OPERATING SYSTEMS LABORATORY

Signals and Signal Programming

Spring 2011

Ali Rabiee

Imam Khomeini International University

LECTURE OVERVIEW

- All about signals
- Signals in the shell
- Programming signals
- Exercises
- Project statement

SIGNALS IN ANCIENT WORLD

SIGNALS IN OS WORLD

- Used for notifying a process of an event.
- Remember hardware interrupts? Signal is a form of software interrupt.
  - On signal delivery, the process's current execution flow is interrupted.
  - Signals may be ignored or blocked.
- A limited form of IPC; an a/synchronous notification.
  - No direct data communication.
- They are always delivered by the kernel of operating system.

SIGNAL DELIVERY PROCEDURE

1. Signal may be initiated by:
   - Other processes
   - The process itself
   - The kernel
2. Signal is generated by the kernel setting the flag in the signals bit-field of the process.
3. Signal is delivered by the kernel.
   - The signal is ignored.
   - The registered handler is called
   - The default handler is called

SIGNAL TYPES WE WORK MORE WITH

- SIGTERM: Termination Signal
- SIGKILL: Kill Signal
- SIGSTOP: OS-Stop Signal

Uncontrollable Signals

- SIGINT: Interrupt from keyboard (Ctrl+C)
- SIGCHLD: Child stopped or terminated
- SIGTSTP: Stop typed at tty (Ctrl+Z)
- SIGUSR1/SIGUSR2: User-defined signals
- Consult signal(7) for a full list. (i.e. run `man 7 signal`)

SIGNALS IN THE SHELL

- The TRAP command
- The KILL command

TRAP COMMAND

- Syntax

```
trap [-lp] [arg sigspec …]
```

- Example

```
1. trap 'echo Bye bye!' SIGCHLD
2. trap 'echo Stay please!' SIGTERM
3. trap 'echo Good luck! exit 1' SIGTERM
4. trap 'echo Hope to see you!' SIGKILL
```

KILL COMMAND

- Syntax

```
kill [-sigspec|-signum] [pid]
```

- Example

1. `kill 26374`
2. `kill –sigterm 26374`
3. `kill –hup 26374`
4. `kill -9 26374`

SIGNAL PROGRAMMING TOOLS

- What programming language?

C/C++

Don’t panic! It is really easy!

- What operating system?

We will use Fedora 1X as our preferred operating system.

Whilst any POSIX-compatible OS may be used.

SIGNAL PROGRAMMING SIGNAL FUNCTION(1)

- Header file: signal.h
- Function signature:

```
          typedef void (*sighandler_t)(int);
sighandler_t signal(int signum, sighandler_t handler);
```

- Example
    - Simple: signal1.c
    - Function interruption: signal2.c, signal3.c
    - Signal blocking/reset: signal4.c
    - Signaling others: signal5.c

SIGNAL PROGRAMMING SIGNAL FUNCTION(2)

- Why signal function is now no longer used?
  - Original Unix implementations reset the handler on signal delivery
  - Now usually it blocks instead of resetting the handler, but what if you need to handle every signal delivered to your program immediately? Registering a handler again works but there is a race.
  - You may need more info about the received signal, such as who has sent it or why it was sent.
  - Better solution developed: sigaction

EXERCISES

1. Write a program in which it sleeps forever until the user interrupts it twice, then exits. Once the first interrupt is received, tell the user: “Interrupt again to break.” Ignore the first interrupt every 5s.
2. Write a simple version of producer-consumer problem using signals in which producer starts producing only if there is no record and consumer starts consuming only if the buffer is full. You can use files as the shared buffer. Are the processes cooperating at their best performance?

PROJECT STATEMENT

Write two programs, named Agent and Controller, with the following definitions.

Agent program

On startup, the agent program creates the file /var/run/agent.pid and writes its PID into the file. It then reads the contents of a file called text.in, prints it on the console, and finally sleeps indefinitely.

The agent would be sensitive to two singal types: SIGHUP and SIGTERM. When the agent receives a SIGHUP, it reads the contents of the text.in and prints it on the console; when it receives a SIGTERM, it prints "Process terminating..." and then exits.

Controller program

On startup, it checks for a running agent by fetching the agent's PID from the file at /var/run/agent.pid and verifying it by sending the signum 0. If it could not find an agent, it prints "Error: No agent found." and then exits; otherwise, it prints "Agent found." and continues.

Then in an infinite loop, the controller prints "Choose a signal to send [1: HUP; 15: TERM]: " and then waits for user input.

When the user enters his/her choice, the controller sends the selected signal to the agent. If the user chooses SIGTERM, the controller should also terminate after sending the signal.

If the user presses Ctrl+C, the controller should send a SIGTERM to the agent and then exit.