Aidan Conlin
CPE 2600 121
Lab 10: Signals
12/11/2024

Part 1: Signal Research

1. What is a signal disposition?
A signal disposition defines how a process responds to a specific signal. Each signal has a default action (disposition) assigned by the operating system. The disposition can be one of:

1. **Ignore the signal.**
2. **Terminate the process.**
3. **Terminate the process and generate a core dump.**
4. **Stop the process.**
5. **Resume the process.** A process can override the default disposition for most signals by installing a custom signal handler.

2. What is a signal handler? What is it used for?
A signal handler is a function defined in a program to handle specific signals sent to the process. It is executed when the process receives a signal that the handler is registered for.

Purpose:

- Allows processes to perform specific actions in response to signals.
- Examples: Cleaning up resources, saving state before termination, or logging events.

3. Name and describe each of the five (5) default dispositions:

1. **Ignore**: The process ignores the signal and continues execution as if nothing happened. (E.g., SIGCHLD by default is ignored.)
2. **Terminate**: The process is terminated. (E.g., SIGTERM terminates gracefully.)
3. **Terminate with core dump**: The process terminates, and a core dump is created for debugging. (E.g., SIGSEGV.)
4. **Stop**: The process is paused and can only be resumed with a SIGCONT signal. (E.g., SIGSTOP.)
5. **Resume**: The process is resumed if it was previously stopped. (E.g., SIGCONT.)

4. Name and describe one way to programmatically send a signal to a process:
The kill system call is commonly used to send a signal to a process by its process ID.

Example: kill(pid, SIGUSR1) == 0

5. Name and describe one way to send a signal to a process from the command line:

- Use the kill command to send a signal by specifying the signal and process ID:
  - Example: kill -SIGUSR1 12345
- Alternatively, use Ctrl+C (for SIGINT) to interrupt a running process in the terminal.

6. POSIX Signals:

1. SIGINT

- **Description**: Interrupt signal, typically sent by the user using Ctrl+C in the terminal.
- **Default Disposition**: Terminate the process.
- **Can it be overridden?**: Yes. A custom handler can be defined, allowing graceful shutdown or resource cleanup.

2. SIGTERM

- **Description**: Termination request signal, usually sent to terminate a process gracefully.
- **Default Disposition**: Terminate the process.
- **Can it be overridden?**: Yes. A custom handler can allow resource cleanup or reject the termination.

3. SIGUSR1

- **Description**: User-defined signal for application-specific purposes.
- **Default Disposition**: Terminate the process.
- **Can it be overridden?**: Yes. It is specifically meant to be handled by custom signal handlers.

4. SIGKILL

- **Description**: Kill signal, used to forcefully terminate a process.
- **Default Disposition**: Terminate the process immediately.
- **Can it be overridden?**: No. The SIGKILL signal cannot be caught or ignored. This ensures the process is always terminated.

5. SIGSTOP

- **Description**: Stop signal, used to pause a process.
- **Default Disposition**: Stop the process.

- **Can it be overridden?**: No. The SIGSTOP signal cannot be caught or ignored, ensuring the process is paused by the OS.

Part 2: Working with a Signal Handler

1. Determine two (2) ways to send the SIGINT signal to the process created for the running program:

- **kill -SIGINT**: This method allows you to send the signal programmatically or from the terminal, even for processes not in the foreground.
- **Ctrl+C**: This is a convenient method for interactive sessions, typically used to interrupt and terminate a running foreground process

2. What do you observe? Why is this happening?
The program continuously repeats the null pointer message. This happens because any signal causes the program to return to the exact place the signal was received, meaning that it repeats the instance of the null pointer infinitely.