Joel Schellinger
CPE-2600
LB10
Signals

Signal Research

Signal Disposition

Signal disposition is what determines how a process will behave when the signal is delivered

Signal Handlers

A signal handler is a defined function that is automatically invoked when the signal is delivered. It is used to do something other than the default action of the signal.

The Five Signal Dispositions

The five signal dispositions are as follows:

- Term
- Ign
- Core
- Stop
- Cont

Sending Signals

There are many ways that a signal can be sent programmatically. The simplest way to do this is using the raise method. Reading section three of the manual on the raise() function, information about using raise() can be found.

Raise sends a signal to the calling process or thread. This signal is the parameter int sig. The man page for SIGNAL(7) shows a list of signals. Below is an example of a library call that can be made to send a signal to a process:

```c
#include <signal.h>
int main() {
    raise(SIGUSR1);
    return 0;
}
```

You can also send signals to programs by using the kill command on the command line. With kill you can choose if you would like to use the default signal (SIGTERM) which terminates the program. By adding the -<Signal Name> you can choose the signal that you send to the program. Finally, when using kill you must specify the pid of the process that you want to send the signal.

Common Signals

- SIGINT
    - Interrupt from keyboard.
    - Terminates the process by default.
    - Can be overridden by a signal handler.
    - Many programs use CTRL+C to copy but by default this sends SIGINT to the process. It's important to be able to override this signal as the user will not want the process to end when they try to copy.
- SIGTERM
    - The termination signal.
    - Terminates the process by default.
    - Can be overridden.
    - SIGTERM is the signal that is usually sent when you close a process by using the "X" button. A time you would want to override this would be when you want to save any work done by the process.
    - In simple terms SIGTERM is the polite way to ask a process to terminate. However, the program might be smart and know that it might not be best to terminate at this time.
- SIGUSR1
    - A user defined signal.
    - Terminates the process by default.
    - Can be overridden by a signal handler.
    - Could be used to inform a process of some action done by another process.
    - It is imperative that a program can override this signal. Since the signal is user-defined it's important that the user is able to define the default action of the signal.
- SIGKILL
    - Kill signal
    - Terminates the process.
    - Cannot be overridden.
    - It is important that SIGKILL cannot be overridden as its only purpose is to kill processes.
    - If a program is not working as expected it is important to always be able to kill the program.
- SIGSTOP
    - Stops (pauses) a process.
    - Stops the process by default.
    - Cannot be overridden.
    - It is important that SIGSTOP cannot be overridden as its only purpose is to stop processes.
    - If a program is not working as expected it is important to always be able to stop the program.

Working With a Signal Handler

Sending SIGINT

There are three ways to send SIGINT to a process. One way is to use the kill command from the command line. Another is to use kill() from within a program. The final way is to press CTRL+C while running the process. When this is done the process receives SIGINT.

When sending SIGINT to the signal_handler program the current function just prints a statement then calls exit. You can modify the code so that it does not exit by removing the call to the exit function. After removing the exit function call, CTRL+C will no longer terminate the program. One trick to killing the process is by running top and finding the pid. Once you have found the pid you can send sigkill by running “kill -SIGKILL <pid>” from the command line. You can also edit the signal handler function so that it will do what may need to be done before exiting (e.g. saving files, freeing memory, etc.) and then using the kill() function send SIGKILL to the process.