IEEE P1003.2a Draft 8 – December 1991

5.15 `jobs` — Display status of jobs in the current session

5.15.1 Synopsis

`jobs [ –l | –p ] [`*job_id* ...`]`

5.15.2 Description

The `jobs` utility shall display the status of jobs that were started in the current shell environment; see 3.12.

When `jobs` reports the termination status of a job, the shell shall remove its pro- 7
cess ID from the list of those ‘‘known in the current shell execution environment’’;
see 3.9.3.1.

5.15.3 Options

The `jobs` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

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

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

By default, the `jobs` utility shall display the status of all stopped jobs, running 7
background jobs, and all jobs whose status has changed and have not been 7
reported by the shell. 7

5.15.4 Operands

The following operand shall be supported by the implementation:

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 shall be displayed. The format of *job_id* is described in 2.2.2.206.

5.15.5 External Influences

5.15.5.1 Standard Input

None.

5.15.5.2 Input Files

None.

5.15.5.3 Environment Variables

The following environment variables shall affect the execution of `jobs`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

5.15.5.4 Asynchronous Events

Default.

5.15.6 External Effects

5.15.6.1 Standard Output

If the `−p` option is specified, the output shall consist of one line for each process ID:

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

Otherwise, if the `−l` option is not specified, the output shall be a series of lines of the form:

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

where the fields shall be as follows:

<current> The character `+` identifies the job that would be used as a default for the `fg` or `bg` utilities; 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>`. 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 shall be a suspended job. If there are at least two suspended jobs, then the previous job shall 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 nonzero exit status, *code*, expressed as a decimal number.

  `Stopped`
  `Stopped (SIGTSTP)`
  Either 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 shall be visibly distinct from all of the other <*state*> formats shown here and shall 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 shall be 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.

5.15.6.2 Standard Error

Used only for diagnostic messages.

5.15.6.3 Output Files

None.

5.15.7 Extended Description

None.

5.15.8 Exit Status

The `jobs` utility shall exit with one of the following values:

0 Successful completion.

>0 An error occurred.

5.15.9 Consequences of Errors

Default.

5.15.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

Both `%%` and `%+` are used to refer to the current job. Both forms are of equal validity—the `%%` mirroring `$$` and `%+` mirroring the output of `jobs`. Both forms reflect historical practice of the KornShell and the C-Shell with job control.

The `−p` option is historical practice in the KornShell and the System V `jsh`. It is the only way to find out the process group of a job because different implementations have different strategies for defining the process group of the job. Usage such as `$(jobs -p)` provides a way of referring to the process group of the job in an implementation-independent way.

The `jobs` utility will not work as expected when it is operating in its own utility execution environment because that environment will have no applicable jobs to manipulate. See the rationale for `bg`. For this reason, `jobs` is generally implemented as a shell regular built-in.

History of Decisions Made

The extensions to the shell specified in POSIX.2 and POSIX.2a have mostly been based on features provided by the KornShell. The job control features provided by `bg`, `fg`, and `jobs` are also based on the KornShell. The developers of the standard examined the characteristics of the C-Shell versions of these utilities and found that differences exist. Despite widespread use of the C-Shell, the KornShell versions were selected for this standard to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command-line editing features).

The output for terminated jobs is left unspecified to accommodate various historical systems. The following formats have been witnessed:

(1) `Killed(`*signal name* `)`

(2) *signal name*

(3) *signal name* `(coredump)`

(4) *signal description* `- core dumped`

Most users should be able to figure these out, although it means that applications will have trouble parsing them.

The calculation of job IDs was not described since this would suggest an implementation, which may impose unnecessary restrictions.

In an earlier draft, a `–n` option was included to ''Display the status of jobs that have changed, exited, or stopped since the last status report.'' It was removed because the shell always writes any changed status of jobs before each prompt. Thus, `jobs –n` is a no-op.

5.16 `man` — Display system documentation

5.16.1 Synopsis

`man [−k]` *name* ...

5.16.2 Description

The `man` utility shall write information about each of the *name* operands. If *name* is the name of a standard utility, `man` shall at a minimum write a message describing the syntax used by the standard utility, its options, and operands. If more information is available, the `man` utility shall provide it in an implementation-defined manner.

An implementation may provide information for values of *name* other than the standard utilities. Standard utilities that are listed as optional and that are not supported by the implementation either shall cause a brief message indicating that fact to be displayed or shall cause a full display of information as described previously.

5.16.3 Options

The `man` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following option shall be supported by the implementation:

`−k` Interpret *name* operands as keywords to be used in searching a utilities summary database that contains a brief ''purpose'' entry for each standard utility and write lines from the summary database that match any of the keywords. The keyword search shall produce results that are the equivalent of the output of the following command:

```
grep -Ei '
name
name
...
' summary-database
```

(This assumes that the *summary-database* is a text file with a 7
single entry per line; this organization is not required and the 7
example using `grep −Ei` is merely illustrative of the type of 7
search intended.) The ''purpose'' entry to be included in the data- 7
base shall consist of a terse description of the purpose of the 7
utility. 7