CSCI 331: *Operating Systems*

Midterm #2 -- open book section

The entire exam is to be turned in at 8:30PM. Work the closed book section first and turn it in before you consult your books and notes to work on the open book section. For the closed book section, write your answers on the exam itself.

There are two pages of questions for the closed book section of the exam.

Problem 1: 5 points

Suppose a C program starting with the following header:

```
main(argc, argv)
     int argc;
     char *argv[];
```

is compiled and the compiled code is stored in the file `now`. If the command:

```
now I get it
```

is executed, what are the values of `argc`, `argv[0]` and `argv[1]`?

Problem 2: 5 points (Exercise 4.11 of the textbook)

After a process has exited, it may enter the state of being a zombie, `SZOMB`, before disappearing from the system entirely. What is the purpose of the `SZOMB` state? What event causes the process to exit from `SZOMB`?

Problem 3: 5 points (Exercise 5.23 of the textbook)

Give two reasons for swapping to be initiated?

Problem 4: 5 points (Exercise 6.2 of the textbook)

Why is the close-on-exec bit located in the per-process descriptor table instead of the system file table?

Problem 5: 5 points (Based on Exercise 7.7 of the textbook.)

The Unix file system on `ivy` supports blocks of 8096 bytes and fragments of 1024 bytes. How many blocks and fragments are allocated for a file 10,000 bytes long? How many for a file 100,000 bytes long?

Problem 6: 5 points

Will the savvy Unix programmer ever use the system call `dup2()` in writing a program that does not contain the `execve()` system call? *Briefly* explain the reasons for your answer.

Problem 7: 10 points

I'm logged into `ivy`. (Honestly, I am. Windows 3.0 can multi-program.) I type the line:

```
ls -id /etc /tmp
```

which displays the inode numbers of `/etc` and `/tmp` with the following output:

```
4096 /etc
2051 /tmp
```

Immediately, I type the line:

```
ls - i /
```

which displays the inode numbers of all files and subdirectories referenced within the root directory. Among the many lines printed are the following:

```
4096 /etc
   2 /tmp
```

Why are different inode numbers given for `/tmp` in these two cases? Why is only one inode number given for `/etc` in both cases?

Big hint #1 -- When does a directory have more than one inode number?
Big hint #2 -- What's so special about inode number 2?

If you have no idea why there are two inode numbers for `/tmp`, you can state where inode numbers are stored in the Unix file system for some partial credit.

Problem 8: 10 points

Suppose the following program is started from the shell.

```
#include <fcntl.h>
main(int argc, char *argv[])
{
  int f, g;
  char buff[100];
  f = open("/etc/motd", O_RDONLY);
  (void) fork() ;
  g = open("/etc/passwd", O_RDONLY);
  /* Stop here until input appears on standard input */
  read(0, buff, 50);
} ;
```

How many descriptor table entries are allocated as this program (with parent and child processes) executes? How many file table entries are allocated?

Sketch the ``state'' of the per-process descriptor tables and the system file table when both processes block on the `read` system call. That is, show how the per-process descriptor table entries point to the relevant system file table entries. The only details you *must* show to receive full credit are these pointers plus the reference counts of the file table entries to which they point.

You may assume that the shell is the only other process that has the terminal open.