a message to the message queue using msgsnd().

**LTID (-T)**

The thread ID of the last thread to attach or detach from the shared memory segment.

**LWAIT (-E)**

The fully-qualified job name and thread id of the last thread to wait for the named semaphore or unnamed semaphore.

**MODE (default)**

An 11 character field that provides information about the state and permissions of the entry.

The first character can be one of the following:

- D** The entry has sustained damage, and no operations can be performed on it. The entry should only be marked damaged if an internal error has occurred.
- T** The entry is a shared memory segment and the segment uses teraspace storage.
- Y** The entry is a shared memory segment and the segment uses teraspace storage and the entry has sustained damage.
- None of the above applies.

The second character can be one of the following:

- R** The entry is a message queue and a thread is waiting on a call to msgrcv().
- S** The entry is a message queue and a thread is waiting on a call to msgsnd().
- D** The entry is a shared memory segment and the shared memory segment is marked to be removed when all the jobs detach from the shared memory.
- None of the above applies.

The next nine characters are interpreted as three sets of three permissions each. The first set refers to the owner's permissions, the second set to group's permissions, and the third set to other's permissions. Within each set, the first character indicates permission to read, the second character indicates permission to write, and the last character is currently unused.

The permissions are indicated as follows:

- r** If read permission is granted.
- w** If write permission is granted.
- If the indicated permission is not granted.

**MSGTYPE (-E)**

The type of the messages that are currently on the message queue.

**NAME (-E)**

The path name of the named semaphore.

**NATTCH (-o, -a)**

The current number of attaches to the shared memory segment.

**NUMATT (-E)**

The number of times the job is attached to the shared memory segment.

**NSEMS (-b, -a)**

The number of semaphores in the semaphore set.

**NWAITERS (-b, -a)**

The number of threads waiting on the named semaphore or unnamed semaphore.

**OTIME (-t, -a)**

The last time that semop() was called using the semaphore set.

**OWNER (default)**

The user profile of the owner of the entry.

**QBYTES (-b, -a)**

The maximum number of bytes allowed on the message queue.

**QNUM (-o, -a)**

The number of messages currently on the message queue.

**RTIME (-t, -a)**

The last time a msgrcv() was called using the message queue.

**SEGSZ (-b, -a)**

The size of the shared memory segment.

**SEMNUM (-E)**

The semaphore number in the semaphore set.

**SEMVAL (-E)**

The value of the semaphore.

**SIZE (-E)**

The size of the message on the message queue.

**STIME (-t, -a)**

The last time a msgsnd() was called using the message queue.

**T (default)**

The entry type. The value is M for a shared memory segment, N for a named semaphore, Q for a message queue, S for a semaphore set, or U for an unnamed semaphore.

**THREAD (-E)**

The thread ID of the thread waiting on the named semaphore or unnamed semaphore.

**TITLE (default)**

The title of the named semaphore or unnamed semaphore.

**VALUE (-b, -a)**

The current value of the named semaphore or unnamed semaphore.

**WAITER (-E)**

The index number of the thread waiting on the named semaphore or unnamed semaphore.

**WAITP (-E)**

The process ID of the job waiting for the semaphore value to reach a positive number.

**» WAITPJID (-Ej)**

The qualified job name of the job waiting for the semaphore value to reach a positive number. «

**WAITPTID (-ET)**

The thread ID of the thread or threads waiting for the semaphore value to reach a positive number.

**WAITVAL (-E)**

The value that the thread is waiting for the semaphore to reach.

**WAITZ (-E)**

The process ID of the job waiting for the semaphore value to reach zero.

### » WAITZJID (-Ej)

The qualified job name of the job waiting for the semaphore value to reach zero. <<

### WAITZTID (-ET)

The thread ID of the thread or thread waiting for the semaphore value to reach zero.

### » WJOBID (-Ej)

The qualified job names of the jobs waiting to receive a message. <<

### WPID (-E)

The process ID of the job or jobs waiting to receive a message.

### WTID (-E)

The thread ID of the thread waiting to receive a message.

### Exit Status

- 0 on success
- >0 if an error occurs

### Related Information

- “ipcrm - Remove interprocess communication identifier” on page 174

## locale - Get locale specific information

### Synopsis

**locale** [ -a ]

**locale** [ -ck ] *name* ...

### Description

The **locale** utility displays information about the current locale environment to standard output.

In the first synopsis form, **locale** writes the names and values of locale environment variables. When the **-a** option is specified, **locale** writes the names of all of the available locales on the system.

In the second synopsis form, **locale** writes detailed information about the locale category or keyword specified by *name*.

### Options

- a** Write information about all available locales.
- c** Display the names of the locale categories.
- k** Display the names of the locale keywords.

### Operands

The *name* operand can be one of the following locale categories or keywords:

- For category LC\_CTYPE the keywords include alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, xdigit, and codeset.
- For category LC\_MESSAGES the keywords include yesexpr, noexpr, yesstr, and nostr.
- For category LC\_MONETARY the keywords include int\_curr\_symbol, currency\_symbol, mon\_decimal\_point, mon\_grouping, mon\_thousands\_sep, positive\_sign, negative\_sign, int\_frac\_digits, frac\_digits, p\_cs\_precedes, p\_sep\_by\_space, n\_cs\_precedes, n\_sep\_by\_space, p\_sign\_posn, n\_sign\_posn, debit\_sign, credit\_sign, left\_parenthesis, right\_parenthesis, and crncystr.

- For category LC\_NUMERIC the keywords include `decimal_point`, `thousands_sep`, `grouping`, and `radixchar`.
- For category LC\_TIME the keywords include `abday`, `abday_1`, `abday_2`, `abday_3`, `abday_4`, `abday_5`, `abday_6`, `abday_7`, `day`, `day_1`, `day_2`, `day_3`, `day_4`, `day_5`, `day_6`, `day_7`, `abmon`, `ab_mon1`, `abmon_2`, `abmon_3`, `abmon_4`, `abmon_5`, `abmon_6`, `abmon_7`, `abmon_8`, `abmon_9`, `abmon_10`, `abmon_11`, `abmon_12`, `mon`, `mon_1`, `mon_2`, `mon_3`, `mon_4`, `mon_5`, `mon_6`, `mon_7`, `mon_8`, `mon_9`, `mon_10`, `mon_11`, `mon_12`, `d_t_fmt`, `d_fmt`, `t_fmt`, `am_pm`, `am_str`, `pm_str`, `era`, `era_d_fmt`, `era_year`, `t_fmt_ampm`, `era_t_fmt`, `era_d_t_fmt`, `alt_digits`.

### Exit Status

- 0 when successful
- >0 when unsuccessful

### Related information

- “`iconv` - Convert characters from one CCSID to another CCSID” on page 68
- “`tr` - Translate characters” on page 76
- Locale overview

### Examples

1. Display the current values of the locale environment variables.  
`locale`
2. Display the list of available locales on the system.  
`locale -a`

## logger - Log messages

### Synopsis

```
logger [-is] [-f file] [-t tag] [message ...]
```

### Description

The **logger** utility provides a shell command interface for writing messages to the QHST system log. If *message* is not specified, and the **-f** flag is not provided, standard input is logged.

### Options

- i** Log the process id of the **logger** process with each line.
- s** Log the *message* to standard error, as well as the system log.
- f** Log the specified *file*.
- t** Mark every line in the log with the specified *tag*.

### Exit Status

- 0 on success
- >0 if an error occurs.

### Examples

1. Send the file “`test.output.log`” to the system log.  
`logger -f test.output.log`
2. Send a message to the system log and standard error, and include a tag.  
`logger -s -t 'Tag your are it' My message is simple`

## logname - Display user's login name

### Synopsis

**logname**

### Description

The **logname** utility writes the user's login name to standard output followed by a newline.

The **logname** utility explicitly ignores the **LOGNAME** and **USER** environment variables because the environment cannot be trusted.

### Exit Status

- 0 on success
- >0 if an error occurs

### Related information

- "id - Return user identity" on page 173

## sysval - Retrieve system values or network attributes

### Synopsis

**sysval** [-p] *systemValue* ...

**sysval** -n [-p] *networkAttr* ...

### Description

The **sysval** utility displays the value of an i5/OS<sup>(TM)</sup> system value or network attribute. One system value or network attribute is displayed per line of output.

**Note:** This utility is unique to i5/OS<sup>(TM)</sup>.

### Options

- n Display network attributes.
- p Display the system value or network attribute name with the value.

### Operands

See the Retrieve System Value API for the names and descriptions of the valid system values. See the Retrieve Network Attributes API for the names and descriptions of the valid network attributes.

### Examples

1. Display the QDATE system value.  
`sysval QDATE`
2. Display the SYSNAME network attribute.  
`sysval -n SYSNAME`

## tee - Duplicate standard input

### Synopsis

**tee** [-ai] [*file* ...]

## Description

The **tee** utility copies standard input to standard output, making a copy in zero or more *files*. The output is unbuffered.

The **tee** utility takes the default action for all signals, except when the **-i** option is specified.

## Options

- a** Append the output to the files rather than overwriting them.
- i** Ignore the SIGINT signal.

## Environment Variables

**tee** is affected by the following environment variables:

### QIBM\_CCSID

The files created by **tee** are created with the CCSID specified by the value of the environment variable.

## Exit Status

- 0 on success
- >0 if an error occurs

## Related information

- “echo - Write arguments to standard output” on page 132

## Examples

1. Save the output of a command into three different files.  

```
grep 'off_set=' code/*.java | tee file1 file2 file3 > logfile
```
2. Make a working and backup copy of the file, "back9".  

```
cat back9 | tee pro.tees pro.tees.bak
```

## ulimit - Set or display resource limits

### Synopsis

```
ulimit [-HS] [-acdfmnst] [limit]
```

## Description

The **ulimit** utility sets or displays resource limits. The resource limits apply to the current process and to any processes that are started after the resource limit is set.

For each resource, there is a hard or maximum limit and a soft or current limit. The soft limit can be changed to any value that is less than or equal to the hard limit. The hard limit can be changed to any value that is greater than or equal to the soft limit. The hard limit can only be increased by a user with \*JOBCTL special authority.

On i5/OS<sup>(TM)</sup>, only the file size (**-f**) and number of descriptors (**-n**) resource limits can be set. All of the resource limits can be displayed.

## Options

- a** Display all of the resource limits.
- c** Display the resource limit for the maximum size of a core file in kilobytes.

- d** Display the resource limit for the maximum size of a process' data segment in kilobytes.
- f** Set or display the resource limit for the maximum size of a file in kilobytes.
- H** Set or display the hard limit for the resource.
- m** Display the resource limit for the maximum size of a process' total available storage.
- n** Set or display the resource limit for the maximum number of file descriptors that can be opened by the process.
- s** Display the resource limit for the maximum size of the process' stack in kilobytes.
- S** Set or display the soft limit for the resource.
- t** Display the resource limit for the maximum amount of CPU time in seconds.

### Operands

When *limit* is not specified, the value of the resource limit is displayed. When the **-H** option is specified, the hard limit is displayed. Otherwise, the soft limit is displayed.

When *limit* is specified, the value of the resource limit is set. The *limit* can be an "Arithmetic expansions" on page 18 or the string "unlimited" for no limit. If neither the **-H** or **-S** options are specified, both the hard and soft limits are set.

If no resource is specified, the default is the file size (**-f**) resource limit.

### Exit Status

- 0 when successful
- >0 when unsuccessful

### Related information

- "umask - Get or set the file mode creation mask" on page 129

## uname - Return system name

### Synopsis

**uname** [-amnrsv]

### Description

The **uname** utility writes the name of the operating system implementation to standard output. When options are specified, strings representing one or more system characteristics are written to standard output.

If the **-a** flag is specified, or multiple flags are specified, all output is written on a single line, separated by spaces.

### Options

- a** Behave as though the **-m**, **-n**, **-r**, **-s**, and **-v** options were specified.
- m** Write the name of the hardware type of the system to standard output.
- n** Write the name of the system to standard output.
- r** Write the current release level of the operating system to standard output.
- s** Write the name of the operating system implementation to standard output.
- v** Write the version level of this release of the operating system to standard output.



### Exit Status

- 0 on success
- >0 if an error occurs

### Related information

- “ulimit - Set or display resource limits” on page 183

---

## Application Programming Interfaces

Qshell provides the following application program interfaces (APIs):

“QzshSystem() - Run a QSH Command”

“QzshCheckShellCommand() - Find QSH Command” on page 187

### QzshSystem() - Run a QSH Command

#### Syntax

```
#include <qshell.h>

int QzshSystem(const char *command);
```

Threadsafe: Yes

The **QzshSystem()** function runs the specified shell command by spawning a child process and invoking **qsh** in the child process. **qsh** interprets and runs *command* and then exits.

The **QzshSystem()** function returns when the child process has ended. While the **QzshSystem()** function is waiting for the child process to end, it ignores the SIGQUIT and SIGINT signals, and blocks the SIGCHLD signal. The **QzshSystem()** function does not affect the status information of any other child processes started by the calling process.

### Parameters

*\*command*

(Input) Pointer to null-terminated string that contains the shell command to run.

### Authorities

| Object Referred To                                            | Authority Required | errno  |
|---------------------------------------------------------------|--------------------|--------|
| Each directory in the path name preceding the executable file | *X                 | EACCES |
| Executable file                                               | *X                 | EACCES |
| If executable file is a shell script                          | *RX                | EACCES |

### Return value

**value** **QzshSystem()** was successful. The return value is the status returned from the **waitpid()** function. An application can use the macros provided in the `sys/wait.h` header file to interpret the status information from the child process. The return value can be a negative number.

**-1** **QzshSystem()** was not successful. The *errno* value is set to indicate the error.

### Error conditions

If **QzshSystem()** is not successful, *errno* typically indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

**[EACCES]**

Permission denied.

An attempt was made to access an object in a way forbidden by its object access permissions.

The thread does not have access to the specified file, directory, component, or path.

**[ECHILD]**

Calling process has no remaining child processes on which wait operation can be performed.

**[EFAULT]**

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

**[EINVAL]**

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

**[ENOMEM]**

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function.

**[ENOSYSRSC]**

System resources not available to complete request.

**[EUNKNOWN]**

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated. Then try the operation again.

**Related information**

- “QzshCheckShellCommand() - Find QSH Command” on page 187
- `spawn()` - Spawn Process
- `waitpid()` - Wait for Specific Child Process

**Example: Using the QzshSystem() and QzshCheckShellCommand() functions**

The following example shows how to use the **QzshSystem()** and **QzshCheckShellCommand()** functions.

```
#include <stdio.h>
#include <qshell.h>
#include <sys/wait.h>
#include <errno.h>

int main(int argc, char *argv[])
{
 int status;
 char *command = "ls";

 /* Verify the user has access to the specified command. */
 if (QzshCheckShellCommand(command, NULL) == 0) {
 /* Run the specified command. */
 status = QzshSystem(command);
 }
}
```

```

 if (WIFEXITED(status)) {
 printf("Command %s completed with exit status %d.\n",
 command, WEXITSTATUS(status));
 }
 else if (WIFSIGNALED(status)) {
 printf("Command %s ended with signal %d.\n",
 command, WTERMSIG(status));
 }
 else if (WIFEXCEPTION(status)) {
 printf("Command %s ended with exception.\n", command);
 }
}
else
 printf("Error %d finding command %s\n", errno, command);

return(0);
}

```

### Output

Command `ls` completed with exit status 0.

## QzshCheckShellCommand() - Find QSH Command

```

Syntax
#include <qshell.h>

int QzshCheckShellCommand(const char *command, const char *path);

Threadsafe: Yes

```

The **QzshCheckShellCommand()** function finds the specified shell command by searching:

- for a built-in utility, then
- in each directory in the list specified by *path* or the **PATH** environment variable in turn.

An application can use **QzshCheckShellCommand()** to verify that *command* exists and the user has authority to *command* before running it.

### Parameters

*\*command*

(Input) Pointer to null-terminated string that contains the shell command to find.

*\*path*

(Input) Pointer to null-terminated string that contains a colon delimited list of directories to search. If this parameter is NULL, **QzshCheckShellCommand()** uses the value of the **PATH** environment variable.

### Authorities

When *command* is an executable file, the user must have the following authorities.

| Object Referred To                                            | Authority Required | errno  |
|---------------------------------------------------------------|--------------------|--------|
| Each directory in the path name preceding the executable file | *X                 | EACCES |
| Executable file                                               | *X                 | EACCES |
| If executable file is a shell script                          | *RX                | EACCES |

### Return value

- 0 **QzshCheckShellCommand()** was successful. The *command* was found in the current environment.
- 1 **Qp0zCheckShellCommand()** was not successful. The *errno* value is set to indicate the error.

### Error conditions

If **QzshCheckShellCommand()** is not successful, *errno* typically indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

#### [EACCES]

Permission denied.

An attempt was made to access an object in a way forbidden by its object access permissions.

The thread does not have access to the specified file, directory, component, or path.

#### [EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

#### [EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

#### [ENOMEM]

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function.

#### [ENOENT]

No such path or directory.

The directory or component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

#### [EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated. Then retry the operation.

### Related information

- “QzshSystem() - Run a QSH Command” on page 185

### Example: Using the **QzshCheckShellCommand()** function

For an example of using this function, see the “QzshSystem() - Run a QSH Command” on page 185 function.

---

## Examples: Using a remote client that connects to a qsh session

The following two example programs show how to use a remote client that connects to an interactive **qsh** session on the server.

- “Example: Server program” on page 189
- “Example: Client program” on page 196

The server program is compiled and run on i5/OS<sup>(TM)</sup>.

- See “Example: Creating and running the server program” on page 205 for more information.

The client program is compiled and run on a remote system.

- See “Example: Creating and running the client program” on page 207 for more information.

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

## Example: Server program

```
/*
/* Name: server.c
/*
/* Description: This program is a server for starting interactive
/* qsh sessions on remote clients. The program
/* listens for connections from clients. When a
/* connection is accepted, it reads the user name
/* and password of the client. It then swaps to the
/* the specified user profile and spawns a new
/* process running the qsh shell interpreter that
/* handles the connection.
/*
/* Parameters: 1. Port number to listen for connections on.
/*
/* Notes: 1. The user name and password are sent as plain text
/* from the client.
/* 2. The user profile running this program must have
/* authority to the QSYGETPH, QSYRLSPH, and
/* QWTSETP APIs.
/* 3. You will need to change the value of the NLSPATH
/* environment variable if your system is using a
/* different language than 2924.
/*
/*
/*****

/*****
/* Includes
/*****

#include <stdio.h> /* fopen(), vfprintf() */
#include <sys/socket.h> /* socket(), bind(), and so on. */
#include <netinet/in.h> /* sockaddr_in, INADDR_ANY, and so on */
```

```

#include <arpa/inet.h> /* inet_ntoa() */
#include <spawn.h> /* spawn() */
#include <unistd.h> /* close(), read(), and so on */
#include <stdlib.h> /* exit()*/
#include <stdarg.h> /* va_start(), va_end() */
#include <qp0z1170.h> /* Qp0zInitEnv() */
#include <qsygetph.h> /* QSYGETPH() */
#include <qwtsetp.h> /* QWTSETP() */
#include <qsyr1sph.h> /* QSYRLSPH() */
#include <qusec.h> /* Qus_EC_t */
#include <pwd.h> /* getpwnam() */
#include <ctype.h> /* toupper() */
#include <time.h> /* ctime(), time() */
#include <except.h> /* Exception and cancel handling */
#include <errno.h> /* errno and constants */

/*****
/* Constants */
*****/

#define DEFAULT_BUF 4096
#define DEFAULT_PORT 6042
#define NULL_PH "\0\0\0\0\0\0\0\0\0\0\0\0"
#define PH_SIZE 12
#define NAME_SIZE 11
#undef PATH_MAX
#define PATH_MAX 4096

/*****
/* Global Variables */
*****/

/* For logging errors */
FILE *log_fp;
char log_file[] = "/tmp/qsh_server.log";
char log_buffer[DEFAULT_BUF];

/*****
/* Function Prototypes */
*****/

int strtoupper(char *);
int GetString(int, char *, size_t);
void LogError(char *, ...);
void SendError(int, char *, ...);
void CleanupHandler(_CNL_Hndlr_Parms_T *);

int main(int argc, char *argv[])
{
 int sfd; /* Server's listening socket */
 int cfd; /* Socket connected to client */
 int on=1; /* Flag for setsockopt() */
 struct sockaddr_in my_addr; /* Address server binds to */
 struct sockaddr_in client_addr; /* Address of connected client */
 int client_addr_len; /* Length of client's socket address */
 unsigned short port; /* Server's TCP port */
 char server_ph[PH_SIZE+1] = NULL_PH; /* Server's profile handle */
 char client_ph[PH_SIZE+1] = NULL_PH; /* Client's profile handle */
 char profile[NAME_SIZE]; /* User profile read from client */
 char password[NAME_SIZE]; /* Password read from client */
 char sy_profile[NAME_SIZE]; /* User profile for i5/OS(TM) APIs */
 char sy_password[NAME_SIZE]; /* Password for i5/OS(TM) APIs */
 char server_profile[NAME_SIZE] = "CURRENT ";
 char no_pwd[NAME_SIZE] = "NOPWD ";
 struct passwd *cpw; /* User information for client */
 Qus_EC_t error = { sizeof(Qus_EC_t), 0 }; /* Error code for SPIs */

```

```

/* Parameters for spawn() to shell process */
char qsh_pgm[] = "/QSYS.LIB/QSHELL.LIB/QZSHSH.PGM";
char *args[5]; /* Argument array */
char *envs[10]; /* Environment variable array */
int fd_count; /* Number of descriptors */
int fd_map[3]; /* Map of descriptors */
struct inheritance inherit; /* Inheritance options */
char server_dir[] = "/"; /* Default current working directory */

/* Environment variables */
char home_var[PATH_MAX+10];
char logname_var[NAME_SIZE+10];
char path_var[] = "PATH=/usr/bin:";
char stdio_var[] = "QIBM_USE_DESCRIPTOR_STDIO=I";
char terminal_type_var[] = "TERMINAL_TYPE=REMOTE";
char nlspace_var[] = "NLSPATH=/QIBM/ProdData/OS400/Shell/MRI2924/%N";

volatile _INTRPT_Hndlr_Parms_T ca; /* For exception handler */

/*****
/* Process the input parameters. */
*****/

/* Use the default port if one is not specified. */
if (argc < 2) {
 port = DEFAULT_PORT;
}

else {
 port = atoi(argv[1]);
}

/*****
/* Initialize the server environment. */
*****/

/* Initialize for environment variables. */
Qp0zInitEnv();

/* Change to default directory. */
chdir(server_dir);

/* Initialize the server's profile handle. */
QSYGETPH(server_profile, no_pwd, server_ph, &error);
if (error.Bytes_Available != 0) {
 LogError("Could not get profile handle for server, "
 "QSYGETPH() failed with exception %7.7s\n",
 error.Exception_Id);
 exit(1);
}

/*****
/* Set up the listening socket. */
*****/

/* Create a socket. */
if ((sfd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP)) < 0) {
 LogError("socket() failed, errno=%d\n", errno);
 exit(1);
}

#pragma cancel_handler(CleanupHandler, sfd)
#pragma exception_handler(Cleanup, ca, _C1_ALL, _C2_ALL)

/* Allow re-use of this socket address. */
if (setsockopt(sfd, SOL_SOCKET, SO_REUSEADDR, (char *)&n,
 sizeof(int)) != 0) {

```

```

 LogError("setsockopt() failed, errno=%d\n", errno);
 exit(1);
}

/* Bind to a port. */
memset(&my_addr, '\0', sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_port = port;
my_addr.sin_addr.s_addr = INADDR_ANY;
if (bind(sfd, (struct sockaddr *)&my_addr, sizeof(my_addr)) != 0) {
 LogError("bind() failed for port %d, errno=%d\n", port, errno);
 close(sfd);
 exit(1);
}

/* Make this a listening socket. */
if (listen(sfd, 10) != 0) {
 LogError("listen() failed, errno=%d\n", errno);
 close(sfd);
 exit(1);
}

/*****
/* Accept connections from clients. */
*****/

while (1) {
 if ((cfd = accept(sfd, NULL, 0)) < 0) {
 LogError("accept() failed, errno=%d\n", errno);
 close(sfd);
 exit(1);
 }

 /* Read the user profile and password from the client. The client
 sends two null-terminated strings - the first one is the user
 profile and the second one is the password. */
 if (GetString(cfd, profile, 11) != 0) {
 getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
 LogError("Could not read profile from client at %s, port %hu\n",
 inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
 close(cfd);
 continue;
 }

 if (GetString(cfd, password, 11) != 0) {
 getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
 LogError("Could not read password from client at %s, port %hu\n",
 inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
 close(cfd);
 continue;
 }

 /* Check for the special values that turn off password checking in QSYGETPH(). */
 if ((profile[0] == '*') || (password[0] == '*')) {
 getpeername(cfd, (struct sockaddr *)&client_addr, &client_addr_len);
 LogError("Invalid password sent from client at %s, port %hu\n",
 inet_ntoa(client_addr.sin_addr), client_addr.sin_port);
 close(cfd);
 continue;
 }

 /* QSYGETPH() requires that the profile be exactly ten characters,
 left-aligned in the field, and padded with blanks. */
 strtoupper(profile);
 sprintf(sy_profile, "%-10.10s", profile);

 /* Get the profile handle for the client's user profile. */

```



```

QSYGETPH(sy_profile, password, client_ph, &error, strlen(password), 0);
if (error.Bytes_Available != 0) {
 LogError("Could not get profile handle for profile %s, "
 "QSYGETPH() failed with exception %7.7s\n",
 sy_profile, error.Exception_Id);
 SendError(cfd, "Could not get profile handle for profile %s\n",
 sy_profile);
 close(cfd);
 continue;
}

/* Switch to client's user profile. */
QWTSETP(client_ph, &error);
if (error.Bytes_Available != 0) {
 LogError("Could not switch to profile %s, "
 "QWTSETP() failed with exception %7.7s\n",
 sy_profile, error.Exception_Id);
 SendError(cfd, "Could not switch to profile %s\n", sy_profile);
 QSYRLSPH(client_ph, NULL);
 close(cfd);
 continue;
}

/* Get the info for this user profile. */
if ((cpw = getpwnam(profile)) == NULL) {
 /* Log error. */
 LogError("Could not retrieve information for profile %s, "
 "getpwnam() failed with errno=%d\n",
 profile, errno);
 SendError(cfd, "Could not retrieve information for profile %s\n",
 profile);

 /* Switch back to the server's user profile. */
 QWTSETP(server_ph, &error);
 if (error.Bytes_Available != 0) {
 LogError("Could not switch back to server's profile, "
 "QWTSETP() failed with exception %7.7s\n",
 error.Exception_Id);
 break;
 }

 /* Release the client's profile handle. */
 QSYRLSPH(client_ph, NULL);
 if (error.Bytes_Available != 0) {
 LogError("Could not release client's profile handle, "
 "QSYRLSPH() failed with exception %7.7s\n",
 error.Exception_Id);
 break;
 }
 close(cfd);
 continue;
}

/* Build the file descriptor map for the child. */
fd_count = 3;
fd_map[0] = cfd;
fd_map[1] = cfd;
fd_map[2] = cfd;

/* Build the argv array for the child. */
args[0] = qsh_pgm;
args[1] = "-login"; /* Do login processing */
args[2] = "-s"; /* Take input from stdin */
args[3] = "-i"; /* Run as an interactive shell */
args[4] = NULL;

/* Build the environ array for the child. */

```

```

sprintf(home_var, "HOME=%s", cpw->pw_dir);
sprintf(logname_var, "LOGNAME=%s", cpw->pw_name);
envs[0] = home_var;
envs[1] = logname_var;
envs[2] = path_var;
envs[3] = stdio_var;
envs[4] = terminal_type_var;
envs[5] = nlspath_var;
envs[6] = NULL;

/* Set up the inheritance structure. */
memset(&inherit, '\0', sizeof(struct inheritance));
inherit.flags = SPAWN_SETTHREADNP;
inherit.pgroup = SPAWN_NEWPGROUP;

/* Change to the home directory for the client. The child process
 inherits this as its current working directory. */
chdir(cpw->pw_dir);

/* Start a child process running the shell interpreter. */
if (spawn(args[0], fd_count, fd_map, &inherit, args, envs) < 0) {
 LogError("Could not start qsh process, spawn() failed with "
 "errno=%d\n", errno);
 SendError(cfd, "Could not start qsh process\n");
}

/* Clean up for the next connection. */
chdir(server_dir);
close(cfd);

/* Switch back to server's user profile. */
QWTSETP(server_ph, &error);
if (error.Bytes_Available != 0) {
 LogError("Could not switch back to server's profile, "
 "QWTSETP() failed with exception %7.7s\n",
 error.Exception_Id);
 break;
}

/* Release the client's profile handle. */
QSYRLSPH(client_ph, &error);
if (error.Bytes_Available != 0) {
 LogError("Could not release client's profile handle, "
 "QSYRLSPH() failed with exception %7.7s\n",
 error.Exception_Id);
 break;
}
} /* End of while */

/* Clean up. */
close(sfd);

#pragma disable_handler /* Exception handler */
#pragma disable_handler /* Cancel handler */

exit(0);
return 0;

/* Exception handler */
Cleanup:

LogError("Unexpected exception %7.7s\n", ca.Msg_Id);
close(sfd);
exit(1);
} /* End of main() */

```

```

/*
 * Convert a string to uppercase.
 */

int
strtoupper(char *string)
{
 for (; *string != '\0'; ++string)
 *string = toupper(*string);

 return 0;
} /* End of strtoupper() */

/*
 * Read a string from a socket.
 */

int
GetString(int fd, char *buffer, size_t nbytes)
{
 char c;
 do {
 if (read(fd, &c, 1) != 1) {
 return -1;
 }
 *buffer++ = c;
 if (--nbytes == 0) {
 return 0;
 }
 } while (c != '\0');

 return 0;
} /* End of GetString() */

/*
 * Write an error message to the log file.
 */

void LogError(char *format, ...)
{
 va_list ap;
 time_t now; /* Time stamp */

 /* If needed, open the log file. */
 if (log_fp == NULL) {
 log_fp = fopen(log_file, "w");
 if (log_fp == NULL) {
 return;
 }
 }

 /* Write timestamp to the log file. */
 now=time(NULL);
 fprintf(log_fp, "\n%s", ctime(&now));

 /* Write the formatted string to the log file. */
 va_start(ap, format);
 vfprintf(log_fp, format, ap);
 va_end(ap);

 /* Flush output to log file. */
 fflush(log_fp);

 return;
} /* End of LogError() */

```

```

/*
 * Send an error message to the client.
 */

void SendError(int fd, char *format, ...)
{
 va_list ap;

 /* Build the formatted string. */
 va_start(ap, format);
 vsprintf(log_buffer, format, ap);
 va_end(ap);

 /* Write the formatted string. */
 write(fd, log_buffer, strlen(log_buffer));

 return;
} /* End of SendError() */

/*
 * Handler to clean up when the program is canceled.
 */

void CleanupHandler(_CNL_Hndlr_Parms_T *cancel_info)
{
 int sfd;
 sfd = *((int *)cancel_info->Com_Area);
 close(sfd);
} /* End of CleanupHandler() */

```

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 212.

## Example: Client program

```

/*****
/*
/* Name: qshc.c
/*
/* Description: This program is a client for an interactive qsh
/* session running on a server. The program
/* first connects to a server on the specified
/* server and sends the user name and password of
/* the client. After the qsh session is started,
/* the program takes input from stdin and sends it
/* to the server and receives output from the server
/* and displays it on stdout.
/*
/*
/* Parameters: 1. Host running the qsh server (either host name or
/* IP address).
/*
/*
/* Options: 1. -n to force prompt for user name and password.
/* 2. -p to specify port of qsh server.
/*
/*
/* Notes: 1. The user name and password are sent as plain text
/* to the server.
/* 2. All translations from ASCII to EBCDIC are done by
/* this program on the client.
/* 3. The program includes translation tables for
/* converting between EBCDIC code page 37 (US English)
/* and ASCII code page 850 (US English). You can
/* modify these tables to support other code pages.
/* Or if your system supports the iconv APIs, you
*****/

```

```

/* can define USE_ICONV to translate using iconv(). */
/* 4. This program has been tested on AIX(R) 4.1.5 and */
/* Linux(TM) 2.0.29. */
/* */
/*****/

/* Remove the comments from the following line to use iconv(). */
/* #define USE_ICONV 1 */

/*****/
/* Includes */
/*****/

#include <stdio.h> /* perror() */
#include <sys/socket.h> /* socket(), bind(), and so on */
#include <netinet/in.h> /* sockaddr_in, INADDR_ANY, and so on */
#include <unistd.h> /* close(), read(), write() and so on */
#include <stdlib.h> /* exit() */
#include <stdlib.h> /* exit(), memset() */
#include <sys/ioctl.h> /* ioctl() */
#include <errno.h> /* errno and values */
#include <string.h> /* strlen() */
#include <arpa/inet.h> /* inet_addr() */
#include <netdb.h> /* gethostbyname() */
#include <pwd.h> /* getpwuid() */
#include <signal.h> /* sigaction(), and so on */

#ifdef _AIX
#include <sys/select.h> /* select() */
#include <strings.h> /* bzero() for FD_ZERO */
#endif
#ifdef __linux__
#include <sys/time.h> /* FD_SET(), select */
#endif

#ifdef USE_ICONV
#include <iconv.h> /* iconv(), and so on */
#endif

/*****/
/* Constants */
/*****/

#define QSH_PORT 6042
#define DEFAULT_BUF 4096

/*****/
/* Types */
/*****/

typedef unsigned char uchar;

/*****/
/* Global Variables */
/*****/

char *sysname; /* Long host name of server system */

#ifdef USE_ICONV
iconv_t ecd; /* Conversion descriptor for ASCII to EBCDIC */
iconv_t acd; /* Conversion descriptor for EBCDIC to ASCII */

#else
/* EBCDIC to ASCII translation table */
static uchar AsciiTable[256] =
{
 0x00,0x01,0x02,0x03,0x20,0x09,0x20,0x7f, /* 00-07 */

```

```

0x20,0x20,0x20,0x0b,0x0c,0x0d,0x0e,0x0f, /* 08-0f */
0x10,0x11,0x12,0x13,0x20,0x0a,0x08,0x20, /* 10-17 */
0x18,0x19,0x20,0x20,0x20,0x1d,0x1e,0x1f, /* 18-1f */
0x20,0x20,0x1c,0x20,0x20,0x0a,0x17,0x1b, /* 20-27 */
0x20,0x20,0x20,0x20,0x20,0x05,0x06,0x07, /* 28-2f */
0x20,0x20,0x16,0x20,0x20,0x20,0x20,0x04, /* 30-37 */
0x20,0x20,0x20,0x20,0x20,0x14,0x15,0x20,0x1a, /* 38-3f */
0x20,0x20,0x83,0x84,0x85,0xa0,0xc6,0x86, /* 40-47 */
0x87,0xa4,0xbd,0x2e,0x3c,0x28,0x2b,0x7c, /* 48-4f */
0x26,0x82,0x88,0x89,0x8a,0xa1,0x8c,0x8b, /* 50-57 */
0x8d,0xe1,0x21,0x24,0x2a,0x29,0x3b,0xaa, /* 58-5f */
0x2d,0x2f,0xb6,0x8e,0xb7,0xb5,0xc7,0x8f, /* 60-67 */
0x80,0xa5,0xdd,0x2c,0x25,0x5f,0x3e,0x3f, /* 68-6f */
0x9b,0x90,0xd2,0xd3,0xd4,0xd6,0xd7,0xd8, /* 70-77 */
0xde,0x60,0x3a,0x23,0x40,0x27,0x3d,0x22, /* 78-7f */
0x9d,0x61,0x62,0x63,0x64,0x65,0x66,0x67, /* 80-87 */
0x68,0x69,0xae,0xaf,0xd0,0xec,0xe7,0xf1, /* 88-8f */
0xf8,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f,0x70, /* 90-97 */
0x71,0x72,0xa6,0xa7,0x91,0xf7,0x92,0xcf, /* 98-9f */
0xe6,0x7e,0x73,0x74,0x75,0x76,0x77,0x78, /* a8-a7 */
0x79,0x7a,0xad,0xa8,0xd1,0xed,0xe8,0xa9, /* a8-af */
0x5e,0x9c,0xbe,0xfa,0xb8,0x15,0x14,0xac, /* b0-b7 */
0xab,0xf3,0x5b,0x5d,0xee,0xf9,0xef,0x9e, /* b8-bf */
0x7b,0x41,0x42,0x43,0x44,0x45,0x46,0x47, /* c0-c7 */
0x48,0x49,0xf0,0x93,0x94,0x95,0xa2,0xe4, /* c8-cf */
0x7d,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f,0x50, /* d0-d7 */
0x51,0x52,0xfb,0x96,0x81,0x97,0xa3,0x98, /* d8-df */
0x5c,0xf6,0x53,0x54,0x55,0x56,0x57,0x58, /* e0-e7 */
0x59,0x5a,0xfc,0xe2,0x99,0xe3,0xe0,0xe5, /* e8-ef */
0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37, /* f0-f7 */
0x38,0x39,0xfd,0xea,0x9a,0xeb,0xe9,0xff /* f8-ff */
};

```

```

/* ASCII to EBCDIC translation table */
static uchar EbcDicTable[256] =
{
0x00,0x01,0x02,0x03,0x37,0x2d,0x2e,0x2f, /* 00-07 */
0x16,0x05,0x25,0x0b,0x0c,0x0d,0x0e,0x0f, /* 08-0f */
0x10,0x11,0x12,0x13,0x3c,0x3d,0x32,0x26, /* 10-17 */
0x18,0x19,0x3f,0x27,0x22,0x1d,0x1e,0x1f, /* 18-1f */
0x40,0x5a,0x7f,0x7b,0x5b,0x6c,0x50,0x7d, /* 20-27 */
0x4d,0x5d,0x5c,0x4e,0x6b,0x60,0x4b,0x61, /* 28-2f */
0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7, /* 30-37 */
0xf8,0xf9,0x7a,0x5e,0x4c,0x7e,0x6e,0x6f, /* 38-3f */
0x7c,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7, /* 40-47 */
0xc8,0xc9,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6, /* 48-4f */
0xd7,0xd8,0xd9,0xe2,0xe3,0xe4,0xe5,0xe6, /* 50-57 */
0xe7,0xe8,0xe9,0xba,0xe0,0xbb,0xb0,0x6d, /* 58-5f */
0x79,0x81,0x82,0x83,0x84,0x85,0x86,0x87, /* 60-67 */
0x88,0x89,0x91,0x92,0x93,0x94,0x95,0x96, /* 68-6f */
0x97,0x98,0x99,0xa2,0xa3,0xa4,0xa5,0xa6, /* 70-77 */
0xa7,0xa8,0xa9,0xc0,0x4f,0xd0,0xa1,0x07, /* 78-7f */
0x68,0xdc,0x51,0x42,0x43,0x44,0x47,0x48, /* 80-87 */
0x52,0x53,0x54,0x57,0x56,0x58,0x63,0x67, /* 88-8f */
0x71,0x9c,0x9e,0xcb,0xcc,0xcd,0xdb,0xdd, /* 90-97 */
0xdf,0xec,0xfc,0x70,0xb1,0x80,0xbf,0x40, /* 98-9f */
0x45,0x55,0xee,0xde,0x49,0x69,0x9a,0x9b, /* a8-a7 */
0xab,0xaf,0x5f,0xb8,0xb7,0xaa,0x8a,0x8b, /* a8-af */
0x40,0x40,0x40,0x40,0x40,0x65,0x62,0x64, /* b0-b7 */
0xb4,0x40,0x40,0x40,0x40,0x40,0x4a,0xb2,0x40, /* b8-bf */
0x40,0x40,0x40,0x40,0x40,0x40,0x46,0x66, /* c0-c7 */
0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x9f, /* c8-cf */
0x8c,0xac,0x72,0x73,0x74,0x89,0x75,0x76, /* d0-d7 */
0x77,0x40,0x40,0x40,0x40,0x6a,0x78,0x40, /* d8-df */
0xee,0x59,0xeb,0xed,0xcf,0xef,0xa0,0x8e, /* e0-e7 */
0xae,0xfe,0xfb,0xfd,0x8d,0xad,0xbc,0xbe, /* e8-ef */
0xca,0x8f,0x40,0xb9,0xb6,0xb5,0xe1,0x9d, /* f0-f7 */
}

```

```

 0x90,0xbd,0xb3,0xda,0xea,0xfa,0x40,0x40 /* f8-ff */
};
#endif /* USE_ICONV */

/*****
/* Function Prototypes */
*****/

int ConvertToEBCDIC(char *, size_t, char *, size_t);
int ConvertToASCII(char *, size_t, char *, size_t);
int GetPassword(char *, char *, char *);
int Translate(uchar *, size_t, uchar *, uchar *);
void MySignalHandler(int);
void usage(void);

int main (int argc, char *argv[])
{
 struct sigaction sigact; /* Signal action */
 int c; /* Option letter */
 int nflag=0; /* True when -n option is specified */
 int port=QSH_PORT; /* Port to connect to on server */
 int sd; /* Socket to server */
 fd_set read_set; /* For select() */
 int rc; /* Return code */
 struct sockaddr_in svr_addr; /* AF_INET socket address */
 long ip_addr; /* IP address of server system */
 struct in_addr host_addr; /* Host address for gethostbyaddr() */
 char *hostname; /* Short host name of server system */
 size_t len; /* Length of input string */
 char *ascii_user; /* Username in ASCII */
 char *ebcdic_user; /* Username in EBCDIC */
 char *ascii_pwd; /* Password in ASCII */
 char *ebcdic_pwd; /* Password in EBCDIC */
 struct hostent *host_p; /* Pointer to hostent structure returned by
 gethostbyname() */

 char *ascii_buf; /* Buffer for ASCII text */
 char *ebcdic_buf; /* Buffer for EBCDIC text */
 int buf_size; /* Amount of data read from server */

 /*****
 /* Initialization. */
 *****/

#ifdef USE_ICONV
 /* Open the conversion descriptors for converting between ASCII and
 EBCDIC. Assume the server job is running in CCSID 37.
 This must be changed if the server job is running in a
 different CCSID. The input parameters to iconv_open() may need to
 be changed depending on the operating system. This iconv_open() is
 coded for AIX. */
 if ((acd = iconv_open("IBM-850", "IBM-037")) < 0) {
 perror("qshc: iconv_open() failed for ASCII to EBCDIC");
 exit(1);
 }

 if ((ecd = iconv_open("IBM-037", "IBM-850")) < 0) {
 perror("qshc: iconv_open() failed for EBCDIC to ASCII");
 exit(1);
 }
#endif /* USE_ICONV */

 /* Set up a signal handler for SIGINT. The signal is sent to the
 process when the user presses <ctrl>c. */
 sigemptyset(&sigact.sa_mask);
 sigact.sa_flags = 0;
 sigact.sa_handler = MySignalHandler;
 if (sigaction(SIGINT, &sigact, NULL) != 0) {

```

```

 perror("qshc: sigaction(SIGINT) failed");
 exit(1);
}

/*****
/* Process the input parameters. */
*****/

if (argc < 2) {
 usage();
}

/* Process the options. */
while ((c = getopt(argc, argv, "hnp:")) != EOF) {
 switch (c) {
 case 'n':
 nflag = 1;
 break;
 case 'p':
 port = atoi(optarg);
 break;
 case 'h':
 default:
 usage();
 break;
 } /* End of switch */
} /* End of while */

/* Convert a dotted decimal address to a 32-bit IP address. */
hostname = argv[optind];
ip_addr = inet_addr(hostname);

/* When inet_addr() returns -1 assume the user specified
a host name. */
if (ip_addr == -1) {
 /* Try to find the host by name. */
 host_p = gethostbyname(hostname);
 if (host_p) {
 memcpy(&ip_addr, host_p->h_addr, host_p->h_length);
 sysname = host_p->h_name;
 }

 else {
 fprintf(stderr, "qshc: Could not find host %s\n", hostname);
 exit(1);
 }
} /* End of if */

/* The user specified a IP address. */
else {
 /* Try to find the host by address. */
 host_addr.s_addr = ip_addr;
 host_p = gethostbyaddr((char *)&host_addr.s_addr, sizeof(host_addr),
 AF_INET);
 if (host_p) {
 sysname = host_p->h_name;
 }

 else {
 fprintf(stderr, "qshc: Could not find host %s\n", hostname);
 exit(1);
 }
} /* End of else */

/*****
/* Connect to the qsh server on the specified system. */
*****/

```



```

/* Create a socket. */
if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP)) < 0) {
 perror("qshc: socket() failed");
 exit(1);
}

/* Connect to the qsh server on the specified system. */
memset(&svr_addr, '\0', sizeof(svr_addr));
svr_addr.sin_family = AF_INET;
svr_addr.sin_port = htons(port);
svr_addr.sin_addr.s_addr = ip_addr;
if (connect(sd, (struct sockaddr *)&svr_addr, sizeof(svr_addr)) != 0) {
 perror("qshc: connect() failed");
 exit(1);
}

/*****
/* Send the user name and password to the server. */
*****/

/* Allocate buffers for translating input and output. */
ascii_buf = (char *)malloc(DEFAULT_BUF);
memset(ascii_buf, '\0', DEFAULT_BUF);
ebcdic_buf = (char *)malloc(DEFAULT_BUF);
memset(ebcdic_buf, '\0', DEFAULT_BUF);

ascii_user = ascii_buf;
ascii_pwd = ascii_buf + 100;
ebcdic_user = ebcdic_buf;
ebcdic_pwd = ebcdic_buf + 100;

/* Prompt the user for the user name and password. */
if (nflag) {
 printf("Enter user name: ");
 gets(ascii_user);
 ascii_pwd = getpass("Enter password: ");
}

/* Get the user name and password from the ~/.netrc file. */
else {
 if (GetPassword(hostname, ascii_user, ascii_pwd) != 0) {
 fprintf(stderr, "qshc: Could not find user or password in ~/.netrc\n");
 exit(1);
 }
}

/* Convert the user name and password to EBCDIC. */
if (ConvertToEBCDIC(ascii_user, strlen(ascii_user)+1, ebcdic_user, 11) < 0) {
 fprintf(stderr, "qshc: Could not convert user %s to EBCDIC\n", ascii_user);
 exit(1);
}

if (ConvertToEBCDIC(ascii_pwd, strlen(ascii_pwd)+1, ebcdic_pwd, 11) < 0) {
 fprintf(stderr, "qshc: Could not convert password %s to EBCDIC\n",
 ascii_pwd);
 exit(1);
}

/* Send the user name and password to the qsh server. Note that the
user name and password are sent as plain text. */
if ((rc = write(sd, (void *)ebcdic_user, strlen(ebcdic_user)+1)) < 0) {
 perror("qshc: write() failed sending username\n");
 close(sd);
 exit(1);
}

```

```

if ((rc = write(sd, (void *)ebcdic_pwd, strlen(ebcdic_pwd)+1)) < 0) {
 perror("qshc: write() failed sending password\n");
 close(sd);
 exit(1);
}
printf("Started qsh session on %s\n\n", sysname);

/*****
/* Process input and output between the user and the remote shell. */
*****/

/* Loop forever. */
while (1) {
 /* Select on stdin and the socket connected to the remote shell. */
 FD_ZERO(&read_set);
 FD_SET(0, &read_set);
 FD_SET(sd, &read_set);

 rc = select(sd+1, &read_set, NULL, NULL, NULL);

 if ((rc < 0) && (errno != EINTR)) {
 perror("qshc: select() failed");
 exit(1);
 }

 if (rc == 0) {
 continue;
 }

 /* Process data entered by the terminal user. */
 if (FD_ISSET(0, &read_set)) {
 /* Read the data from the terminal. */
 gets(ascii_buf);

 /* Convert the string to EBCDIC. */
 len = strlen(ascii_buf);
 if (ConvertToEBCDIC(ascii_buf, len, ebcdic_buf, DEFAULT_BUF) < 0) {
 fprintf(stderr, "qshc: Could not convert input string to EBCDIC\n");
 continue;
 }

 /* Put a newline on the end of the string. */
 *(ebcdic_buf+len) = 0x25;

 /* Send the data to the remote shell. */
 if (write(sd, ebcdic_buf, len+1) < 0) {
 perror("qshc: write() failed sending input");
 }
 }

 /* Process data from the remote shell. */
 if (FD_ISSET(sd, &read_set)) {
 /* Read the data from the remote shell. */
 buf_size = read(sd, ebcdic_buf, DEFAULT_BUF-1);

 /* There was a failure reading from the remote shell. */
 if (buf_size < 0) {
 perror("\nqshc: error reading data from remote shell");
 printf("\nEnded qsh session on %s\n", sysname);
 exit(0);
 }

 /* The remote shell process ended. */
 else if (buf_size == 0) {
 printf("\nEnded qsh session on %s\n", sysname);
 exit(0);
 }
 }
}

```

```

 /* Process the data from the remote shell. */
 else {
 /* Convert to ASCII. */
 *(ebcdic_buf+buf_size) = '\0';
 if (ConvertToASCII(ebcdic_buf, buf_size+1, ascii_buf,
 DEFAULT_BUF) >= 0) {
 write(1, ascii_buf, buf_size);
 }
 }
}
} /* End of while */

exit(0);
} /* End of main() */

/*
 * Convert a string from ASCII to EBCDIC.
 */

int
ConvertToEBCDIC(char *ibuf, size_t ileft, char *obuf, size_t oleft)
{
 int rc;

 #ifdef USE_ICONV
 rc = iconv(ebcd, (const char**)&ibuf, &ileft, &obuf, &oleft);
 #else
 rc = Translate((uchar *)ibuf, ileft, (uchar *)obuf, EbcdicTable);
 #endif
 if (rc < 0)
 perror("qshc: error converting to EBCDIC");

 return rc;
} /* End of ConvertToEBCDIC() */

/*
 * Convert a string from EBCDIC to ASCII.
 */

int
ConvertToASCII(char *ibuf, size_t ileft, char *obuf, size_t oleft)
{
 int rc;

 #ifdef USE_ICONV
 rc = iconv(asc, (const char**)&ibuf, &ileft, &obuf, &oleft);
 #else
 rc = Translate((uchar *)ibuf, ileft, (uchar *)obuf, AsciiTable);
 #endif
 if (rc < 0)
 perror("qshc: error converting to ASCII");

 return rc;
} /* End of ConvertToASCII() */

/*
 * Get the user name and password for the specified system from the
 * ~/.netrc file.
 */

int
GetPassword(char *sysname, char *logname, char *password)
{

```

```

#define BUFSIZE 256
char buffer[BUFSIZE];
char *systag, *logtag;
int logflag = 0, pwdflag = 0;
FILE *netrc;
struct passwd *pwdbuf;
int rc=0;

/* Get user's home directory. */
pwdbuf = getpwuid(getuid());

/* Does user have a .netrc file in their home directory? */
strcat(strcpy(buffer, pwdbuf->pw_dir), ".netrc");

if ((netrc = fopen(buffer, "r")) == NULL) {
 perror("qshc: open() failed for ~/.netrc file");
 return -1;
}

while (!(logflag || pwdflag) && fgets(buffer, BUFSIZE, netrc) != NULL) {
 /* Find system name in ~/.netrc. */
 if ((systag = (char*)strtok(buffer, " \t\n")) != NULL &&
 !strcmp(systag, "machine", 7)) {
 systag = (char *)strtok(NULL, " \t\n");
 if (!strcmp(systag, sysname)) {
 /* Find login and password. */
 while (!(logflag || !pwdflag)) {
 if ((logtag = (char *)strtok(NULL, " \t\n")) == NULL) {
 /* Nothing else on that line... get another. */
 while (!logtag) {
 fgets(buffer, BUFSIZE, netrc);
 logtag = (char *)strtok(buffer, " \t\n");
 }
 }

 if (!strcmp(logtag, "login", 5)) {
 strcpy(logname, strtok(NULL, " \n\t"));
 ++logflag;
 }
 else if (!strcmp(logtag, "password", 8)) {
 strcpy(password, strtok(NULL, " \n\t"));
 ++pwdflag;
 }
 else
 ;
 } /* while flags not set */
 } /* if found login and passwd in .netrc */
 } /* if machine in .netrc */
} /* while fgets */

fclose(netrc);

/* Login and password not found for system. */
if (!(logflag && pwdflag)) {
 rc = -1;
}

return rc;
} /* End of GetPassword() */

```

```

#ifdef USE_ICONV
/*
 * Translate bytes using the specified translation table.
 */

```

```
int
```

```

Translate(uchar *ip, size_t ilen, uchar *op, uchar *table)
{
 int index;
 for (index = 0; index < ilen; ++index) {
 *op = table[*ip];
 ip++;
 op++;
 }

 return 0;
} /* End of Translate() */
#endif

/*
 * Signal handler.
 */

void
MySignalHandler(int signo)
{
 switch (signo) {
 case SIGINT:
 printf("\nqshc: <ctrl>c ends this program\n");
 printf("Ended qsh session on %s\n", sysname);
 exit(0);
 break;

 default:
 exit(1);
 break;
 } /* End of switch */

 return;
} /* End of MySignalHandler() */

/*
 * Display usage message.
 */

void usage(void)
{
 fprintf(stderr, "Usage: qshc [-n] [-p port] hostname\n");
 exit(1);
} /* End of usage() */

```

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 212.

## Example: Creating and running the server program

**Creating the server program** The following example shows how to create the server program on i5/OS<sup>™</sup>. The example assumes that the source for the server program is in member SERVER in the file QGPL/QCSRC. The server program is owned by a special user profile QSHSVR that has minimal authorities but private authority to the QSYGETPH(), QSYRLSPH(), and QWTSETP() APIs. It is not possible to sign on using the QSHSVR user profile. The server program adopts the authority of QSHSVR so it can switch to the client’s user profile.

```

CRTBND PGM(QGPL/SERVER)
 SRCFILE(QGPL/QCSRC)
 SRCMBR(SERVER)
 OPTIMIZE(40)
 SYSIFCOPT(*IFSI0)
 LOCALETYPE(*LOCALE)
 USRPRF(*OWNER)

```

```

AUT(*USE)
TEXT('Shell server')
CRTUSRPRF USRPRF(QSHSVR)
PASSWORD(*NONE)
USRCLS(*USER)
TEXT('Shell server profile')
CHGOBJOWN OBJ(QGPL/SERVER)
OBJTYPE(*PGM)
NEWOWN(QSHSVR)
GRTOBJAUT OBJ(QSYS/QSYGETPH)
OBJTYPE(*PGM)
USER(QSHSVR)
AUT(*USE)
GRTOBJAUT OBJ(QSYS/QSYRLSPH)
OBJTYPE(*PGM)
USER(QSHSVR)
AUT(*USE)
GRTOBJAUT OBJ(QSYS/QWTSETP)
OBJTYPE(*PGM)
USER(QSHSVR)
AUT(*USE)

```

**Running the server program** You may want to run the server program and any child processes started by the server in their own subsystem. The following example shows how to create the following objects:

- A subsystem description and related routing entry and prestart job entries for both non-threaded and multi-thread capable jobs.
- A class.
- A job description.
- A job queue.

```

CRTSBSD SBSD(QGPL/SHELL)
POOLS((1 *BASE))
AUT(*USE)
TEXT('Shell server subsystem')
CRTCLS CLS(QGPL/SHELL)
RUNPTY(20)
TIMESLICE(2000)
DFTWAIT(30)
AUT(*USE)
TEXT('Shell server class')
CRTJOBQ JOBQ(QGPL/SHELL)
AUTCHK(*DTAAUT)
AUT(*USE)
TEXT('Shell server job queue')
CRTJOBQ JOBQ(QGPL/SHELL)
JOBQ(QGPL/SHELL)
AUT(*USE)
TEXT('Shell server job description')
ADDJOBQE SBSD(QGPL/SHELL)
JOBQ(QGPL/SHELL)
MAXACT(*NOMAX)
ADDRTGE SBSD(QGPL/SHELL)
SEQNBR(1)
CMPVAL(*ANY)
PGM(*LIBL/QCMD)
ADDPJE SBSD(QGPL/SHELL)
PGM(QSYS/QP0ZSPWP)
USER(QSHSVR)
STRJOBS(*YES)
INLJOBS(10)
THRESHOLD(2)
ADLJOBS(3)
MAXJOBS(*NOMAX)
JOBQ(QGPL/SHELL)
ADDPJE SBSD(QGPL/SHELL)

```

```
PGM(QSYS/QPOZSPWT)
USER(QSHSVR)
STRJOBS(*YES)
INLJOBS(10)
THRESHOLD(2)
ADLJOBS(3)
MAXJOBS(*NOMAX)
JOBQ(QSYS/QAMTJOBQ)
```

**Starting the subsystem** The following example shows how to start the subsystem described in the previous example and the server program.

```
STRSBS SBSQ(QGPL/QSHELL)
SBMJOB CMD(CALL QGPL/SERVER)
 JOB(SERVER)
 JOBQ(QGPL/SHELL)
 JOBQ(QGPL/SHELL)
 USER(QSHSVR)
```

## Example: Creating and running the client program

**Creating the client program** The following example shows how to create the client program on AIX using xlc. The example assumes that the source for the client program is in file qshc.c in the current working directory. The client program has been compiled and tested on AIX 4.1.5 using xlc and Linux 2.0.29 using gcc 2.7.2.1.

```
xlc -o qshc qshc.c
```

**Running the client program** The following example shows how to run the client program and connect to a server running on system myas400. Before running the command, there must be an entry in your ~/.netrc file for the specified system and the server must be started and listening on TCP/IP port 6042.

```
qshc myas400
```





---

## Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Programming Interface Information

This Qshell publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of i5/OS.

---

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX  
AS/400  
DB2  
DB2 Universal Database  
eServer  
IBM  
iSeries  
i5/OS  
OS/2  
RDN

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

---

## Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

---

## Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.





Printed in USA