Systems Programming Final Exam

CS 283 Winter 2013
Professor: William M. Mongan
Date: March 11, 2013

Question	Score
1	
2	
3	
4	
5	
Total:	

Name: ____

Drexel ID: ____

Instructions: The exam must be completed in two hours. You should read all of the questions before you begin and plan your time accordingly. The exam is worth **100 points**.
There are five questions.

The exam is closed-book and you may not use your class notes. You may have a crib sheet of your own notes. You may also use a calculator. **You may not**, **however**, **use any notes that are not your own**: specifically copies of web pages (other than this course web pages) and notes that belong to other students are not allowed. You cannot use your notebook computer during this exam. **There will be absolutely no access to any wireless network during this exam.**

You should write your answers in the space provided. Try to make your answers as clear and concise as possible, and write them legibly. Show all your work as partial credit may be given.

1. [35 Points] Signal Handlers and Process Management

- **(10 Points)** What is the output of this program, and WHY.

```
int val = 10;

void handler(sig)
{
    val += 5;
    return;
}

int main()
{
    int pid;

    signal(SIGCHLD, handler);
    if ((pid = fork()) == 0) {
        val -= 3;
        exit(0);
    }
    waitpid(pid, NULL, 0);
    printf("val = %d\n", val);
    exit(0);
}
```

- **(10 Points)** What is the output of this program? There are several possibilities (indicate all of them). Here are a few hints: wait() returns -1 when there is no child to wait for, and WEXITSTATUS gives the exit status given by the exiting process.

```
#include <sys/wait.h>

main() {
  int status;

  printf("%s\n", "Hello");
  printf("%d\n", !fork());

  if(wait(&status) != -1)
    printf("%d\n", WEXITSTATUS(status));

  printf("%s\n", "Bye");

  exit(2);
}
```

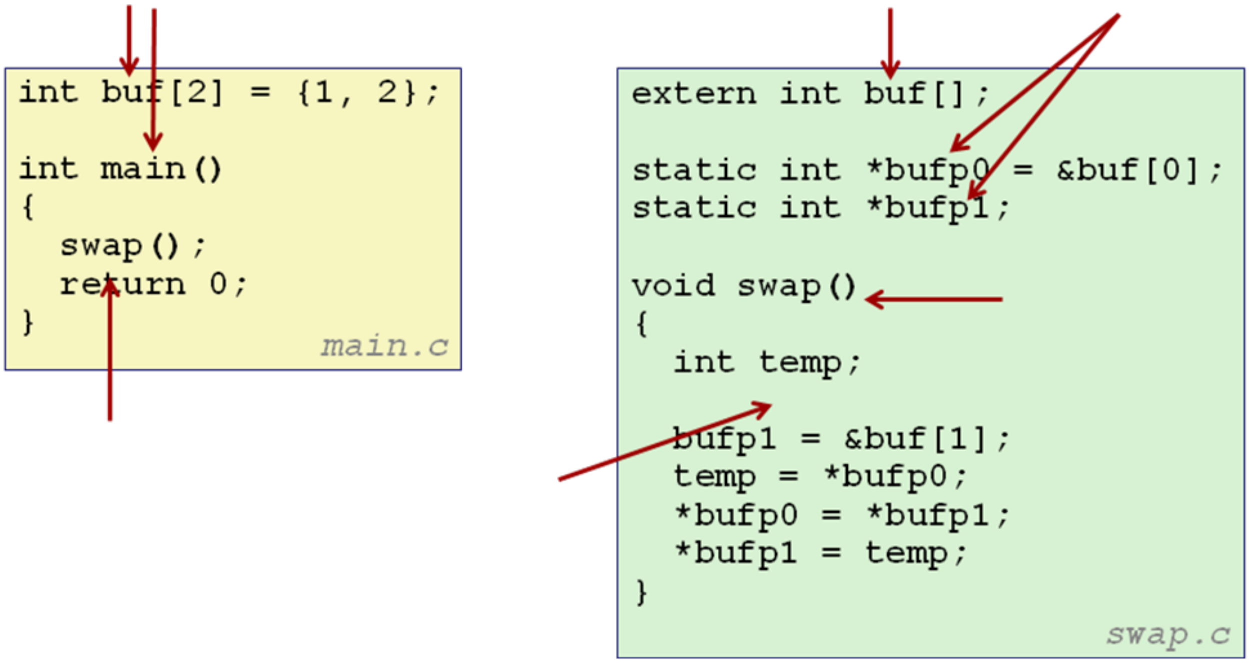
- **(5 Points)** What does the wait() function do?

- **(5 Points)** What would happen if we forked a child, forgot to call wait(), and then main() exited normally, while the child entered an infinite loop?

- **(5 Points)** What would happen if we forked a child, forgot to call wait(), and then main() entered an infinite loop, while the child exited?

2. Linkers and Memory [15 points]

- **[5 points]** In the diagram below, label each of the symbols pointed to by arrows as global, local, or external. Then label all the globals as either strong or weak.



```
int buf[2] = {1, 2};

int main()
{
  swap();
  return 0;
}
```

```
extern int buf[];

static int *bufp0 = &buf[0];
static int *bufp1;

void swap()
{
  int temp;

  bufp1 = &buf[1];
  temp = *bufp0;
  *bufp0 = *bufp1;
  *bufp1 = temp;
}
```

- **[10 points]** Consider the following code fragment. What will be the output **and why**?

p1.c
```
int x = 5;
p1() {

    printf("%d", x);
}
```

p2.c
```
int x;
p2() {
      x = 10;
      printf("%d", x);

}
```

  - **(5 Points)** What will be the output and **why**?

  - **(5 Points)** Suppose this is not the output you wanted. How could you correct the program (hint, you need only change one line of code; and you must *not* change any variable names).

3. [10 Points] Memory Systems

- **[10 Points]** An UltraSPARC-III processor has a 64 Kilobyte four-way set associative L1 data cache with block size of 32 bytes.
  - **[5 points]** Draw a diagram of this cache, or make a list that indicates the number of words per block, number and size of blocks, and number and size of sets.
  - **[3 points]** What is the block number for the word address of 67? (**Show your calculation.**)
  - **[2 points]** What are the addresses of the words within that block? (**Show your calculation**

4. [20 Points] Interprocess Communication: Shell Management

- **[5 Points]** When writing your shell, you may have noticed that it was necessary to sleep(1) in a loop after spawning a foreground process rather than waiting for it immediately. This is likely contrary to your expectation. Why was this done?

- **[5 Points]** Both foreground and background processes are reaped by a loop that is invoked on SIGCHLD. However, each child that terminates throws a SIGCHLD signal, so why is it necessary to call waitpid() in a loop in the SIGCHLD handler?

- **[5 Points]** How did you prevent this waitpid() loop from blocking indefinitely in the event that there were no children to reap?

- **[5 Points]** In your waitpid() loop, you may have simply gathered the process that you just reaped by waitpid() and removed it from your jobs list. Is there ever a time when this behavior would be incorrect as a response to receiving SIGCHLD? Why or why not?

5. [20 Points] Concurrent Programming

- **(5 Points)** What does it mean for a function to be "reentrant?" Given a function that is non-reentrant, discuss some strategies for making it so.

- **(10 Points)** Consider the library function char* strtok(char* str, const char* delimiters).
  - **(2 Points)** What does this function do?
  - **(2 Points)** How does it do it?
  - **(3 Points)** Why is this code non-reentrant?
  - **(3 Points)** Modify the function prototype to one that would help make strtok() reentrant. Then describe (you do not have to code) its new, reentrant behavior.

- **[5 Points]** A friend approaches you with a program that computes matrix multiplication of two 1000x1000 matrices, using 8 threads on a quad-core computer. The code that each thread calls to do its share so appears below (you may assume that it works, and that the pthreads are correctly created and used).
  - **(3 Points)** Your friend tells you that the performance observed from this program degrades as more threads are added. Why is this happening and what can be done about it?
  - **(2 Points)** Further, you notice that the memory access patterns of C, A, and B are not the most efficient for a typical cache/memory architecture. Suggest a memory-based improvement to our friend as well.

```
// SIMD function to perform matrix multiplication
// Taken from http://heshans.blogspot.com/2009/05/matrix-multiplication-using-pthread.html
void mm(int me_no, int noproc, int n, double a[NDIM][NDIM], double b[NDIM][NDIM],
double c[NDIM][NDIM])
{
   int          i,j,k;
   i=me_no;
   while (i<n) {
      for (j = 0; j < n; j++) {
         c[i][j] = 0.0;
         for (k = 0; k < n; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
         }
      }
      i+=noproc;
   }
}
```