IC221 System Programming
Spring 2016 NAME:_____
HW7

COLLABORATOR(S):_____

150 POINTS

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

5/3/1/0 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

5/3/1/0 3. Why are certain operations in an on *privileged*? What is the Operating System protecting us from?

5/3/1/0 4. What is the kernel? And why must it be trusted?

5/3/1/0 5. What section of the man pages are system call found and in and what sections are library functions in?

___/25

NAME: _____

5/3/1/0 6. Open the manual page for **read()** and **fread() (),** which is the system call and which is the library function? How did you determine this?

5/3/1/0 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)

5/3/1/0 8. Explain a *context switch* with respect to the kernel-space, user-space and system calls.

5/3/1/0 9. What is a **trap**? How does it relate to context switching?

5/3/1/0 10. Find the man page for the system call **open(),** what is the man command you need to access it? Explain why you can't just type **man open?**

__/25

NAME: _____

5/3/1/0 11. What is the output of this code snippet, and to which standard file descriptor does each write to.

```
write(1, "Go Navy!", 7 );
write(2, "Beat Army!", 9);
```

5/3/1/0 12. What is the difference between a string and a buffer

5/3/1/0 13. What is the type of a file descriptor? What does a file descriptor reference?

8/5/3/0 14. Complete the following code segment for write the bytes of the float f to the standard output file descriptor, and reading in the bytes of a float into f.

```
float f=3.1415926;
write(                                  );

float f;
read(                                   );
```

5/3/1/0 15. Explain the concept of an ORing and how it encodes options and mode to open()?

__/28

NAME: _____

10/8/4/0 16. Complete the ORing option strings to match the equivelent fopen() options:

r

w

r+

w+

a

10/8/4/0 17. Complete the program below that properly opens the file **helloworld.txt** with read permission and then writes the string "Hello World" to that file:

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char * argv[]){

      int fd;
      char helloworld[]="Hello World!";

      //open file helloworld.txt
      
      //write helloworld to file
      
      //close file
      
}
```

__/20

NAME: _____

10/8/4/0 18. Write the equivlent mode ORing for the octal permissions:
(note: leading 0 indicates number is in octal)

0777

0640

0740

0501

5/3/1/0 19. What is the umask and when is it applied? Why is it considered a security perameter?

5/3/1/0 20. Type **umask** in the shell on a lab computer, what is your current umask?

5/3/1/0 21. The **touch** command will opeb a file with the creation mode of 0666, that is, read+write for user, group, and everyone. What should be the permission of the new created file? **Exlpain.**

5/3/1/0 22. What is the umask such that alll created files should never have group should never have default write or execute, but everyone and user can have any permission?

__/30

NAME: _____

10/8/4/0 23. Match the following description to the **struct stat** member:

st_mode ____	(a) The user id of the owner
st_uid ____	(b) The last modificaiton time
st_atime ____	(c) The size in bytes of the file
st_mtime ____	(d) The group id of file
st_ctime ____	(e) The creation time
st_size ____	(f) The number of file-system blocks to store the file
st_blksize ____	(g) The last access time
st_gid ____	(h) The permissions for the file

5/3/1/0 24. System calls return what on error? What function can print a succient error message based on checking **errno**?

7/5/3/0 25. The system call **utimes()** modifies the access and modification time and has the following protype:

utimes(const char *path, const struct timeval times[2]);

The array of **struct timeval** is of length 2, what is the first timevalin the array refer to and what does the second with respect to the file being modified?

__/22