Name of Student			
Lab Experiment No.	3.2	**Roll No.**	
Date Of Perf.:		**Date Of Sub.:**	
Expt. Title	To study file management in OS (System calls and Unix commands)		
CO Mapping	LO1,LO2,LO5		

Aim: To implement System Calls for File management

Objectives of the Experiment:

Implement the following system calls on files– creat, open, read, write, close, delete.

Theory:

File-I/O through system calls is simpler and operates at a lower level than making calls to the C file-I/O library. There are seven fundamental file-I/O system calls:

- creat() Create a file for reading or writing.
- open() Open a file for reading or writing.
- close() Close a file after reading or writing.
- write() Write bytes to file.
- read() Read bytes from file.
- remove() Delete a file

Use of these system calls requires a header file named "**fcntl.h**".

Creat() System Call

The "creat()" system call, of course, creates a file.
It has the syntax: <file descriptor variable> = creat(<filename>, <protection bits>);
This system call returns an integer, called a "file descriptor", which is a number that identifies the file generated by "creat()". This number is used by other system calls in the program to access the file. Should the "creat()" call encounter an error, it will return a file descriptor value of -1.

Open System Call

The "open()" system call opens an existing file for reading or writing.
It has the syntax: <file descriptor variable> = open(<filename>, <access mode>);
The "open()" call is similar to the "creat()" call in that it returns a file descriptor for the given file, and returns a file descriptor of -1 if it encounters an error. However, the second parameter is an "access mode", not a permission code.
There are three modes (defined in the "fcntl.h" header file):
O_RDONLY Open for reading only.
O_WRONLY Open for writing only.
O_RDWR Open for reading and writing.

Close () System Call:

The "close()" system call is very simple. All it does is "close()" an open file when there is no further need to access it.
The "close()" system call has the syntax: close(<file descriptor>);
The "close()" call returns a value of 0 if it succeeds, and returns -1 if it encounters an error.

Write System Call:
The "write()" system call writes data to an open file.
It has the syntax: write(<file descriptor>, <buffer>, <buffer length>);
The file descriptor is returned by a "creat()" or "open()" system call. The "buffer" is a pointer to a variable or an array that contains the data; and the "buffer length" gives the number of bytes to be written into the file.

Read System Call:
The read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);

Remove System Call:
The remove() deletes a name from the filesystem. It calls unlink(2) for files, and rmdir(2) for directories. On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

1. Write a program which will create a new file and write your personal information into this file.
2. Write a program which will read the contents of the file you have created and write it to a new file and will delete the old file.
3. Explore dup() and lseek()

Post Lab Assignment:

Explore following commands

1. head,tail, diff,cmp,uniq,comm.
2. zip, unzip, compress, uncompress, pack,unpack
3. file,find,locate,which,fsck
4. commands to create hard link and symbolic links.