wait — Command

Await completion of background process
wait [*pid***]**

Typing the character '&' after a command tells the shell **sh** to execute it as a *background* (or *detached*) process; otherwise, it is executed as a *foreground* process. You can perform other tasks while a background process is being executed. The shell prints the process id number of each background process when it is invoked. **ps** reports on currently active processes.

The command **wait** tells the shell to suspend execution until the child process with the given *pid* is completed. If no *pid* is given, **wait** suspends execution until all background processes are completed. If the process with the given *pid* is not a child process of the current shell, **wait** returns immediately.

The shell executes **wait** directly.

See Also

commands, ksh, ps, sh

Notes

If a subshell invokes a background process and then terminates, **wait** returns immediately rather than waiting for the termination of the grandchild process.

wait.h — Header File

Define wait routines
#include <sys/wait.h>

Header file **wait.h** declares prototypes for the functions **wait()** and **waitpid()**. It also defines manifest constants used with those functions.

See Also

header files, wait(), waitpid()

wait() — System Call (libc)

Await completion of a child process
#include <sys/wait.h>
wait(*statp***)**
int **statp***;*

wait() suspends execution of the invoking process until a child process (created with **fork()**) terminates. It returns the process identifier of the terminating child process. If there are no children or if an interrupt occurs, it returns -1.

If it is successful, **wait()** returns the process identifier of the terminated child process. In addition, **wait()** fills in the integer pointed to by *statp* with exit-status information about the completed process. If *statp* is NULL, **wait()** discards the exit-status information.

wait() fills in the low byte of the status-information word with the termination status of the child process. A child process may have terminated because of a signal, because of an exit call, or have stopped execution during **ptrace()**. Termination with **exit()**, which is normal completion, gives status 0. Other terminations give signal values as status (as defined in the article on **signal()**). The **0200** bit of the status code indicates that a core dump was produced. A status of **0177** indicates that the process is waiting for further action from **ptrace()**.

The high byte of the returned status is the low byte of the argument to the **exit()** system call.

If a parent process does not remain in existence long enough to **wait()** on a child process, the child process is adopted by process 1 (the initialization process).

Example

For an example of this system call, see the entry for **msgget()**.

See Also

_exit(), fork(), ksh, libc, ptrace(), signal(), sh, waitpid(), wait.h
POSIX Standard, §3.2.1

waitpid() — System Call (libc)

Wait for a process to terminate
#include <sys/types.h>
#include <sys/wait.h>
pid_t waitpid(*pid***,***status***,** *flags***)**
pid_t *pid***; int** ******status***,** *flags***;**

waitpid() waits until a given process terminates. *pid* identifies the child process whose termination is awaited. The value of *pid* sets the behavior of **waitpid()**, as follows:

pid>0 Wait for the termination of the child process whose identifier is *pid*.

pid=0 Wait for the termination of any child in the current process group.

pid=-1 Wait for the termination of any child process. In this mode, **waitpid()** behaves the same as the system call **wait()**.

pid<-1 Wait for termination of any child in the group given by **-***pid*.

status points to the place where you want **waitpid()** to write the termination status of *pid*.

flags is the logical OR of the following values:

WNOHANG
If *pid* has already terminated, write its termination status into *status*; but if *pid* has not yet terminated, do not wait for it to do so.

WUNTRACED
Report the status of every child process of *pid* that is stopped, and whose status has not been returned since it stopped.

By default, **waitpid()** returns the process identifier of the chid process whose status is being reported, or -1 if something went wrong. If *flags* includes **WNOHANG**, **waitpid()** returns zero if no status information is available.

See Also

libc, wait(), wait.h
POSIX Standard, §3.2.1

wall — Command

Send a message to all logged-in users
/etc/wall

wall types a message to every user currently logged into the COHERENT system, with the exception of the sender. It can be used to inform users of information of general interest; for example, that man has landed on the moon, or that the system is going down in 15 minutes.

wall reads the message to be broadcast from the standard input until end of file. When it sends the message, it prefaces it with the herald "Broadcast message ...", which includes an audible warning. Only the superuser should invoke **/etc/wall** (to override access protections of the target terminals).

Files

/etc/utmp — Current users file
/dev/tty*

See Also

commands, msg, who, write

Diagnostics

The message "Cannot send to *user* on *ttyname*" indicates that **wall** cannot write to the given *user*.

wc — Command

Count words, lines, and characters in text files
wc [-clw] [*file…***]**

wc counts words, lines, and characters in each *file*. If no *file* is given, **wc** uses the standard input. If more than one *file* is given, **wc** also prints a total for all of the files.

wc defines a *word* to be a string of characters surrounded by white space (blanks, tabs, or newlines). It defines the number of lines to be the number of newline characters in the file, plus one.

wc recognizes the following options:

-c Count only characters.

-l Count only lines.

-w Count only words.

The default action is to print all counts.

See Also

commands

welcome — System Administration

Welcome a new user
/etc/default/welcome

The command **login** normally displays the contents of file **$HOME/.lastlogin** when you log in. This file holds the date and time that you last logged into your COHERENT system.

If this file does not exist, **login** assumes that you are logging in for the first time, and executes the script **/etc/default/welcome**. This script displays information about COHERENT, to help welcome you to it and provide you with a "friendly" environment.

For information on what this file does, you should read it. If you wish, you can modify this file to suit the layout and special features of your system.

See Also

Administering COHERENT, login, newusr

whence — Command

List a command's type
whence [-v] *command ...*

The command **whence** is built into the Korn shell **ksh**. It lists the type for each *command*. Option **-v** lists function and alias values as well.

See Also

commands, ksh

whereis — Command

Locate source, binary, and manual files
whereis [-bmrsu] [-BMS *dir ...* **-f]** *name ...*

The command **whereis** locates source files, binary files (executables), and manual pages (documentation) that match a given *name*. Prior to searching, **whereis** strips *name* of any path information, extensions, and the **s.** prefix.

By default, **whereis** searches the following directories:

Sources	*Binaries*	*Manual Pages*
/usr/src/cmd	/bin	/usr/man/*
/usr/src/games	/usr/bin	
/usr/src/local	/usr/games	
/usr/src/alien	/usr/local	
/usr/include	/etc	
/usr/include/sys	/lib	
	/usr/lib	

Options

whereis recognizes the following command-line options:

-b Search only for binary files.

-B Use the directory list specified by *dir* instead of the default directory list for binary files.

-f Terminate the directory list introduced by options **-B**, **-M**, or **-S**, and treat any additional command-line arguments as file names to be searched for.

-m Search only for manual pages (documentation files).

-M Use the directory list specified by *dir* instead of the default directory list for manual pages.

-r Search recursively downward from the directories specified by *dir* or from the default directories. This option is useful when the searched directories contain sub-directories. By default, **whereis** searches only the directories specified or the default directories.

-s Search only for source files.

-S Use the directory list specified by *dir* instead of the default directory list for source files.

-u Search for "unusual" files. A file is said to be unusual if it does not have one entry for each of the three search categories.

Please note that if you use options **-B**, **-S**, or **-M**, you must use the **-f** option to terminate the directory list specified by *dir*.

Example

The following example finds all commands in directory **bin** that have either no corresponding source code in directory **src** or no corresponding documentation in directory **doc**:

```
whereis -u -M doc -S src -B bin -f bin/*
```

See Also

commands, find, qfind, which

Notes

whereis is copyright © 1980,1990 by The Regents of the University of California. All rights reserved.

whereis is distributed as a service to COHERENT customers, as is. It is not supported by Mark Williams Company. *Caveat utilitor.*

which — Command

Locate executable files
which *command ...*

which displays the full path name associated with *command*. It searches the directories named by environment variable **PATH** for the first executable that matches *command* and that you have permission to execute. If **which** can find no executable that matches your request, an error message is displayed.

Example

The following example displays the path names that correspond to commands **write**, **vi**, **myprog**, and **fsck**:

```
which write vi myprog fsck
```

See Also

commands, find, PATH, qfind, whereis

while — Command

Execute commands repeatedly
while *sequence1* **[do** *sequence2***] done**

The shell construct **while** controls a loop. It first executes the commands in *sequence1*. If the exit status is zero, the shell executes the commands in the optional *sequence2* and repeats the process until the exit status of *sequence1* is nonzero. Because the shell recognizes a reserved word only as the unquoted first word of a command, both **do** and **done** must occur unquoted at the start of a line or preceded by ';'.

The shell commands **break** and **continue** may be used to alter control flow within a **while** loop. The **until** construct has the same form as **while**, but the sense of the test is reversed.

The shell executes **while** directly.

See Also

break, commands, continue, ksh, sh, test, until

while — C Keyword

Introduce a loop
while(*condition***)**

while is a C keyword that introduces a conditional loop. *condition* is tested on reiteration of the loop, and the loop ends when *condition* is no longer satisfied. For example,

```
while (foo < 10)
```

introduces a loop that will continue until the variable **foo** is reset to ten or greater. Note that the statement

```
while (1)
```

will loop forever, unless interrupted by a **break**, **goto**, or **return** statement.

See Also

break, C keywords, continue, do, for
ANSI Standard, §6.6.5.1

who — Command

Print who is logged in
who [*file***] [am i]**

The command **who** prints the names of the users who are logged in to the system. For each user, **who** prints her name, terminal name, login date, and login time. The form **who am i** prints this information only about yourself.

If *file* is specified, **who** uses it instead of **/etc/utmp** to obtain information about who is logged in. This is useful, for example, with the file **/usr/adm/wtmp**, which contains a continuous record of logins, logouts and reboots. When *file* is specified, **who** displays logouts; otherwise, they are suppressed.

Files

/etc/utmp — To get user information

See Also

ac, commands, sa

wildcards — Definition

Wildcards are characters that, in some circumstances, can represent a range of ASCII characters. Another name for them is "metacharacters". The wildcards available under COHERENT are as follows:

? Match any one character.

* Match any number of characters, or no characters at all.

[ ] A set of characters enclosed between '[' and ']' match any one character of the set. Sets of characters may include ranges, such as **[a-z]** for all lower-case letters or **[0-9]** for all numerals.

[!] A set of characters enclosed between '[!' and ']' match any one character *except* one of the set. Sets of characters may include ranges, such as **[a-z]** for all lower-case letters or **[0-9]** for all numerals. For example, the command

```
ls [!abc]*
```

prints the names of all files *except those that begin with* **a**, **b**, *or* **c**.

**** Ignore the special meaning of a wildcard.

See Also

egrep, pattern, pnmatch(), Using COHERENT

write — Command

Converse with another user
write *user* **[** *tty* **]**

The COHERENT system provides several commands that allow users to communicate with each other. **write** allows two logged-in users to have an extended, interactive conversation.

write initiates a conversation with *user*. If *tty* is given, **write** looks for the *user* on that terminal; this is useful if a user is marked as being logged in on more than one device. Otherwise, **write** holds the conversation with the first instance of *user* found on any tty.

If found, **write** notifies *user* that you are beginning a conversation with him. All subsequent lines typed into **write** are forwarded to the *user*'s terminal, except lines beginning with '!', which are sent to the shell **sh**. Typing end of file (usually **<ctrl-D>**) terminates **write** and sends *user* the message "EOT" to tell him that communication has ended.

Two users typing lines to **write** at about the same time can cause extreme confusion, so users should agree on a protocol to limit when each is typing. The following protocol is suggested. One user initiates a **write** to another, and waits until the other user replies before beginning. The first user then types until he wishes a reply and suffixes "o" (over) to indicate he is through. The other user does the same, and the conversation alternates until one user wishes to terminate it. This user types "oo" (over and out). The other user replies in the same way, indicating he too is finished. Finally each of the users leave **write** by typing end-of-file (usually **<ctrl-D>**).

Any user may deny others the permission to **write** to his terminal by using the command **mesg**.

Files

/etc/utmp
/dev/*

See Also

commands, mail, mesg, msg, sh, wall, who

Notes

You should use **write** only for extended conversations. Use **msg** to send brief communications to a logged in user, and **mail** to communicate with a user who is not logged in. **wall** broadcasts a message to all logged in users.

write() — System Call (libc)

Write to a file
#include <unistd.h>
int write(*fd***,** *buffer***,** *n***)**
int *fd***; char** **buffer***; **int** n;*

write() writes *n* bytes of data, beginning at address *buffer*, into the file associated with the file descriptor *fd*. Writing begins at the current write position, as set by the last call to either **write()** or **lseek()**. **write()** advances the position of the file pointer by the number of characters written.

Example

For an example of how to use this function, see the entry for **open()**.

See Also

libc, unistd.h
POSIX Standard, §6.4.2

Diagnostics

write() returns -1 if an error occurred before the **write()** operation commenced, such as a bad file descriptor *fd* or invalid *buffer* pointer. Otherwise, it returns the number of bytes written. It should be considered an error if this number is not the same as *n*.

Notes

write() is a low-level call that passes data directly to COHERENT. Do not use it with the STDIO routines **fread()**, **fwrite()**, **fputs()**, or **fprintf()**.

wtmp — System Administration

File that records past login events
/usr/adm/wtmp

File **/usr/adm/wtmp** records every login event that has concluded — that is, the user has logged in and logged out again. You can comb this file to trace which user have logged onto your system, and when.

wtmp records each active login event as a record of type **utmp**, which is defined in header file **<utmp>**. For details, see the Lexicon entry **utmp.h**.

See Also

Adminstering COHERENT, utmp utmp.h