Week-4
Assignments

File Management System Calls

1. `int creat(char* filename, mode)`
2. `int open(char* filename, int access, int mode);`
3. `size_t read(int fd, char *buffer , size_t bytes);`
4. `size_t write(int fd, char *buffer, size_t bytes);`
5. `int close(int fd);`
6. `long lseek(int fd,int offset,int origin)`
7. `int mkdir(char* name, int mode);`
8. `int rmdir(char* name);`
9. `int chmod(const char *path, mode_t mode);`
10. `DIR *opendir(char* dir_name)`
11. `s struct dirent *readdir(DIR* dp)`
12. `stat(char* file, struct stat *buf); fstat(int fd, struct stat *buf)`
13. `char* getcwd(char * dirname, int );`

a) Write a function foo(int fd, char* buf, int b_size, int n, int skip) that reads to buf from file with file descriptor fd, n blocks of size b_size each. The last argument specifies how many bytes to skip after reading each block. Return -1 if the operation is unsuccessful. Else return total number of bytes read.

b) Write a program to read all txt files (that is files that ends with .txt) in the current directory and merge them all to one txt file and returns a file descriptor for the new file.

c) Write a program that takes a string as an argument and return all the files that begins with that name in the current directory and its sub directories. For example > ./a.out foo will return all file names that begins with foo.

d) Write a program that will categorize all files in the current folder based on their file type. That is all .txt file in one folder called txt, all .bmp files in another folder called bmp etc. The argument to the program is a folder name.

e) Given a directory, write a program that will find all files with the same name in the directory and its sub directories. Show their name, which folder they are in and what day they were created. Expand the program to remove all duplicate copies based on user input. That is, ask user if each one of the file is to be kept or deleted. Based on user input, perform the appropriate action.

Process Management System Calls

1. fork() or vfork()
2. exec() - Family of System Calls
3. wait() or exit()

a. Create two functions: one to generate the list of Odd_Numbers b/w 'n' and'm', and another to generate Even_Numbers b/w 'n' and'm'. In order to invoke the aforementioned techniques, construct a C programme that creates two processes, one of which calls the Even_Number list and the other the Odd_Number list.

b. Develop two distinct programmes. One should be used to generate Fibonacci Series upto 'n' terms (let's say Fibo.c), and the other should produce a list of PrimeNumbers b/w 'n' and 'm' (let's say Prime.c). Now develop a C programme that uses fork() to split into two processes, one that runs Fibo.c and the other that runs Prime.c.