LAB
WEEK 9

Append

Difference between >> and >

Try this:

```
Echo    Good Students    >>    file.txt
Echo    “I replaced everything ☹”    >   file.txt
```

Exercise 1

Make a file program.txt and append stderr of ex1.txt to the same program.txt file.

Linking Files

Linking connects file name to data. More than one file name can be linked to same data.

Command for creating a hard link:

```
$ln    OriginalFileName    NewFile/LinkName
```

Hard links

A directory may contain several filenames that all map to the same **inode** number and thus to the same file in the file system. Unix call these names **pointers** or **links** to the file.

Hard links are new names for the same inode. Link count in the inode keeps track of how many directories contain a name number mapping for that inode.

Hard links cannot be made to a directory. This restriction means that every sub directory has one parent directory.

Hard Links

A hard link and data it links to, must always exist in the same file system.

// To find inode number of files: ***$ls -i filename***

Exercise 2

Create ex2_file.txt. Link it to ex2_link1.txt and ex2_link2.txt. Check inode numbers of ex2_file.txt, ex2_link1.txt and ex2_link2.txt.

Exercise 3

- Create file1.txt in week1 directory and access this file from week9 directory by link ex3.txt.
- Trace all links to file1.txt [***$find –inum inodenumberofthefile***].
- Remove all links from file1.txt [***exec rm{}***]

Soft Links

A soft link or symbolic link contains a path of another file or directory.

// May point to any file or directory.

// Can cross file systems and link to directories.

// Created by ***$ls –s original_file_name link_name***

File Permissions

Read (r) -> with read permission we can see the contents of the file.

Write (w) -> allows us to change the file such as add to a file, overwrite it etc.

Execute (x) -> with execute permission we can ask the operating system to run the program.

Directory Permissions

Read -> we can list the contents of the directory

Write-> Add, Rename and move files in the directory

Execute-> (sometimes called search permission).
we can list information about the files in the directory.

User, Group, World

The read, write and execute permissions are stored in three different places called user (u), group (g), world/other (o).

All (a) means ugo.

There are three sets of rwx and determined by 9 bits of i-node information.

```
$chmod u=rwx filename
$chmod g=rwx filename        or        $chmod a=rwx filename
$chmod o=rwx filename
```

Octal (Numeric) Permissions

r--	4	Read
-w-	2	Write
--x	1	Execute
---	0	No permissions/access

Exercise 4

- Write a C program to make a directory called “ex4_dir” with read and execute permissions to user, group and

  others.
- Create a soft link to ex4_dir.
- Remove Soft link.
- Remove ex4_dir.

// Hint: man 2 mkdir, man 2 stat, man 2 symlink.

Exercise 5

Make a file “ex5.txt” and try the following exercises:

1. Remove execute permission for everybody
2. Grant all permissions to *uo*.
3. Make group permissions equal to user permissions.
4. What does 660 mean for ex5.txt?
5. What does 775 mean for ex5.txt?
6. What does 777 mean for ex5.txt?

File Pointers, File Structures and FCBs

Opening a file returns a file pointer to a file structure defined in <stdio.h> that contains information used to process the file.

File Structure includes a file descriptor which is an index into an OS array called the open file table.

Each array element contains FCB that OS uses to administer a particular file.

Creating a File pointer

FILE *myPtr;

creates a FILE pointer called myPtr

myPtr = fopen("myFile.dat", openmode);

Function fopen() returns a FILE pointer and takes two arguments – file to open and file open mode.

fclose(FILE pointer)

Function fclose() closes specified file.

// Even though it's performed automatically when program ends, it is good practice to close files explicitly.

File Modes

MODE	Purpose	If file doesn’t exist, will it be created?	What happens to the EXISTING FILE?
"a"	Appending	Yes	Appended to
"a+"	Reading/appending	Yes	Appended to
"r"	Read only	No	Reading
"w+"	Reading/writing	Yes	Discarded

End of File

End of file indicator informs the program that there is no more data to be processed.

feof(file stream)

Linux/Mac OS X/UNIX	<ctrl> d
Windows	<ctrl> z

Exercise 6

Write a C program that writes variables of different data types(int, char, float) into a file and then read file sequentially.

Use fprintf() and fscanf().

Random Access
fseek(), ftell(), rewind()

fseek() – positions the file pointer.

fseek(file pointer, 0L, SEEK_SET);

SEEK_SET – from the start.

SEEK_CUR – from the current position.

SEEK_END – from the end of the file.

ftell() – return the current offset of the file pointer.

long offset = ftell(FILE * f)

rewind() – return the file pointer to the beginning.

Exercise 7

Write a program that takes character input until EOF hits. Using file pointer put character input into a file. Print the current position of file pointer using ftell().

Start from the beginning and print the 3rd location using fseek().

Start from the end and print 5th location again using fseek().