IC221 System Programming
Spring 2014                                    NAME:_____

HW5                              COLLABORATOR(S):_____

5/3/0  1. Explain how the OS provides abstraction and isolation via
          the System Call API.

      ┌─────────────────────────────────────────────────────────────┐
      │                                                             │
      │                                                             │
      │                                                             │
      │                                                             │
      │                                                             │
      └─────────────────────────────────────────────────────────────┘

10/8/5/0  2. Match the OS system resource to the action. (match all that
             apply)

          Device Management   (1)        ___ Writing to a file

          Process Management  (2)        ___ Reading user input from the
                                             terminal

          Memory Management   (3)        ___ Adjusting the break point

          File Management     (4)        ___ Executing a program

5/3/0  3. Why are certain operations in an on *privileged*? What is the
          Operating System protecting us from?

      ┌─────────────────────────────────────────────────────────────┐
      │                                                             │
      │                                                             │
      │                                                             │
      │                                                             │
      └─────────────────────────────────────────────────────────────┘

5/3/0  4. What is a kernel? And why must it be trusted?

      ┌─────────────────────────────────────────────────────────────┐
      │                                                             │
      │                                                             │
      │                                                             │
      └─────────────────────────────────────────────────────────────┘

5/3/0  5. What section of the man pages are system call found and in
          and what sections are library functions in?

      ┌─────────────────────────────────────────────────────────────┐
      │                                                             │
      │                                                             │
      │                                                             │
      └─────────────────────────────────────────────────────────────┘

/30                              1 of 4

5/3/0   6. Open the manual page for **sigaction()** and **signal()**, which is
        the system call and which is the library function? How did you
        determine this?

```

```

5/3/0   7. What is the difference between **malloc()** and **sbrk()** from a
        system programmer perspective? Why is one a system call and
        one a library function? (APUE discusses this)

```

```

5/3/0   8. Explain a *context switch* with respect to the kernel-space,
        user-space and system calls.

```

```

9. What does this code do?

10/8/5/3/0                  write(1, "Go Navy!", 7 );
                            write(2, "Beat Army!", 9);

```

```

/25

5/3/0   10. What is the difference between a string and a buffer?

```



```

5/3/0   11. What type is a file descriptor? And what does the file descriptor reference?

```


```

12. Complete the code segment for writing the bytes of a double to standard out and then reading a double from standard input.

10/8/5/3/0

   double a;

write( [_____] );

   double a;

read( [_____] );

5/3/0   13. Explain the concept of an ORing and how it encodes option flags.

```


```

/25

5/3/0    14. What is the open command opening a file at "path/to/file" for append mode and create, if it doesn't already exist? Use a mode of 0644.

```



```

15. According to the empirical study in APUE, what is the best buffer size for reading and writing from devices and why?

10/8/5/3/0

```




```

5/3/0    16. Why does the O.S. buffer writes to disks? Why not just write everything as it is requested from the user?

```



```