ISA 563: Fundamentals of Systems Programming

Signals

Feb. 19, 2013

Signals in Unix/Linux

- Signals are interrupts sent to processes from:
  - the OS
  - other processes
  - hardware interrupts are sent to the OS by the hardware
- Each signal has:
  - an integer number to represent it
  - a symbolic name

Signals in Unix/Linux (Cont'd)

- For a list of supported signals:
  - $ kill -l
- Common signals:
  - SIGINT – causes process to terminate
  - SIGSTP – causes process to suspend
  - SIGHUP – sends a hang-up signal (when the controlling terminal closes)
    - Use nohup command to make your process immune to SIGHUP

Sending Signals

- From the keyboard:
  - Ctrl-C:
    - sends SIGINT.
    - by default causes the process to terminate
  - Ctrl-Z:
    - sends SIGTSTP
    - by default suspends the process
  - Ctrl-\:
    - sends SIGQUIT
    - by default, causes the process to terminate

Sending Signals (cont'd)

- To send a signal from the command line:
  - `$ kill -<signal> <pid>`
    - kills by pid
  - `$ pkill -<signal> pattern`
    - kills by process name
  - Both commands send SIGTERM by default
- To send signals from a program, use kill (2) system call:
  - `int kill(pid_t pid, int sig);`

Handling Signals

- Programs can register to catch signals using the signal library call:

```
#include <signal.h>

typedef void (*sighandler_t)(int);

sighandler_t signal(int signum, sighandler_t handler);
```

- Two signals cannot be caught:
  - SIGKILL – kills the process
  - SIGSTOP – always stops/pauses the process

Demo

sigcatch.c

Reentrant Functions

- A reentrant functions can be safely called again before previous invocation completes
- Non-reentrant functions introduce uncertainty when called from signal handlers
- Partial list of requirements for reentrancy:
  - Should not hold static/global data
  - Should not return a static/global non-const data
  - Must not call other non-reentrant functions, such as:
    - malloc/free
    - and many other standard IO library functions

alarm() / pause() functions

- alarm(int n):
  - sends a SIGALRM signal to the calling process in n seconds
- pause():
  - puts the calling process to sleep until a signal arrives

Demo

alarm.c