Systems Programming
Final Exam
Fall 2016

Name _____

RUID _____

Section Number _____

- **Do not open this exam** until everyone has an exam.
- Write your name, RUID and section number in the space provided.
- There are 10 pages in this exam, including this one. Make sure you have them all.
- This exam is closed book – closed notes.
- You must leave all electronic devices not explicitly exempted at the front of the room.
- You **may not** use a calculator.
- **Write clearly** - if we can't read or find your answer, your answer is wrong.
- **Make clear** which questions you want graded.

Question Type	**Points Total**	**Scored Amount**
0. Required Stuff	140	
1. Choose Your Own Exam	100	
Total:	230	

A. Required Stuff

0. (a) Presume the following code has run:

```
int x = 4;
int* y = &x;
int** z = &y;
int a = 10;
```

Write 4 different assignment operations that set x equal to the value of a.

(b) Presume the following code has run:

```
int x = 4;
int* const y = &x;
int const **z = &y;
int a = 10;
```

Which of your assignments (if any) from 0.a are now invalid?

What does the `const` modifier do to each of the pointers it is applied to?

1. Write some code that forks a process and prints "parent" in the parent process and "child" in the child process

2. What is the difference between `fork()` and `exec()`? Can one `fork()` without `exec()`ing

3. Assume a parent forks a child immediately on start and then immediately waits on the child. The parent takes 1s of execution time and the child takes 2s of execution time.

   (a) On a single core processor, what is the total time (wall clock time) for execution? Why?

   (b) On a multi-core processor, what is the total time (wall clock time) for execution? Why?

4. Assume a parent creates a thread immediately on start and does not join in. Assume the thread is scheduled by the kernel. The process' main function takes 1s of execution time, and the thread takes 2s of execution time.

   (a) On a single core processor, what is the total time (wall clock time) for execution? Why?

   (b) On a multi-core processor, what is the total time (wall clock time) for execution? Why?

5. Are synchronization constructs required for multiprocessing programs? Why or why not?

6. Show how a race condition can occur between 2 threads running the code below at the same time:

```
int i = 0;
void* foo() {
    printf("%d", i);
    i++;
}
```

7. Consider the following communication calls:

   `Socket, bind, connect, accept, close`

   If you are building a client program, which calls would you most likely use? What do they do?

8. What is a file descriptor? Why would you need a 'file' descriptor to do socket communication?

9. Why does server code usually involve concurrent programming?

10. What does `getaddrinfo()` do when a given a host name rather than an IP address?

11. What does `gethostbyname()` do when given an IP address rather than a host name?

12. When might `read()` on a socket return fewer bytes than specified?

13. A program calls `bind()`, and `bind()` again. What could happen?

B. Choose Your Own Exam!

Choose and answer 10 of the questions below:

0. Does a mutex lock guarantee thread safety? Why or why not?

1. When can casting a struct ultimately end in a segmentation fault?

2. What information is stored on the stack?

3. What processes inherits zombie processes? Why?

4. How can a parent process communicate information to a child process?

5. What happens to the parent if the child segfaults?

6. What happens to a process when one of its threads segfaults?

7. What is a signal? How are they different from Exceptions in Java?

8. What is shared among threads in a multithreaded process?

9. Make the following code thread safe:

```
int total = 0;
void add(int value) {
    if (value < 1) return;
    total += value;
}
void sub(int value) {
    if (value < 1) return;
    total -= value;
}
```

10. If we wish to ensure that the value of `total` in the code in 9 NEVER exceeds 1000, what can we do? Insert the necessary constructs/calls to do so:

11. Provide an example of code that can lead to deadlock. Describe how deadlock might occur.

12. There are 2 problems with the code below, what are they?

```
char* xpx(char* src) {
    char result[sizeof(src)];
    strcpy(result, src);
    return result;
}
```

Scratch/Additional space: