IEEE P1003.2a Draft 8 – December 1991

2.2.2 General Terms

⇒ **2.2.2 General Terms.** *Modify the following two subclauses of 2.2.2, General Terms, to add the synonym shown for each:*

2.2.2.10 background process group [background job]: Any process group, other than a foreground process group, that is a member of a session that has established a connection with a controlling terminal. [POSIX.1 {8}]

2.2.2.73 foreground process group [foreground job]: A process group whose member processes have certain privileges, denied to processes in background process groups, when accessing their controlling terminal.

Each session that has established a connection with a controlling terminal has exactly one process group of the session as the foreground process group of that controlling terminal. See POSIX.1 {8}7.1.1.4. [POSIX.1 {8}]

⇒ **2.2.2 General Terms.** *Modify the contents of subclause 2.2.2, General Terms, to add the following definitions in the correct sorted order [disregarding the subclause numbers shown here].*

2.2.2.201 alert: To cause the user's terminal to give some audible or visual indication that an error or some other event has occurred.

When the standard output is directed to a terminal device, the method for alerting the terminal user is unspecified. When the standard output is not directed to a terminal device, the alert shall be accomplished by writing the `<alert>` character to standard output (unless the utility description indicates that the use of standard output produces undefined results in this case).

2.2.2.202 block-mode terminal: A terminal device operating in a mode incapable of the character-at-a-time input and output operations described by some of the standard utilities.

See 2.14.

2.2.2.203 cursor position: The line and column position on the screen denoted by the terminal's cursor.

2.2.2.204 display: To output to the user's terminal.

If the output is not directed to a terminal, the results are undefined.

2.2.2.205 job: A set of processes, comprising a shell pipeline, and any processes descended from it, that are all in the same process group.

See the definition of *pipeline* in 3.9.2.

2.2.2.206 job control job ID: A handle that is used to refer to a job.

The job control job ID can be any of the forms shown in Table 2-100.

Table 2-100 – Job Control Job ID Formats

Job Control Job ID	Meaning
%%	Current job
%+	Current job
%−	Previous job
%n	Job number *n*
%string	Job whose command begins with *string*
%?string	Job whose command contains *string*

2.2.2.207 mount point: Either the root directory or a directory for which the *st_rdev* field of the POSIX.1 {8} *struct stat* differs from that of its parent directory.

2.2.2.208 refresh: To ensure that the information on the user's terminal screen is up-to-date.

2.2.2.209 screen: A rectangular region of columns and lines on a terminal display.

A screen may be a portion of a physical display device or may occupy the entire physical area of the display device.

2.2.2.210 scroll: To move the representation of data vertically or horizontally relative to the terminal screen.

There are two types of scrolling:

(1) The cursor moves with the data

(2) The cursor remains stationary while the data moves

2.2.2.211 suspended job: A job that has received a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal that caused the process group to stop.

A suspended job is a background job, but a background job is not necessarily a suspended job.

2.2.2.212 system scheduling priority: A number used as advice to the system to alter process scheduling priorities.

Raising the value should give a process additional preference when scheduling a process to run. Lowering the value should reduce the preference and make a process less likely to run. Typically, a process with higher system scheduling priority will run to completion more quickly than an equivalent process with lower system scheduling priority. A scheduling priority of zero specifies the default policy of the system.

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

This standard assigns precise requirements for the terms *display* and *write,* unlike some historical user manuals. (The term *write* has been moved into the POSIX.2 base standard as of Draft 11.) Some historical systems have chosen to implement certain utilities without using the traditional UNIX system file descriptor model. For example, the `vi` editor might employ direct screen memory updates on a personal computer, rather than a *write* () system call. An instance of user prompting might appear in a little ‘‘pop-up’’ dialog box, rather than with standard error. When the standard uses the term *display,* the method of outputting to the terminal is unspecified; many historical implementations use *termcap* or *terminfo* , but this is not a requirement. The term *write* is used when the standard mandates that a file descriptor be used and that the output can be redirected. However, it is assumed that when the writing is directly to the terminal (it has not been redirected elsewhere), there is no practical way for a user or test suite to determine whether a file descriptor is actually being used or not. Therefore, the use of a file descriptor is mandated only for the redirection case and the implementation is free to use any method when the output is not redirected. In previous drafts, there were the terms *print* and *type,* which attempted to exert a finer degree of specification on output and input differences for terminals versus other types of files. These verbs were discarded when it was realized that there is no practical or useful way to determine how I/O is accomplished when it is not redirected. *Read* and *write* are used almost exclusively, with the very few exceptions of those utilities where output redirection need not be supported: `tabs`, `talk`, `tput`, and `vi`.

The definition of *block-mode terminal* refers to a mode of operation because some terminals are capable of full character-at-a-time operation only in certain hardware- or software-selected modes. Terminals that can support all the features of this standard in one mode should not be excluded if there are some cases where they may be set up with other characteristics. There is no implication that the implementation is required to sense changes in these modes during operation of the utilities.

The definition of *cursor* is assumed to be a standard in computer science.

The definition of *scroll* omits the situation where the cursor moves while the data remains stationary; this is considered to be a cursor movement, not scrolling.

The definition of *system scheduling priority* is not intended to suggest that all 7
processes in a system have priorities that are comparable. Scheduling policy 7

extensions such as the realtime priorities in POSIX.4 make the notion of a single underlying priority for all scheduling policies problematic. Some systems may implement the `nice`-related features to affect all processes on the system, others to affect just the general time-sharing activities implied by POSIX.2, and others may have no effect at all. Because of the use of ''implementation defined'' in `nice` and `renice`, a wide range of implementation strategies are possible.

2.3 Built-in Utilities

⇒ **2.3 Built-in Utilities.** *Modify the contents of clause 2.3, Built-in Utilities, to add the following utilities to the table of regular built-ins.*

`alias`	`fc`	`jobs`	`unalias`
`bg`	`fg`	`newgrp`	

Built-in Utilities Rationale. *(This subclause is not a part of P1003.2a)*

All of the job-control related utilities are eligible for built-in status because that is the model most historical implementations have used.