Practical-1

Aim: Implementation of "cat" and "cp" command in C. (use of open, read, write, and close system calls)

Explanation:

Standard Values of Descriptors:

fd: Descriptor obtained from a file
0 : Standard Input (for example : Keyboard)
1 : Standard Output (for example : Terminal)
2 : Standard Error

Read System Call:

#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);

read() attempts to read up to *count* bytes from file descriptor *fd* into the buffer starting at *buf*.

On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number.

On error, -1 is returned, and *errno* is set appropriately.

Example: n=read(0,buff,sizeof(buff))

Write System Call:

#include <unistd.h>

ssize_t write(int *fd***, const void ***buf***, size_t** *count***);**

write() writes up to *count* bytes from the buffer pointed *buf* to the file referred to by the file descriptor *fd*.

On success, the number of bytes written is returned (zero indicates nothing was written). On error, -1 is returned, and *errno* is set appropriately.

Example: write(1,buff,n)

Task 1: Write a program to achieve following:

1. Read input from terminal
2. Display the information read on the terminal.

Task 2: Extend the above code to implement “cat” without options.

Open System Call:

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

int open(const char *pathname, int flags);

int open(const char *pathname, int flags, mode_t mode);

Given a *pathname* for a file, **open**() returns a file descriptor, a small, nonnegative integer for use in subsequent system calls.

The argument *flags* must include one of the following *access modes*:

O_RDONLY, **O_WRONLY**, or **O_RDWR**. These request opening the file read-only, write-only, or read/write, respectively.

In addition, zero or more file creation flags and file status flags can be bitwise-*or*'d in *flags*. The *file* creation flags are **O_CLOEXEC**, **O_CREAT**, **O_DIRECTORY**, **O_EXCL**, **O_NOCTTY**, **O_NOFOLLOW**, **O_TRUNC**, and **O_TTY_INIT**.

mode specifies the permissions to use in case a new file is created. This argument must be supplied when **O_CREAT** is specified in *flags*; if **O_CREAT** is not specified, then *mode* is ignored.

Example: fd=open(“test.txt”,O_RDONLY)

Close System Call:

#include <unistd.h>

int close(int fd);

close() closes a file descriptor, so that it no longer refers to any file and may be reused.

close() returns zero on success. On error, -1 is returned, and *errno* is set appropriately.

Example: close(fd)

Task 2:

Write a program to achieve following:

1. Read a file name from terminal.
2. Open the file and read the contents from the file.
3. Write the file contents on the terminal.

Assignments:

1. **Implement basic "cat" command using system calls.**

   Solution Logic:

   - The solution program should either execute without any argument or with any number of arguments.
   - When program executes without argument, it should echo the content given on standard input.
   - When program executes with one argument, it should print content of that file on standard output.
   - When program executes with more than one arguments, it should print content of each file one by one on the standard output.
   - To print the content of file, first program should open the input file in read mode, then it should read the file content and finally write it back to the standard output.

2. **Implement basic "cp" command using system calls.**

   Solution Logic:

   - This program should accept two arguments as source and destination file names.
   - It should open a source file in read mode.
   - It should open a destination file in read-write mode. If destination file does not exist, It should be created.
   - Finally program should write the content of source file into destination file.

Create a pdf with appropriate code and output and upload on moodle.

File name should be RollNo-Lab-1.