Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_BOT THANOS]

Experiment Title: UNIX FILE OPERATIONS AND SYSTEM CALLS

Aim/Objective:

The objective of using file operations is to provide a way to manage and manipulate files stored on storage devices such as hard drives, solid-state drives, or network drives. System calls in Unix are used for file system control, process control, inter-process communication, etc. Access to the Unix kernel is only available through these system calls. Generally, system calls are like function calls, the only difference is that they remove the control from the user process.

Description:

File operations in operating systems refer to the various actions or tasks that can be performed on files within a file system. These operations are essential for managing files and manipulating their contents.

Pre-Requisites:

- Creation of a new file (open with attributes as “a” or “a+” or “w” or “w++”)
- Opening an existing file (open) and Reading from a file (fscanf, fgets, or fgetc)
- Writing to a file int fputc, fputs
- Moving to a specific location in a file (fseek, rewind) and closing a file(close)
- Basic system calls on files.

Pre-lab task: Learn the System Calls below before starting In-Lab

System Calls	Windows	Unix
Process Control	CreateProcess()	Fork()
	ExitProcess()	Exit()
	WaitForSingleObject()	Wait()
File manipulation	CreateFile()	Open()
	ReadFile()	Read()
	WriteFile()	Write()
		Close()
Device Management	SetConsoleMode()	Ioctl()
	ReadConsole()	Read()
	WriteConsole()	Write()
Information Maintenance	GetCurrentProcessID()	Getpid()
	SetTimer()	Alarm()
	Sleep()	Sleep()
Communication	CreatePipe()	Pipe()
	CreateFileMapping()	Shmget()
	MapViewOfFile()	Mmap()
Protection	SetFileSecurity()	Chmod()
	InitializeSecurityDescriptor()	Umask()
	SetSecurityDescriptorgroup()	Chown()

SLNO	FUNCTIONS	FUNCTIONALITY/PROTOTYPE
1	fopen()	Opens a file and returns a file pointer.
2	fclose()	Closes an open file associated with a file pointer.
3	getc()	Reads a character from a file.
4	put()	Writes a character to a file.
5	fscanf()	Reads formatted data from a file.
6	fprintf()	Writes formatted data to a file.
7	gets()	Reads a string from standard input.
8	puts()	Writes a string to standard output.
9	fseek()	Moves file pointer to a specific position.
10	tell()	Returns the current position of file pointer.
11	rewind()	Resets file pointer to the file's beginning.

SLNO	System Call	FUNCTIONALITY/PROTOTYPE
12	access()	Checks user permissions for a specific file.
13	chdir()	Changes the current working directory.
14	chmod()	Changes file or directory permissions.
15	chown()	Changes file ownership to a specific user.
16	kill()	Sends a signal to a specific process.
17	link()	Creates a new link for an existing file.
18	open()	Opens a file for reading or writing.
19	pause()	Suspends the process until receiving a signal.
20	exit()	Terminates the program with a specific status.
21	alarm()	Sets a timer to deliver a signal.
22	fork()	Creates a new process by duplicating.

In lab task

1. Write a C program that reads file.txt line by line and prints the first 10-digit number in the given file (digits should be continuous), If not found then print the first 10 characters excluding numbers.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int main() {
   FILE *file = fopen("file.txt", "r");
   if (file == NULL) {
      printf("File not found or could not be opened.\n");
      return 1;
   }

   char line[1024];
   while (fgets(line, sizeof(line), file)) {
      char *ptr = line;
      char result[11];
      int result_index = 0;
      while (*ptr != '\0' && result_index < 10) {
         if (isdigit(*ptr)) {
            result[result_index] = *ptr;
            result_index++;
         } else {
            result_index = 0;
         }
         ptr++;
      }
      if (result_index == 10) {
         result[10] = '\0';
         printf("First 10-digit number: %s\n", result);
         break;
```

```
        }
    }


    if (feof(file)) {
        fseek(file, 0, SEEK_SET);
        char first_10_chars[11];
        int char_index = 0;
        while (char_index < 10 && (first_10_chars[char_index] = fgetc(file)) != EOF) {
            if (!isdigit(first_10_chars[char_index])) {
                char_index++;
            }
        }
        first_10_chars[char_index] = '\0';
        printf("First 10 non-digit characters: %s\n", first_10_chars);
    }

    fclose(file);
    return 0;
}
```

2. Write a C program that saves 10 random numbers to a file, using your own “rand. h” header file whichcontains your own random () function.

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int myRandom(int min, int max) {

   return min + rand() % (max - min + 1);

}

int main() {

   srand(time(NULL));

   int randomNumbers[10];

   for (int i = 0; i < 10; i++) {

      randomNumbers[i] = myRandom(1, 100);

   }

   printf("Generated random numbers: ");

   for (int i = 0; i < 10; i++) {

      printf("%d ", randomNumbers[i]);

   }

   printf("\n");

   return 0;

}
```

- **Data and Result:**

Data

Random numbers generated using `myRandom` function, ranging between 1 and 100, stored in array.

Result

Displayed 10 random numbers generated within the specified range.

- **Analysis and Inferences:**

Analysis

The `myRandom` function generates numbers uniformly between 1 and 100 using modular arithmetic.

Inferences

Random numbers are distributed uniformly within the specified range and printed correctly.

Post Lab

1. Write commands/code in the space provided for each of the questions below:

a. Use the cut command on the output of a long directory listing to display only the file permissions. Then pipe this output to sort and unique to filter out any double lines. Then use the wcto count the different permission types in this directory.

Ans.

- **ls -l | cut -c 1-10 | sort | uniq | wc -l**

b. Try ln -s/etc/passwd passwords and check with ls-l.
Did you do anything extra?

Ans.

- **ln -s /etc/passwd passwords**
- **ls -l**

c. Create a new Directory LABTEMP and Copy the files from /var/log into it and display the files whose first alphabet is consonant that do not begin with upper case letters that have an extension of exactly three characters.

Ans.

- **mkdir LABTEMP**
- **cp /var/log/* LABTEMP/**
- **ls LABTEMP/ | grep '^[^AEIOUaeiou][a-z]*\.[a-z]\{3\}$'**

d. Find how many hours has the system been running.

Ans.

- **uptime | awk '{print $3}'**

e. What command can be used to display the current memory usage?

Ans.

- **free -h**

f. Represent UNIX/LINUX File Hierarchy

Ans

- **/ (Root)**
- **|-- bin**
- **|-- etc**
- **|-- usr**
- **|-- var**
- **|-- home**

g. Create a file using cat and find the number of lines, words, and charactersin it.

Ans.

- **cat > file.txt**
- **wc file.txt**

Perform the Following

h. How do you compare files? (Hint:

CMP)Ans.

- **cmp file1.txt file2.txt**

i. How do you display\using echo command?

Ans.

- **echo "\\"**

j. How do you split a file into multiple files? (Hint: split)

Ans.

- **split -l 10 file.txt output_prefix**

k. What happens when you enter a shell meta character * with the echo command?

Ans.

- **echo ***

l. Perform the Following

a. Check$>newfile<infile wc and give theresult.

Ans.

- **newfile < infile | wc**

b. Redirect error message to file ERROR on cat command for non-existing file.

Ans.

- **cat non_existing_file 2> ERROR**

c. Redirect standard output and standard error streams for cat command with an example in one step.

Ans.

- **cat non_existing_file > output 2>&1**

m. Perform the following.

a. Compress and uncompress a file ERROR

Ans.

- **gzip ERROR**
- **gunzip ERROR.gz**

b. Zip a group of files (ERROR, new file, in file, passwd, group) and unzip them

Ans.

- **zip files.zip ERROR newfile infile passwd group**
- **unzip files.zip**

c. zip a group of files and unzip them.

Ans.

- **zip myfiles.zip file1.txt file2.txt**
- **file3.txt**
- **unzip myfiles.zip**

n. Create a file called "hello.txt" in your home directory using the command.
cat-u>hello. txt.
Ask your partner to change into your home directory and
run tail -f hello.txt.
Now type sever allies into hello.txt. What appears on your partner's screen?

Ans.

- **cat -u > ~/hello.txt**
- **tail -f ~/hello.txt**

o. Change the unmask value and identify the difference with the earlier using touch, ls–l

Ans.

- **umask 022**
- **touch newfile**
- **ls -l newfile**

p. Save the output of the who command in a file, display it, and display lines count on theterminal.

Ans.

- **who > who_output.txt**
- **cat who_output.txt**
- **wc -l who_output.txt**

q. Two or more commands can be combined, and the aggregate output can be sent to an output file. Howto perform it. Write a sample command line.

Ans.

- **ls -l; who > combined_output.txt**

r. Display all files in the current directory that begins with "a", "b" or "c" and are at least 5 characters long.

Ans.

- **ls | grep '^[abc].\{4,\}$'**

s. **Display all files in the current directory that begin and don't end with"x", "y" or "z"**
Ans.

- **ls | grep '^[xyz].*[^xyz]$'**

t. **Display all files in the current directory that begin with the letter D and have three more characters**.
Ans.

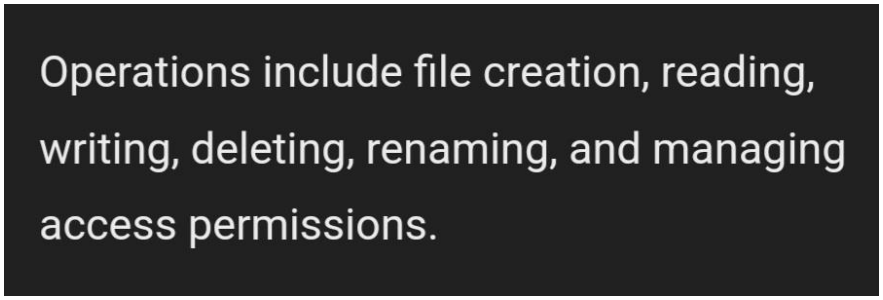
- **ls | grep '^D...$'**

u. **How to redirect td out of a command to a file, use the “>”**
Ans.

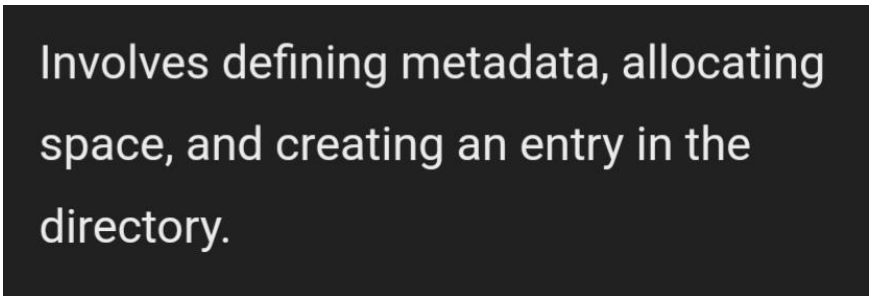
- **command > output_file**
- **ls > directory_list.txt**

Sample VIVA-VOCE Questions:

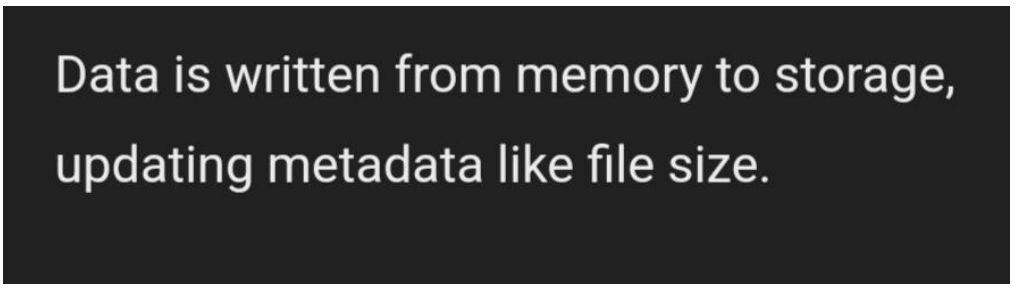
1. **What are file operations in an operating system?**



2. **Explain the process of file creation in an operating system.**

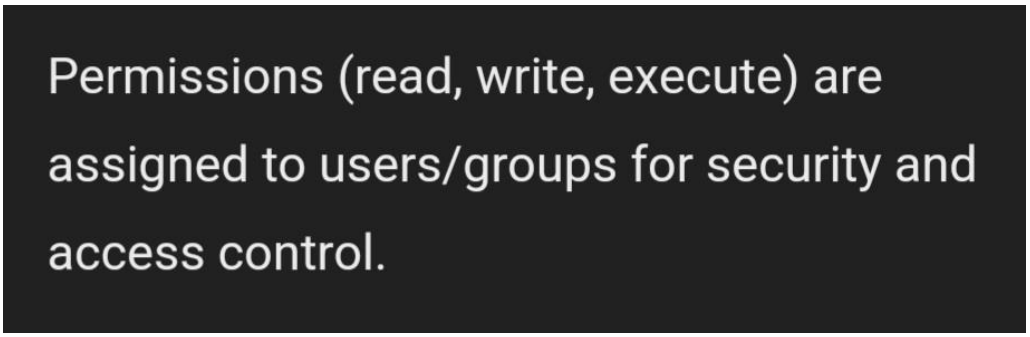


3. **Describe the file writing operation in an operating system.**

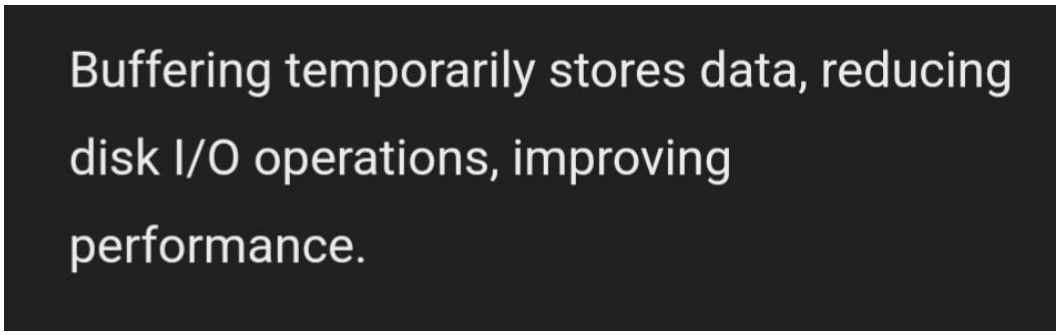


Sample VIVA-VOCE Questions:

4. How are file permissions managed in an operating system?

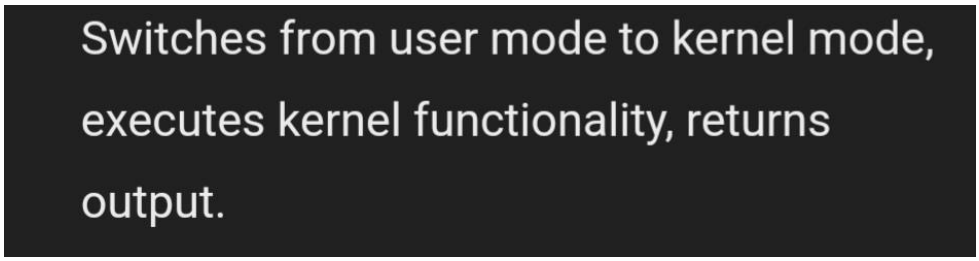


5 . What is the role of buffering in file operations, and how does it affect performance?

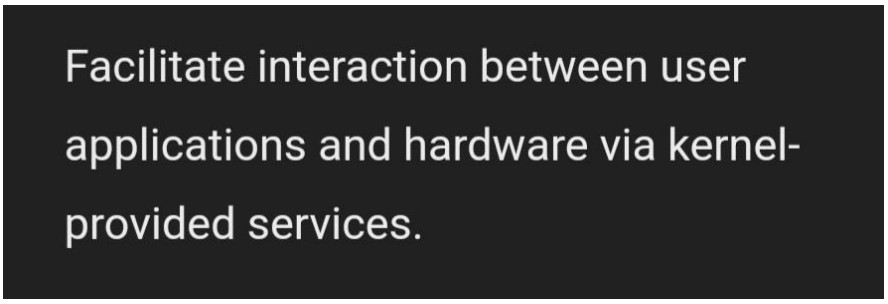


Sample VIVA-VOCE Questions:

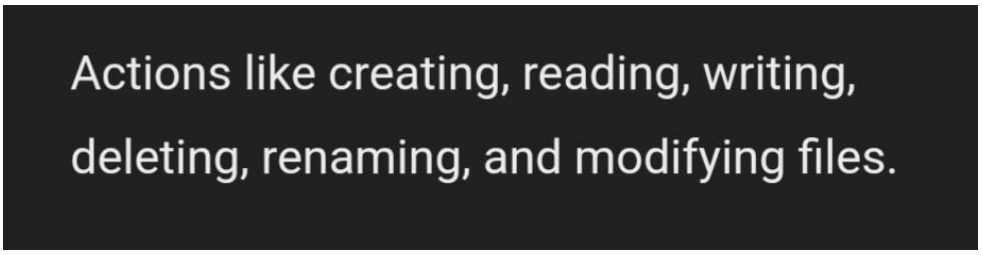
1. **How does a system call work?**



2. **Why do you need system calls in the Operating System?**



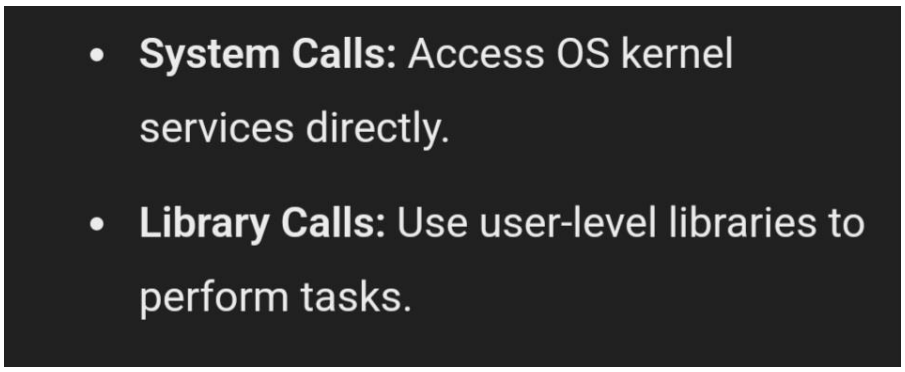
3. **What do you mean by file operations?**



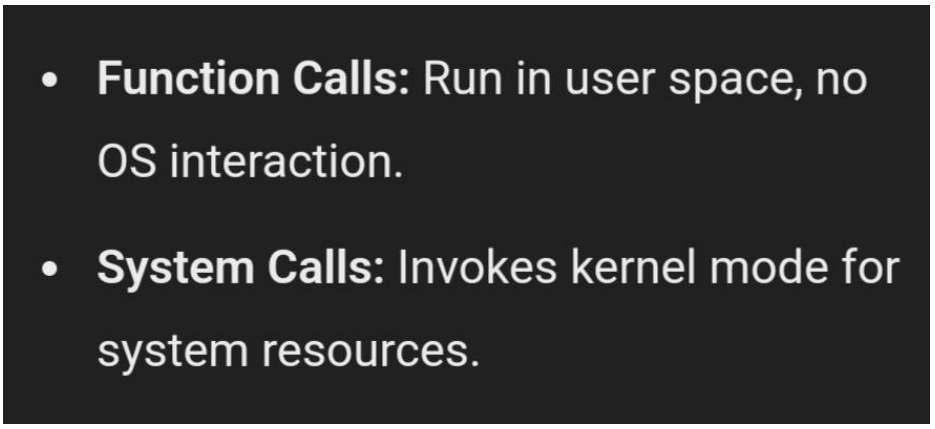
Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_BOT THANOS]

Sample VIVA-VOCE Questions:

4. **Differentiate between system calls and library calls.**



5. **Differentiate between function and system calls.**



Evaluator Remark (if any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Note: Evaluator MUST ask Viva-voce before signing and posting marks for each experiment.