SIGNALS AND DEAMON PROCESSES

6 . Introduction

1. Signals are triggered by events and are posted on a process to notify it that some thing has happened and requires some action.

2. Signals can be generated from a process, a user, or the UNIX kernel.

3. Example:-

    a. A process performs a divide by zero or dereferences a NULL pointer.
    b. A user hits <Delete> or <Ctrl-C> key at the keyboard.

4. A parent and child processes can send signals to each other for process synchronization.

5. Thus, signals are the software version of hardware interrupts.

6. Signals are defined as integer flags in the <signal.h> header file.

7. The following table 6.1 lists the POSIX – defined signals found in most UNIX systems.

Name	Description	Default action
SIGALRM	timer expired (alarm)	terminate
SIGABRT	abnormal termination (abort)	terminate+core
SIGFPE	arithmetic exception	terminate+core
SIGHUP	controlling terminal hangup	terminate
SIGILL	illegal machine instruction	terminate+core
SIGINT	terminal interrupt character <delete> or <ctrl-c> keys	terminate
SIGKILL	kill a process, kill -9 <pid> command.	terminate
SIGPIPE	write to pipe with no readers	terminate
SIGQUIT	terminal quit character	terminate+core
SIGSEGV	segmentation fault - invalid memory reference	terminate+core
SIGTERM	terminate process, kill <pid> command	terminate

SIGUSR1	user-defined signal	terminate
SIGUSR2	user-defined signal	terminate
SIGCHLD	change in status of child	ignore
SIGCONT	continue stopped process	continue/ignore
SIGSTOP	stop a process execution	stop process
SIGTTIN	stop a background process when it read from its control tty	stop process
SIGTSTP	stop a process execution by the ctrl-z key	stop process
SIGTTOU	stop a background process when it writes to its control tty	stop process

8. When a signal is sent to a process, it is *pending* on the process to handle it.

9. The process can react to signals in one of the three ways.

    a. Accept the default action of the signal – most signals terminate the process.
    b. Ignore the signal.
    c. Invoke a user defined function – The function is called *signal hander routine* and the signal is said to be *caught* when the function is called. If the function finishes execution without terminating the process, the process will continue execution from the point it was interrupted by the signal.

10. Most signals can be caught or ignored except the SIGKILL and SIGSTOP signals.

11. A companion signal to SIGSTOP is SIGCONT, which resumes a process execution after it has been stopped, both SIGSTOP and SIGCONT signals are used for job control in UNIX.

12. A process is allowed to ignore certain signals so that it is not interrupted while doing certain mission – critical work.

13. Example:- A DBMS process updating a database file should not be interrupted until it is finished, else database file will be corrupted, it should restore signal handling actions for signals when finished mission – critical work.

14. Because signals are generated asynchronously to a process, a process may specify a per signal handler function, these function would then be called when their corresponding signals are caught.

15. A common practice of a signal handler function is to clean up a process work environment, such as closing all input – output files, before terminating gracefully.