Log Book Week 10

Jose Devian Hibono
1706039603
System Programming - C

This week's material is about Scripting and Signals. First I watched the asynchronous lecture and did the worksheet using the resources on the internet. Here are the resources I gathered, I tried to write my own understandings in this log book.

Signals

Signals are software interrupts used for interprocess communication in Unix and Unix-like operating systems such as Linux. One process can send a signal and interrupt another process signal is a small int typically < 32. One process may transmit a signal to another process (if it has the necessary permissions). Each signal, beginning sequentially from 1, is specified as a specific small) integer.

The operating system interrupts the regular flow of the execution of the operation when a signal is transmitted to a process and provides the message. If a way to treat the specific signal has already been registered by the process, this routine is executed, otherwise the default signal handler is executed by the method.

Here a few examples of signals in Unix Systems

Signal	Number	Description
SIGHUP	1	The HUP signal is sent to a process when its controlling terminal is closed. It was originally designed to notify the process of a serial line drop (HUP stands for "Hang Up"). In modern systems, this signal usually means that the controlling pseudo or virtual terminal has been closed.
SIGINT	2	The INT signal is sent to a process by its controlling terminal when a user wishes to interrupt the process. This signal is typically initiated by pressing Control-C, but on some systems, the "delete" character or "break" key can be used.

SIGQUIT	3	The QUIT signal is sent to a process by its controlling terminal when the user requests that the process perform a core dump.
SIGILL	4	Illegal instruction. The ILL signal is sent to a process when it attempts to execute a malformed, unknown, or privileged instruction.
SIGTRAP	5	Trace trap. The TRAP signal is sent to a process when a condition arises that a debugger is tracing — for example, when a particular function is executed, or when a particular variable changes value.
SIGABRT, SIGIOT	6	Abort process. ABRT is usually sent by the process itself, when it calls the abort() system call to signal an abnormal termination, but it can be sent from any process like any other signal. SIGIOT is a synonym for SIGABRT. (IOT stands for input/output trap, a signal which originated on the PDP-11.)
SIGBUS	7	The BUS signal is sent to a process when it causes a bus error, such as an incorrect memory access alignment or non-existent physical address. In Linux, this signal maps to SIGUNUSED, because memory access errors of this kind are not possible.
SIGFPE	8	Floating point exception. The FPE signal is sent to a process when it executes an erroneous arithmetic operation, such as division by zero.
SIGKILL	9	Forcefully terminate a process. Along with STOP, this is one of two signals which cannot be intercepted, ignored, or handled by the process itself.

Signal Generation

There was a hardware exception, indicating the hardware observed a failure state that the kernel was told of. One of the special terminal characters that produce signals (e.g. ctrl+z) was typed by the operator. A machine incident happened. For instance, on a file descriptor, input became available, the terminal window was resized, a timer went off the process's time limit was extended, or a child of this process was terminated.

Finding the signals available on your system

Signals are defined in the system library signal.h. To view the signals used by your operating system, open a terminal and run man signal or man 7 signal.

Trapping Signals

During the execution of a shell program, when you press the Ctrl+C or Split key on your terminal, the program is usually automatically terminated and your command prompt returns. Maybe this is not necessarily desirable. You can end up for example, leaving a lot of temporary files that will not be cleaned up.

Trapping these signals is quite easy, and the trap command has the following syntax –

```
$ trap commands signals
```

References

Computer Hope. (2019). *Linux Signals*. Accessed at November 24, 2020 from https://www.computerhope.com/unix/signals.htm

TutorialsPoint. (n.d.) *Unix / Linux - Signals and Traps.* Accessed at November 24, 2020 from https://www.tutorialspoint.com/unix/unix-signals-traps.htm