Signal names

Signals cannot carry information directly, which limits their usefulness as a general inter-process communication mechanism. However, each type of signal is given a mnemonic name - SIGINT is an example - which indicates the purpose for which the signal is normally used. Signal names are defined in the standard header file <signal.h> with the pre-processor directive #define. As you might expect, these names just stand for small, positive integers. For example, SIGINT is usually defined as:

```
#define    SIGINT 2    /* interrupt   (rubout)   */
```

Most of the signal types provided by UNIX are intended for use by the kernel, although a few are provided to be sent from process to process. The complete list of standard signals, as described in the XSI, and their meanings follows. The list of signals has been put in alphabetical order for ease of reference. On first reading you can safely skip this list.

SIGABRT ***Process abort signal.*** This is sent by a call to the abort function by IK the current process. SIGABRT will result in what the XSI describes as abnormal termination. The actual effect on UNIX implementations is a core dump, indicated by the message Quit - core dumped, where the process image is dumped into a disk file for debugging purposes

SIGALRM ***Alarm clock.*** This is sent by the kernel to a process after a timer has expired. All processes can set at least three interval timers. The first of these can be set to measure real elapsed time. The timer is actually set by the process itself using the alarm system call (or setting the first parameter to the more exotic setitimer system call to ITIMER_REAL).

SIGBUS ***Bus error.*** This error is sent if a hardware fault is detected. A bus error is implementation defined and, like SIGABRT, causes an abnormal termination.

SIGCHLD ***Child process terminated or stopped***. Whenever a child process terminates or stops, the kernel will inform its parent by sending it SIGCHLD, the death of child signal. By default the parent will ignore the signal, and therefore a parent must explicitly catch the SIGCHLD signal if it wishes to know about any child processes, which have completed.

SIGCONT ***Continue executing if stopped.*** This is a job control signal which will continue the process if it is stopped; otherwise the process will ignore the signal. It is the inverse of SIGSTOP.

SIGFPE ***Floating-point exception***. This is sent by the kernel when a floating point error occurs (such as overflow or underflow). It causes abnormal termination.

SIGHUP The ***hangup*** signal. This is sent by the kernel to all processes attached to a **control terminal** when that terminal is disconnected. (Normally, a process group's control terminal will be the user's own terminal, although this is not always the case. It is also sent to all members of a session when the session leader, normally a shell process, exits, providing the session is associated with a control terminal. This ensures background processes are terminated when a user logs out, unless explicit arrangements have been made otherwise.

SIGILL ***Illegal instruction***. This is sent by the operating system when a process attempts to execute an illegal instruction. It is possible that the program has corrupted its own code, but this is unlikely. Other causes, such as an attempt to execute floating-point

instructions without the right hardware support, are more probable. SIGILL results in abnormal termination.

SIGINT ***Interrupt***. This is sent by the kernel to all processes associated with a terminal session when a user hits the interrupt key. It is the conventional way of halting a running program.

SIGKILL ***Kill.*** This is a rather special signal that is sent from one process to another to terminate the receiver. It is also occasionally sent by the system (that is, during system shutdown). SIGKILL is one of only two signals that cannot be ignored or 'caught' (that is, handled via a user-defined interrupt routine).

SIGPIPE ***Write on a pipe or socket when recipient has terminated.*** A pipe and a socket are other inter-process communication mechanisms

SIGPOLL ***Pollable event***. This signal is generated by the kernel when an open file descriptor is ready for input or output. However, an easier way to implement polling is to use the select system call.

SIGPROF ***Profiling time expired***. As mentioned above for SIGALRM, all processes can set at least three interval timers. The second of these timers can be set to measure the time the process is executing in user and system mode. A SIGPROF signal is generated when this timer expires and can therefore be used by interpreters to profile the execution of a program. The timer is set by setting the first parameter of the function setitimer to ITIMER_PROF.)

SIGQUIT ***Quit.*** Very similar to SIGINT, this is sent by the kernel when the user hits the quit key associated with his or her terminal. The default value for the quit key is ASCII FS or Ctrl-\. Unlike SIGINT, this signal will cause an abnormal termination and therefore a core dump.

SIGSEGV ***Invalid memory reference.*** SEGV stands for segmentation violation and is generated when a process tries to access an invalid memory address. SIGSEGV results in abnormal termination.

SIGSTOP ***Stop executing***. This is a job control signal which will stop the process. It is similar to SIGKILL in that it cannot be caught or ignored.

SIGSYS ***Invalid system call***. This is sent by the kernel if a process attempts to execute a machine instruction which is not a system call. This is another signal that results in abnormal termination.

SIGTERM ***Software termination signal***. By convention, it is used to terminate a process. The programmer can use this signal to allow a process some tidying up time before sending SIGKILL.

SIGTRAP ***Trace trap***. This is a special signal used by debuggers such as **sbd** and **adb**, in conjunction with the ptrace system call. Because of its specialized nature, we will not discuss it further. By default, SIGTRAP results in abnormal termination.

SIGTSTP ***Terminal stop signal***. This signal is generated by the user typing the suspend key (normally *Ctrl-Z)*. SIGTSTP is similar to SIGSTOP; however, it can be caught and ignored.

SIGTTIN ***Background process attempting read***. Whenever a process is executing in the background and attempts to read from its controlling terminal the SIGTTIN signal will be sent. The default action is to stop the process.

SIGTTOU ***Background process attempting write***. This is similar to SIGTTIN except that this signal will be generated when the background process attempts to write to the controlling terminal. Again, the default action is to stop the process.

SIGURG ***High bandwidth data is available at a socket***. This signal tells a process that urgent or out of band data has been received on a network connection.

SIGUSR1 and SIGUSR2 Like SIGTERM, these are never sent by the kernel and may be used for whatever purpose a user wishes.

SIGVTALRM ***Virtual timer expired***. As mentioned for SIGALRM and SIGPROF, all processes have at least three interval timers. The last of these timers can be set to measure the time the process is executing in user mode. (The timer is set by setting the first parameter of the function setitimer to ITIMER_VIRTUAL.)

SIGXCPU ***CPU time limit exceeded.*** This signal is generated if a process exceeds its maximum CPU time limit. The default action is an abnormal termination.

SIGXFSZ ***File size limit exceeded.*** This signal is generated if a process exceeds its maximum file size limit. The default action is an abnormal termination.

There are a number of other signals that you may encounter which are implementations dependent and lie outside the territory of XSI. Again, most of these are used by the kernel to indicate error conditions, for example: SIGEMT (emulator trap) often indicates an implementation-defined hardware fault.