Recitation 15: Exam Review - Signals

Instructor: TA(s)

Outline

- Proxylab
- Final Exam
- Signals

Proxylab

- **Proxylab is due Thursday (or late by Friday)**
  - No submissions will be accepted after Friday!
  - Submit something, even if doesn’t pass everything
- **Worth almost a letter grade**
- **Submit early**
  - Autolab may compile / run differently if you have undefined behavior or race conditions

Final Exam Details

- **Signups Out**
- **Final review session:**
  - Rashid Auditorium,
    Sunday Dec. 9 from 7-9 PM
- **Eight problems**
  - Nominal Time is 90-120 minutes, but you get four hours
  - Problems cover the entire semester, focus is on second half
- **Report to the room**
  - TA will verify your notes and ID
  - TAs will give you your exam server password
  - Login via Andrew, then navigate to exam server and use special exam password

Signals and Handling Reminders

- **Signals can happen at any time**
  - Control when through blocking signals

- **Signals also communicate that events have occurred**
  - What event(s) correspond to each signal?

- **Write separate routines for receiving (i.e., signals)**
  - What can you do / not do in a signal handler?

Signal Blocking

- **We need to block and unblock signals. Which sequence?**

```
pid_t pid;    sigset_t mysigs, prev;
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGCHLD);
sigaddset(&mysigs, SIGINT);
// need to block signals. what to use?
// A. sigprocmask(SIG_BLOCK, &mysigs, &prev);
// B. sigprocmask(SIG_SETMASK, &mysigs, &prev);

if ((pid = fork()) == 0) {
    // need to unblock signals. what to use?
    /* A. sigprocmask(SIG_BLOCK, &mysigs, &prev);
     * B. sigprocmask(SIG_UNBLOCK, &mysigs, &prev);
     * C. sigprocmask(SIG_SETMASK, &prev, NULL);
     * D. sigprocmask(SIG_BLOCK, &prev, NULL);
     * E. sigprocmask(SIG_SETMASK, &mysigs, &prev);
```

Signal Blocking

- **We need to block and unblock signals. Which sequence?**

```
pid_t pid;    sigset_t mysigs, prev;
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGCHLD);
sigaddset(&mysigs, SIGINT);
// need to block signals. what to use?
// [✓ A.] sigprocmask(SIG_BLOCK, &mysigs, &prev);
// B. sigprocmask(SIG_SETMASK, &mysigs, &prev);

if ((pid = fork()) == 0) {
    // need to unblock signals. what to use?
    /* A. sigprocmask(SIG_BLOCK, &mysigs, &prev);
     * B. sigprocmask(SIG_UNBLOCK, &mysigs, &prev);
     * C. sigprocmask(SIG_SETMASK, &prev, NULL);
     * D. sigprocmask(SIG_BLOCK, &prev, NULL);
     * E. sigprocmask(SIG_SETMASK, &mysigs, &prev);
```

Signal Blocking

- **We need to block and unblock signals. Which sequence?**

```
pid_t pid;    sigset_t mysigs, prev;
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGCHLD);
sigaddset(&mysigs, SIGINT);
// need to block signals. what to use?
// [✓ A.] sigprocmask(SIG_BLOCK, &mysigs, &prev);
// B. sigprocmask(SIG_SETMASK, &mysigs, &prev);

if ((pid = fork()) == 0) {
    // need to unblock signals. what to use?
    /* A. sigprocmask(SIG_BLOCK, &mysigs, &prev);
     * B. sigprocmask(SIG_UNBLOCK, &mysigs, &prev);
     * [✓ C.] sigprocmask(SIG_SETMASK, &prev, NULL);
     * D. sigprocmask(SIG_BLOCK, &prev, NULL);
     * E. sigprocmask(SIG_SETMASK, &mysigs, &prev);
```

Signal Blocking cont.

- **Someone implemented the wrong choices. Which signals are now blocked?**

```
pid_t pid;    sigset_t mysigs, prev;
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGCHLD);
sigaddset(&mysigs, SIGINT);

sigprocmask(SIG_SETMASK, &mysigs, &prev);
// What is blocked?

if ((pid = fork()) == 0) {
  sigprocmask(SIG_BLOCK, &prev, NULL);
  // What is blocked?
```

Signal Queuing

- **How many times is the handler invoked?**

```
void handler(int sig)
{ ...}

...
sigset_t mysigs, prev;
signal(SIGUSR1, handler);
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGUSR1);
sigprocmask(SIG_BLOCK, &mysigs, &prev);
kill(getpid(), SIGUSR1);
kill(getpid(), SIGUSR1);
sigprocmask(SIG_SETMASK, &prev, NULL);
```

Signal Delivery

- What can be printed?
- When is a blocked signal delivered?

```
sigset_t mysigs, prev;
sigemptyset(&mysigs);
sigaddset(&mysigs, SIGINT);
sigprocmask(SIG_BLOCK, &mysigs, &prev);
pid_t pid = fork();

if (pid > 0) {
    kill(pid, SIGINT);
    sigprocmask(SIG_SETMASK, &prev, NULL);
    printf("A");
} else {
    kill(getppid(),SIGINT);
    sigprocmask(SIG_SETMASK, &prev, NULL);
    printf("B");
}
```

Signal Delivery

- **Child calls kill(parent, SIGUSR{1,2}) between 2-4 times.
  What sequence of kills may only print 1?
  Can you guarantee printing 2?
  What is the range of values printed?**

```
int counter = 0;
void handler (int sig) {
  counter++;
}
int main(int argc, char** argv) {
  signal(SIGUSR1, handler);
  signal(SIGUSR2, handler);
  int parent = getpid();   int child = fork();
  if (child == 0) {
    /* insert code here */
    exit(0);
  }
  sleep(1);   waitpid(child, NULL, 0);
  printf("Received %d USR{1,2} signals\n", counter);
```

See http://www.cs.cmu.edu/~213/oldexams/final-s07.pdf

Signal Delivery

- **Suppose the program is currently inside the signal handler, which signals are blocked?**

```
int counter = 0;
void handler (int sig)
{
  counter++;
}
int main(int argc, char** argv)
{
  signal(SIGUSR1, handler);
  signal(SIGUSR2, handler);
}
```

Final Exam Q&A