Recitation 14: Final Exam Review

Instructor: TA(s)

Outline

- Exam Details
- Thread Synchronization
- Signals
- Processes
- Virtual Memory

Final Exam Details

- **Signups Coming**
- **Full review session coming Sunday May 7**
- **Eight problems**
  - Nominal Time is 90-120 minutes, but you get six hours
  - Problems cover the entire semester, focus 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

Thread Synchronization

- **Three types of locks**
  - Mutex
  - Semaphore
  - Reader-Writer lock
- **When would you want to use one over the others?**
- **Rule of thumb: protect shared variables and IO to the same file descriptor**
- **Avoid deadlocks: acquire locks in the same order in each thread**

Threads Questions

- What is a scenario where a reader-writer lock would be a more appropriate choice than a mutex?
- What happens when you join on a detached thread?

Threads Questions

- How many characters does "hello.txt" contain after this example?

```
void *work(void *data)
{
        write(*(int *) data, "a", 1);
        return NULL;
}


int main(void)
{
        int i, fd = open("hello.txt", O_RDWR);
        pthread_t tids[NTHREADS];
        for (i = 0; i < NTHREADS; ++i) {
                pthread_t tid;
                pthread_create(&tid, NULL, work, &fd);
                pthread_detach(tid);
        }
}
```

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 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

Processes

- Parent and child run in parallel as different processes
- No data in memory is shared between the two
- fork(): call once return twice
- execve(): never retuns (except in error)

Processes Question

- **What is printed to the terminal?**

```
const char *msg = “hello there”;
pid_t cpid;
int fd = open(“hello.txt”, O_RDWR);
char contents[12];
ssize_t nbytes;
if ((cpid = fork()) == 0) {
        write(fd, msg, strlen(msg));
        close(fd);
        exit(0);
}
waitpid(cpid, NULL, 0);
nbytes = read(fd, contents, strlen(msg));
contents[nbytes] = ‘\0’;
close(fd);
printf(“%s\n”, contents);
```

Virtual Memory

- Virtual to physical address conversion (TLB lookup)
- TLB miss
- Page fault, page loaded from disk
- TLB updated, check permissions
- L1 Cache miss (and L2 … and)
- Request sent to memory
- Memory sends data to processor
- Cache updated

Virtual Memory Example

- Translate 0x15213, given the contents of the TLB and the first 32 entries of the page table below.
- 1MB Virtual Memory
- 256KB Physical Memory
- 4KB page size

Index	Tag	PPN	Valid
0	05	13	1
	3F	15	1
1	10	0F	0
	05	18	1
2	1F	01	1
	11	1F	0
3	03	2B	1
	1D	23	0

VPN	PPN	Valid	VPN	PPN	Valid
00	17	1	10	26	0
01	28	1	11	17	0
02	14	1	12	0E	1
03	0B	0	13	10	1
04	26	0	14	13	1
05	13	0	15	18	1
06	0F	1	16	31	1
07	10	1	17	12	0
08	1C	0	18	23	1
09	25	1	19	04	0
0A	31	0	1A	0C	1
0B	16	1	1B	2B	0
0C	01	0	1C	1E	0
0D	15	0	1D	3E	1
0E	0C	0	1E	27	1
0F	2B	1	1F	15	1