The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-08

POSIX Signals

Introduction

There are many mechanisms through which the processes communicate and in this lab we will discuss one such mechanism: **Signals**.

Signals inform processes of the occurrence of asynchronous events. In this lab we will discuss how user defined handlers for particular signals can replace the default signal handlers and also how the processes can ignore the signals. By learning about signals, you can "protect" your programs from Control-C, arrange for an alarm clock signal to terminate your program if it takes too long to perform a task, and learn how UNIX uses signals during everyday operations.

Assignment Hints

The signal() method

```
void (*signal(int sig, void (*func)(int)))(int)
/*
signal() returns the address of a function that takes an integer argument and has no return value. It requires two arguments: first is integer which is a signal name and second is the name of the function

For example: If you want to handle ctrl+c (which is SIGINT) and want to call your function on
ctrl+c, you will call function signal as follows

*/
  signal(SIGINT,your_func_name);
/*
from this point on all ctrl+c will call the function that you have defined. If you want to revert back to default, you can pass SIG_DFL as the function name for signal() call.
If you want to ignore a signal completely, you can specify SIG_IGN as the function name for
signal() call.
*/
/*
Ref: http://www.csl.mtu.edu/cs4411.ck/www/NOTES/signal/handler.html
Ref: https://www.tutorialspoint.com/c_standard_library/c_function_signal.htm
*/
```

The alarm() method

```
  unsigned int alarm(unsigned int seconds)
  /*
  this function generates SIGALRM signal to the calling process after 'seconds' seconds

  If you want to do something after a certain time, you can make a call to alarm(xx) so that it will
generate a SIGALRM after xx seconds. You need to make sure you have a function defined that
handles SIGALRM function and you have appropriate call to signal() function.
  */
int pause()
/*
pause ( ) suspends the calling process and returns when the calling process receives a signal. It
is most often used to wait efficiently for an alarm signal. pause ( ) doesn't return anything useful.
*/
```

The kill() method

```
int kill(pid, sig)
/*
The command kill sends the specified signal to the specified process or process group. If no signal is
specified, the TERM signal is sent. The TERM signal will kill processes which do not catch this signal.

- pid is the process-ID of the process to receive the signal
- sig is the signal number. The effective user IDs of sending and receiving processes must be the same,
or else the effective user ID of the sending process must be the super user.

If pid is equal to zero, the signal is sent to every process in the same process group as the sender. This
feature is frequently used with the kill command (kill 0) to kill all background processes without referring
to their process-IDs. Processes in other process groups (such as a DBMS you happened to have
started) won't receive the signal.

If pid is equal to -1, the signal is sent to all processes whose real user-ID is equal to the effective
user-ID of the sender. This is a handy way to kill all processes you own, regardless of process group.
*/
```

Demo Code

```
#include<stdio.h>
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>
#include <cstdlib>

//This function kills the process that makes a call to this function
 void kill_func(int killSignal)
{
        printf("Received kill signal %d\n",killSignal);
        printf("\tDying process %d\n",getpid());
        exit(0);
}
//this function prints a message and changes the function to handle the SIGINT command to
kill_func()
void myFunction(int sigVal)
{
        printf("Received signal %d\n", sigVal);
        printf("Now you can kill me..\n");
        signal(SIGINT,kill_func);
}
int main()
{
        //ignore the SIGINT signal
        // .. SIGINT is ctrl+c
        // .. SIG_IGN is signal ignore
        // .since we ignore the signal ctrl+c, you cannot terminate this process with ctrl+c
        signal(SIGINT,SIG_IGN);

        //for alarm signal, call function myFunction()
        // .. means when alarm goes off, myFunction will be called
        signal(SIGALRM,myFunction);

        //set alarm for 15 seconds from now
        alarm(15);

        printf("This gets printed as soon as alarm is called\n");

        //just running infinitely
        while(1);
}
```

Assignment 1

Write a program in which :

- A parent process creates a child process
- If the user sends ctrl+c to the parent process before 10 seconds from start, then the parent catches the signal and sends a message to indicate that the system is protected. The child just ignores the signal. Print messages from both parent and child process when they detect ctrl+c signal, both of which should indicate that they cannot be killed yet. Note that the child process must still ignore the ctrl+c signal, so the parent may have to send a different signal to it.
- After 10 seconds the parent process can now be terminated using ctrl+c, but before the parent terminates it must send a signal to the child and print statements stating it is trying to terminate the child process and has sent a signal. The child then prints its process id with a message saying it receives a signal to terminate from the parent and terminates. The parent then terminates printing a goodbye message.

Programming Hints

- Your parent forks a child.
- The parent and child both ignores ctrl+c signal
- After 10 seconds, when ctrl+c is received, the parent process should receive the signal, and send the kill signal to the child.
    - remember child still ignores ctrl+c command
    - maybe you need to send another signal from parent to child. Do you know how to create user defined signals?
- The child should handle the signal sent from the parent to terminate itself.
- The parent should terminate itself after the child is terminated.

Assignment 2

Create a Linux Time Bomb. This program stays idle throughout and does nothing. But when a user enters ctrl+c to terminate the program, it should print a page full of gibberish messages and only quit after 10 seconds of completion of printing gibberish.

Topics for Theory

- How many inter-process communication methods have we studied until now?
- For any 3 methods of Inter Process Communication listed, describe each of them.

Signal Reference

Signal Name	**Used For**
SIGABRT	Abnormal termination, abort
SIGALRM	Alarm clock expired (real time clocks)
SIGFPE	Floating point exception
SIGHUP	Controlling terminal hung up
SIGILL	Illegal instruction
SIGINT	Interactive termination (CTRL-C)
SIGKILL	Unstoppable termination (signal 9 on most UNIX systems)
SIGPIPE	Writing to a pipe with no readers
SIGQUIT	Quit from keyboard
SIGSEGV	Invalid memory reference
SIGTERM	Terminate process
SIGUSR1	Application-defined uses
SIGUSR2	Application-defined uses

Ref: https://man7.org/linux/man-pages/man7/signal.7.html

Deliverables

Lab Report

The following material in each section is expected:

1. Cover page with your name, lab number, course name, and dates
2. Theory/Background (Material or methods relevant to the lab, a few sentences on each in your own words)
    a. List all of the inter-process communication methods we have used in this class until now.
    b. Discuss any 3 methods of inter-process communication. Describe each of them.
3. Observations
    a. Screenshot of full execution of both assignments (must be terminal output)
4. Conclusion (Did your program work as expected, what can you take away from the lab?)
5. Appendix (for source code, submit the text in a table)

The report should be submitted as a single pdf document with the source code for your program within it.

Recorded Demonstration

The following material in each section is expected:

1. Introduce yourself and give the name of the lab
2. Compile the program
3. Show that the programs run similar to what was reported under observations

The demonstration may be in person or recorded and submitted as an mp4 file alongside the report.