FILE SYSTEM CALL

open()

- **Used to open an existing file for reading/writing or to create a new file.**
- **Returns a file descriptor whose value is negative on error.**
- **The mandatory flags are O_RDONLY, O_WRONLY and O_RDWR**
- **Optional flags include O_APPEND, O_CREAT, O_TRUNC, etc**
- **The flags are ORed.**
- **The mode specifies permissions for the file.**

creat()

- **Used to create a new file and open it for writing.**
- **It is replaced with open() with flags O_WRONLY|O_CREAT | O_TRUNC**

read()

- **Reads no. of bytes from the file or from the terminal.**
- **If read is successful, it returns no. of bytes read.**
- **The file offset is incremented by no. of bytes read.**
- **If end-of-file is encountered, it returns 0.**

write()

- **Writes no. of bytes onto the file.**
- **After a successful write, file's offset is incremented by the no. of bytes written.**
- **If any error due to insufficient storage space, write fails.**

close()

- **Closes a opened file.**
- **When process terminates, files associated with the process are automatically closed.**

Exp# 2a open system call

Aim

To create a file and to write contents.

Algorithm

1. Declare a character buffer *buf* to store 100 bytes.
2. Get the new filename as command line argument.
3. Create a file with the given name using open system call with O_CREAT and O_TRUNC options.
4. Check the file descriptor.
    a) If file creation is unsuccessful, then stop.
5. Get input from the console until user types Ctrl+D
    a) Read 100 bytes (max.) from console and store onto buf using read system call
    b) Write length of *buf* onto file using write system call.
6. Close the file using close system call.
7. Stop

Result

Thus a file has been created with input from the user. The process can be verified by using cat command.