Signal System Calls

Section 7.4

Introduction

- Humans use signals to communicate in various situations
  - Traffic lights, ships, etc.
- Programs can also use **signals** to communicate with each other
- A form of **interprocess communication** (**IPC**)

- Signals consist of an "alphabet" of predefined signal types
  - Common for operating system to have 30–40 different signals
- Common POSIX signals
  - Table 7.2 page 230

Signal Handlers

- **Signal handlers** are code that runs when a process receives a signal
  - Default signal handler usually terminates the process
- We can write a function and have it be the signal handler for a specific signal
  - Kinda like event handlers in GUI programming

signal()

- **signal()** is used to set a signal handler for a particular event
  - **signal(*SIGNAL*, *signal_handler*)**
- When the specified signal happens, *SIGNAL*, the specified signal handler, *signal_handler()*, will be called automatically
- Need to **#include <signal.h>** to use signal()

- **signal1.c**
  - SIGFPE is a signal that indicates a floating point error happened
  - Runs f() when the error happens

- **signal2.c**
  - Signals can be ignored as well
  - When user presses CTRL-C, sends SIGINT to the program running
    - Program set to ignore SIGINT!
  - Need to press CTRL-\ to quit, which generates a SIGQUIT signal

kill()

- Signals can come from users, other programs, or the operating system
- A way a process can generate a signal is with kill()
  - **kill(*pid*, *SIGNAL*)**
  - Sends the specified signal to the specified PID

- **kill1.c**
  - Forks and has child wait for signal
  - SIGUSR1 is a sort of user defined signal
  - Parent waits for user to enter a command