Overview

- Exam 2 will be in class on **Thursday, November 9, 2017** from 10:00-11:45AM (please arrive early and sit with empty seats next to you on both sides).

- Exam 2 will be 105 minutes; therefore, if you have 50% additional time, you will have 160 minutes; if you have 100% additional time, you will have 210 minutes.

- Exam 2 will count as 10% of your final course grade.

- You may bring two double-sided (or four single-sided) 8.5”x11” crib sheets containing anything you would like; crib sheets will not be collected.

- Exam 2 will be handed back and reviewed on Monday, November 20, 2017.

- Grade grievances must be made no later than Monday, November 27, 2017 by handing your exam back to Professor Goldschmidt (after class or during office hours).

- Exam 2 will be comprehensive, covering everything we have done since the beginning of the semester. There will certainly be questions on more recent material (i.e., since Exam 1), except for Java. It is a good idea to review the quizzes, sample Exam 1 questions, and Exam 1 before taking Exam 2.
Exam 2 Review Questions

1. What do the following system calls do? What are their input arguments and return values?
   - pthread_create()
   - pthread_detach()
   - pthread_self()
   - pthread_join()
   - pthread_exit()

2. What are the system calls for working with shared memory segments in multi-process systems? What does each system call do? What are their input arguments and return values?

3. In a multi-threaded program, what kinds of synchronization errors can occur? How can these situations be avoided?

4. What is a critical section? Why is it important?

5. For semaphores, what are the definitions of the P() and V() operations? Further, what is a binary semaphore and what is a counting semaphore? How are each used to avoid synchronization problems?

6. Using P() and V() operations, write a solution to the Readers/Writers problem introduced in class on October 23.

7. Using P() and V() operations, write a solution to the Dining Philosophers problem introduced in class on October 26.

8. What is the OSI reference model?

9. What is data marshalling? Why is it important?

10. What are the key differences between TCP and UDP?

11. Describe the addressing scheme used in IP.

12. What do the following system calls do? What are their input arguments and return values? Note that the system calls marked with an asterisk (*) will not be on the exam.
   - socket()
   - bind()
   - recvfrom()
   - sendto()
   - listen() *
   - connect() *
   - gethostbyname() *
   - recv() *
   - send() *
Sample Exam 2 Questions

1. Answer the questions below.

(a) Given the following C program, what is the exact terminal output? If multiple outputs are possible, use a diagram to succinctly show all possibilities. Assume that all system calls complete successfully. Further, assume that the main thread ID is 1111, with any child threads numbered sequentially 8888, 8889, 8890, etc.

```c
void * omg( void * arg )
{
    int * x = (int *)arg;
    int rc = pthread_detach( pthread_self() );
    printf( "%u lucky %d %d\n",
            (unsigned int)pthread_self(), *x, rc);
    return NULL;
}

int main()
{
    pthread_t tid;

    int x = 7;
    printf( "%u \n", (unsigned int)pthread_self(), x );

    int rc = pthread_create( &tid, NULL, omg, &x );
    printf( "%u \n", (unsigned int)pthread_self(), rc );
    rc = pthread_create( &tid, NULL, omg, &x );

    usleep( 1000 ); /* wait for child threads to end ... */

    return EXIT_SUCCESS;
}
```

(b) What is the most likely terminal output of the code above if the `sleep()` call is commented out? Again assume that all system calls complete successfully. If multiple outputs are possible, do not show all possibilities.
2. Answer the questions below.

(a) Given the following C program, what is the **exact** terminal output? If multiple outputs are possible, use a diagram to succinctly show all possibilities. Assume that all system calls complete successfully. Further, assume that the main thread ID is 1111, with any child threads numbered sequentially 8888, 8889, 8890, etc.

```c
int x = 12;

void * lmao(void * arg)
{
    int q = 4;
    int * z = (int *)arg;
    q -= 2;
    printf( "%u %d %d\n", (unsigned int)pthread_self(), *z, q );
    *z *= 2;
    return NULL;
}

int main()
{
    printf("%u %d\n", (unsigned int)pthread_self(), x );
    pthread_t tid1, tid2;
    pthread_create( &tid1, NULL, lmao, &x );
    pthread_create( &tid2, NULL, lmao, &x );
    pthread_join( tid1, NULL );
    pthread_join( tid2, NULL );
    printf("%u %d\n", (unsigned int)pthread_self(), x );
    return EXIT_SUCCESS;
}
```

(b) What is the **exact** terminal output of the code above if the two `pthread_join()` calls are removed? Again assume that all system calls complete successfully. If multiple outputs are possible, use a diagram to succinctly show all possibilities.
3. Expand the code from Question 2 (shown below) by implementing a “mutex” to ensure no synchronization problems occur with shared variable $x$.

```c
int x = 12;

void * lmao( void * arg )
{
    int q = 4;
    int * z = (int *)arg;
    q -= 2;
    printf( "%u %d %d\n", (unsigned int)pthread_self(), *z, q );
    *z *= 2;
    return NULL;
}

int main()
{
    printf( "%u %d\n", (unsigned int)pthread_self(), x );
    pthread_t tid1, tid2;
    pthread_create( &tid1, NULL, lmao, &x );
    pthread_create( &tid2, NULL, lmao, &x );
    pthread_join( tid1, NULL );
    pthread_join( tid2, NULL );
    printf( "%u %d\n", (unsigned int)pthread_self(), x );
    return EXIT_SUCCESS;
}
```
4. Answer the questions below.

(a) Given the following C program, what is the exact terminal output? If multiple outputs are possible, use a diagram to succinctly show all possibilities. Assume that all system calls complete successfully. Further, assume that the main thread ID is 1111, with any child threads numbered sequentially 8888, 8889, 8890, etc.

(b) What is the exact contents of the EE.txt file?

```c
#define SNAPCHAT 2

void * rofl( void * arg )
{
    int * f = (int *)arg;
    printf( "%uAA%d\n", (unsigned int)pthread_self(), *f );
    fprintf( stderr, "%uBB\n", (unsigned int)pthread_self() );
    return NULL;
}

int main()
{
    close( SNAPCHAT );
    printf( "%uCC\n", (unsigned int)pthread_self() );
    fprintf( stderr, "%uDD\n", (unsigned int)pthread_self() );

    int fd = open( "EE.txt", O_WRONLY | O_CREAT | O_TRUNC, 0660 );
    printf( "%uFF\n", (unsigned int)pthread_self() );
    fprintf( stderr, "%uGG%d\n", (unsigned int)pthread_self(), fd );

    pthread_t tid1, tid2;
    int rc = pthread_create( &tid1, NULL, rofl, &fd );
    rc = pthread_create( &tid2, NULL, rofl, &rc );

    pthread_join( tid1, NULL );
    pthread_join( tid2, NULL );

    printf( "%uHH\n", (unsigned int)pthread_self() );
    fprintf( stderr, "%uII\n", (unsigned int)pthread_self() );

    fflush( NULL );
    close( fd );

    return EXIT_SUCCESS;
}
```

(c) Repeat parts (a) and (b) above if SNAPCHAT is set to 1 instead of 2.