NAME

jobs, fg, bg, stop, notify – control process execution

SYNOPSIS

sh

jobs [–**p** | –**l**] [%*job_id* ...]
jobs –**x** *command* [*arguments*]
fg [%*job_id* . . .]
bg [%*job_id* . . .]
stop %*job_id* . . .
stop *pid* . . .

csh

jobs[–**l**]
fg [%*job_id*]
bg [%*job_id*] . . .
notify [%*job_id*] . . .
stop %*job_id* . . .
stop *pid* . . .

ksh

jobs [–**lnp**] [%*job_id* . . .]
fg [%*job_id* . . .]
bg [%*job_id* . . .]
stop %*job_id* . . .
stop *pid* . . .

DESCRIPTION

sh

When Job Control is enabled, the Bourne shell built-in **jobs** reports all jobs that are stopped or executing in the background. If %*job_id* is omitted, all jobs that are stopped or running in the background will be reported. The following options will modify/enhance the output of **jobs**:

–**l** Report the process group ID and working directory of the jobs.

–**p** Report only the process group ID of the jobs.

–**x** Replace any *job_id* found in *command* or *arguments* with the corresponding process group ID, and then execute *command* passing it *arguments*.

When the shell is invoked as **jsh**, Job Control is enabled in addition to all of the functionality described previously for **sh**. Typically Job Control is enabled for the interactive shell only. Non-interactive shells typically do not benefit from the added functionality of Job Control.

With Job Control enabled every command or pipeline the user enters at the terminal is called a *job_id*. All jobs exist in one of the following states: foreground, background or stopped. These terms are defined as follows: 1) a job in the foreground has read and write access to the controlling terminal; 2) a job in the background is denied read access and has conditional write access to the controlling terminal (see **stty**(1)); 3) a stopped job is a job that has been placed in a suspended state, usually as a result of a **SIGTSTP** signal (see **signal**(5)).

Every job that the shell starts is assigned a positive integer, called a *job_id number* which is tracked by the shell and will be used as an identifier to indicate a specific job. Additionally the shell keeps track of the *current* and *previous* jobs. The *current job* is the most recent job to be started or restarted. The *previous job* is the first non-current job.

The acceptable syntax for a Job Identifier is of the form:

%*job_id*

where, *job_id* may be specified in any of the following formats:

% or **+**	for the current job
–	for the previous job
?<*string*>	specify the job for which the command line uniquely contains *string*.
n	for job number *n*, where *n* is a job number
pref	where *pref* is a unique prefix of the command name (for example, if the command **ls –l name** were running in the background, it could be referred to as **%ls**); *pref* cannot contain blanks unless it is quoted.

When Job Control is enabled, **fg** resumes the execution of a stopped job in the foreground, also moves an executing background job into the foreground. If %*job_id* is omitted the current job is assumed.

When Job Control is enabled, **bg** resumes the execution of a stopped job in the background. If %*job_id* is omitted the current job is assumed.

stop stops the execution of a background job(s) by using its *job_id*, or of any process by using its *pid*; see **ps**(1).

csh

The C shell built-in, **jobs**, without an argument, lists the active jobs under job control.

–l List process IDs, in addition to the normal information.

The shell associates a numbered *job_id* with each command sequence to keep track of those commands that are running in the background or have been stopped with **TSTP** signals (typically CTRL-Z). When a command or command sequence (semicolon separated list) is started in the background using the **&** metacharacter, the shell displays a line with the job number in brackets and a list of associated process numbers:

[1] 1234

To see the current list of jobs, use the **jobs** built-in command. The job most recently stopped (or put into the background if none are stopped) is referred to as the *current* job and is indicated with a \`**+**´. The previous job is indicated with a \`–´; when the current job is terminated or moved to the foreground, this job takes its place (becomes the new current job).

To manipulate jobs, refer to the **bg**, **fg**, **kill**, **stop**, and % built-in commands.

A reference to a job begins with a \`%´. By itself, the percent-sign refers to the current job.

% %+ %%	The current job.
%–	The previous job.
%*j*	Refer to job *j* as in: \`**kill –9** %*j*´. *j* can be a job number, or a string that

uniquely specifies the command line by which it was started; `**fg %vi**´ might bring a stopped **vi** job to the foreground, for instance.

%?*string* Specify the job for which the command line uniquely contains *string*.

A job running in the background stops when it attempts to read from the terminal. Background jobs can normally produce output, but this can be suppressed using the `**stty tostop**´ command.

fg brings the current or specified *job_id* into the foreground.

bg runs the current or specified jobs in the background.

stop stops the execution of a background job(s) by using its *job_id*, or of any process by using its *pid*; see **ps**(1).

notify will notify the user asynchronously when the status of the current job or specified jobs changes.

ksh

jobs displays the status of the jobs that were started in the current shell environment. When **jobs** reports the termination status of a job, the shell removes its process ID from the list of those "known in the current shell execution environment."

job_id specifies the jobs for which the status is to be displayed. If no *job_id* is given, the status information for all jobs will be displayed.

The following options will modify/enhance the output of **jobs**:

–l (The letter ell.) Provide more information about each job listed. This information includes the job number, current job, process group ID, state and the command that formed the job.

–n Display only jobs that have stopped or exited since last notified.

–p Displays only the process IDs for the process group leaders of the selected jobs.

By default, **jobs** displays the status of all the stopped jobs, running background jobs, and all jobs whose status has changed and have not been reported by the shell.

If the **monitor** option of the **set** command is turned on, an interactive shell associates a **job** with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line which looks like:

[1] 1234

indicating that the **job**, which was started asynchronously, was job number 1 and had one (top-level) process, whose process id was 1234.

If you are running a job and wish to do something else you may hit the key **^Z** (CTRL-Z) which sends a **STOP** signal to the current job. The shell will then normally indicate that the job has been `**Stopped**´ (see **OUTPUT** below), and print another prompt. You can then manipulate the state of this job, putting it in the background with the **bg** command, or run some other commands and then eventually bring the job back into the foreground with the foreground command **fg**. A **^Z** takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed.

There are several ways to refer to jobs in the shell. A job can be referred to by the process id of any process of the job or by one of the following:

%*number*	The job with the given number.
%*string*	Any job whose command line begins with *string*; works only in the interactive mode when the history file is active.
%**?***string*	Any job whose command line contains *string*; works only in the interactive mode when the history file is active.
%%	Current job.
%**+**	Equivalent to %%.
%–	Previous job.

The shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work. When the monitor mode is on, each background job that completes triggers any trap set for **CHLD**. When you try to leave the shell while jobs are running or stopped, you will be warned that `You have stopped (running) jobs.´ You may use the **jobs** command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

fg will move a background job from the current environment into the foreground. Using **fg** to place a job in the foreground will remove its process ID from the list of those "known in the current shell execution environment." The **fg** command is available only on systems that support job control. If *job_id* is not specified, the current job is brought into the foreground.

bg resumes suspended jobs from the current environment by running them as background jobs. If the job specified by *job_id* is already a running background job, **bg** has no effect and will exit successfully. Using **bg** to place a job into the background causes its process ID to become ‘‘known in the current shell execution environment’’, as if it had been started as an asynchronous list. The **bg** command is available only on systems that support job control. If *job_id* is not specified, the current job is placed in the background.

stop stops the execution of a background job(s) by using its *job_id*, or of any process by using its *pid*; see **ps**(1).

OUTPUT

If the **–p** option is specified, the output consists of one line for each process ID:

"%d\n", <*"process ID"*>

Otherwise, if the **–l** option is not specified, the output is a series of lines of the form:

"[%d] %c %s %s\n", <*job-number*>, <*current*>, <*state*>, <*command*>

where the fields are as follows:

<*current*> The character **+** identifies the job that would be used as a default for the **fg** or **bg** commands; this job can also be specified using the *job_id* %**+** or %% **.** The character – identifies the job that would become the default if the current default job were to exit; this job can also be specified using the *job_id* %– **.** For other jobs, this field is a space character. At most one job can be identified with **+** and at most one job can be identified

with –. If there is any suspended job, then the current job will be a suspended job. If there are at least two suspended jobs, then the previous job will also be a suspended job.

<*job-number*> A number that can be used to identify the process group to the **wait**, **fg**, **bg,** and **kill** utilities. Using these utilities, the job can be identified by prefixing the job number with **%**.

<*state*> One of the following strings (in the POSIX Locale):

Running Indicates that the job has not been suspended by a signal and has not exited.

Done Indicates that the job completed and returned exit status zero.

Done(*code*)
Indicates that the job completed normally and that it exited with the specified non-zero exit status, *code*, expressed as a decimal number.

Stopped

Stopped (SIGTSTP)
Indicates that the job was suspended by the **SIGTSTP** signal.

Stopped (SIGSTOP)
Indicates that the job was suspended by the **SIGSTOP** signal.

Stopped (SIGTTIN)
Indicates that the job was suspended by the **SIGTTIN** signal.

Stopped (SIGTTOU)
Indicates that the job was suspended by the **SIGTTOU** signal.

The implementation may substitute the string **Suspended** in place of **Stopped**. If the job was terminated by a signal, the format of **state** is unspecified, but it will be visibly distinct from all of the other **state** formats shown here and will indicate the name or description of the signal causing the termination.

<*command*> The associated command that was given to the shell.

If the –**l** option is specified, a field containing the process group ID is inserted before the **state** field. Also, more processes in a process group may be output on separate lines, using only the process ID and **command** fields.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **jobs**, **fg**, and **bg**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned for **jobs**, **fg**, and **bg**:

0 Successful completion.

>0 An error occurred.

SEE ALSO

csh(1), **kill**(1), **ksh**(1), **ps**(1), **sh**(1), **stop**(1), **shell_builtins**(1), **stty**(1), **wait**(1), **environ**(5), **signal**(5)