Assignment 1:   my_copy.c

Implement a cp(copy) command with –p option.

Pre-requisites:-

- Knowledge about system calls, How to read and understand 'man pages'.
- Command line arguments, File operation system calls (open, read, write, close, fstat ..etc)

Objective: -

- To understand and implement using basic system calls.

Requirements: -

1. Copy source file to destination file which passed through cmd-line arguments. After copying both files must have equal size, even it's a zero sized file.
   Eg: - ***./my_copy source.txt dest.txt.***
1. If arguments are missing show the usage (help) info to user.
2. Implement a my_copy() function where you have to pass two file descriptors.
   Int my_copy(int source_fd, int dest_fd);
3. If –p option passed copy permissions as well to destination (refer 'fstat' man page).
   Eg: - ***./my_copy -p source.txt dest.txt.***
4. If the destination file is not present then create a new one with given name. Incase if it is present show a confirmation from user to overwrite.
   Eg: -   ***./my_copy source.txt destination.txt***
   File "destination.txt" is already exists.
   Do you want to overwrite (Y/n)
5. If user type 'Y/n' or enter key overwrite the destination with new file. In n/N don't overwrite and exit from program.
6. Program should able handle all possible error conditions.

Sample execution: -

1. When no arguments are passed
   ./my_copy
   Insufficient arguments
   Usage:- ./my_copy [option] <source file> <destination file>
2. When destination file is not exists
   ./my_copy source.txt dest.txt
   New dest.txt file is created and source.txt file will be copied to dest.txt file
3. When destination file exists
   ./my_copy source.txt dest.txt
   File "dest.txt" is already exists.
   Do you want to overwrite (Y/n)
4. When –p option passed
   ./my_copy -p source.txt dest.txt
   Permissions also copied from source file to destination file

Also try:-

5. ***./my_copy /etc/hosts /etc/services***
6. ***./my_copy /dev/zero /tmp/new***

Useful commands:-

Man, cksum, ls –l

Assignment 2: my_wc.c

Implement a wc(word count) command with –l –w -c options.

Pre-requisites:-

- Knowledge about system calls, How to read and understand ‘man pages’.
- Command line arguments, File operation system calls (open, read, write, close ..etc)
- Working of wc command

Objective: -

- To understand and implement using basic system calls.

Requirements: -

1. Count the number of words, lines and characters(bytes) from files passed through command line.
2. If more than one files passed, print individual count values and file name + calculate the total of all values and print at the end.
3. If no file are passed wc will read from std input till end of file(**Ctrl + D**) then count lines, words and characters(bytes).
4. Implement a word_count() function where you have to pass fd and 3 integer addresses(pass by refference).   ***Int word_count (int fd, int *lines, int *words, int *bytes);***
5. Word_count function will read from the fd and calculates lines, words and bytes, then stores into respective addresses passed (don’t print values inside function).
6. Main function will open the files in a loop and call word_count function depends upon number of files passed. Print values after calling functions in main.

   Eg: -   ***./word_count file1.txt file2.txt***

```
for(I = 0; I < 2; i++)
{
        .
        .
        fd = open(…);
        word_count(fd, &lines, &words, &bytes);
        print_values(…);
        .
        .
}
Print_total(…);
```

7. If options passed [-l –w –c] print only respective values.
8. Program should able to handle all possible error conditions.

Sample execution: -

1. When no arguments are passed
   ./word_count
   Hai hello world
   <Ctrl + d>
   1 3 16   lines words characters
2. When one file passed
   ./word_count file.txt
   10 20 45
3. When no file passed.

```
./my_copy
Hello world hai (CTRL + D for end of file)
1   3   13
```

Reads from stdin till EOF (ctrl + d) and count lines words and bytes.

4. When options passed (-l, -w, -c).

```
./word_count file.txt -l -w
10   20
```

Prints according to given option. Option combination also should work. (Use **getopt** function)

Assignment 3:   dup.c

Write a program to understand usage of dup and dup2 system calls.

Pre-requisites:-

- Knowledge about system calls, How to read and understand 'man pages'.
- Command line arguments, File operation system calls (open, read, write, close ..etc)
- Working of dup system calls.

Objective: -

- To understand and implement using basic system calls.

Requirements: -

1. Using dup or dup2 redirect printf out to a given file instead of printing to stdout.
2. Pass the file name using command-line arguments.
3. Try using both system calls (dup and dup2).

Assignment 4:   fcntl_lock.c

Write a program to understand advanced file control system calls.

Pre-requisites:-

- Knowledge about system calls, How to read and understand 'man pages'.
- Command line arguments, File operation system calls (open, read, write, close ...etc)
- Working of fcntl system calls.

Objective: -

- To understand and implement using advanced system calls.
- Understand the need of file synchronization between processes.

Requirements: -

1. Using fcntl system call synchronize a file between two processes (parent and child process).
2. Pass the file name using command-line arguments.
3. Before writing to file check file is locked, in case it is locked must wait the process until its unlocked.
4. If its unlocked, lock file and continue with writing.
5. Both process will do the same procedure.