University of Massachusetts Dartmouth
CIS 370, Fall 2013
10/29/2013, 10/31/2013
Lab 7: Signal Handling
Due: 11/05/2013 (Tuesday), 11/07/2013 (Thursday)

Objective

In this lab, you are going to understand and experiment with the UNIX (or Linux) **signal handling** mechanisms. Please review the attached slides before beginning.

Description

This lab consists of two parts:

1. Signal Override

In the UNIX system, a process (or the user) can notify another process about the occurrence of an event, by sending a **signal**. The receiving process of a signal decides how to respond to a signal:

1) Ignore it.
2) Handle it with some predefined function.
3) Let the system handle it with some default function.

Like IPC mechanisms such as message queues (Lab #5) and sockets (Lab #10), signals can be regarded as one way of implementing inter-process communication. However, signals are different in the fact that a signal interrupts the receiver's normal execution flow. When a signal is delivered to a process, the process's normal execution is interrupted and a signal handler (usually a function inside the process) is called and executed. After the completion of the signal handler, the normal execution flow of the process is resumed. Therefore, unlike normal IPC mechanisms, signals are unexpected events for a process. In this sense, signals are more like interrupts, often called soft interrupts. Just like an operating system designer must carefully design an interrupt handler, a programmer must carefully design a signal handler, such that a signal does not affect the normal execution of the program.

Deliverable

Download the template code from online:
www.cis.umassd.edu/~jplante/cis370/lab07/signalOverride.c

Your assignment is to modify this program to do the following:

- Ignore interrupts (triggered by pressing `<ctrl> + c`).
- Upon receiving a `SIGQUIT` signal (`<ctrl> + \`), print out a message indicating that `SIGQUIT` was triggered, and then exit.

2. Poker Cheat

In the old west, poker was more than just a fun game to play with friends and family. Wild Bill Hickok, who was well known as a gunfighter and scout, was shot dead while he was playing. When shot, he was holding a pair of aces and a pair of eights, all black. From then on that hand has been referred to as Dead Man's hand. Lesson to be learned: don't cheat!

Deliverable

Download the template code from online:
www.cis.umassd.edu/~jplante/cis370/lab07/poker.c

This program creates two poker players. You (child process) always lose, and since this program is obviously an extremely accurate substitute for the wild west, you can accuse your opponent of cheating...even if he isn't! If you think he is cheating, you reserve the right to shoot him. When accused of cheating, you (child process) should send a custom signal to the cheating player (parent process). When the parent process is caught cheating, and receives the signal from the child, it should get shot (by invoking the `getShot()` function included with the template code) and the game should end (loop should be broken and process should exit).

Hint: This is IPC so you will need to make sure you are able to pass signals between processes.
Second Hint: It's no coincidence that the *Dead* Man's hand is referenced above.