



Lecture – 10
Section-B

Theoretical concept of Unix
Operating System

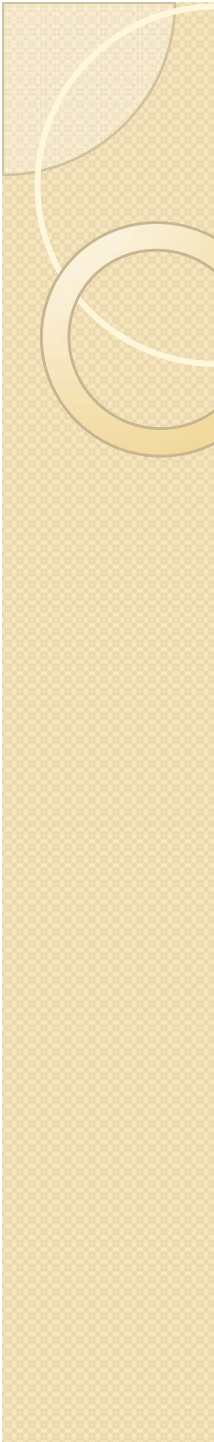


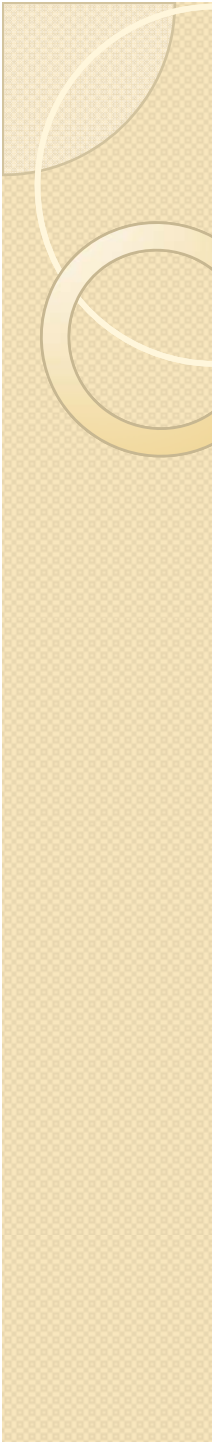
Introduction

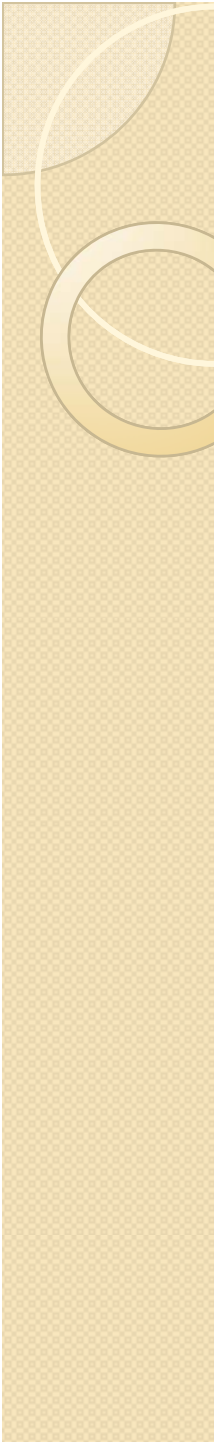
- File system and Inodes
 - Blocks & File system
 - Directory Structure
 - UNIX commands
-

File System and Inodes

- The hard disk is split into distinct partitions(or slices), with a separate file system in each partition(or slice).
- Every file system has a directory structure headed by root.
- If you've three file systems on one hard disk, then they will have three separate directories.
- When the system is up, we see only a single file system with a single root directory.
- Of these multiple file systems, one of them is considered to be the main one, and contains most of the essential files of the UNIX system, this is the root file system, its root directory is also the root directory of the combined UNIX system.
- At the time of booting, all secondary file systems attach themselves to the main file system, creating the illusion of a single file system to the user.

- 
- Every file is associated with a table that contains all that you could possibly need to know about a file – except its name and contents.
 - This table is called the **inode** (shortened from index node) and is associated by the **inode number**.
 - The inode contains the following attributes of a file:
 1. File type (regular, directory, devices, etc)
 2. File permissions (read, write etc).
 3. Number of links.
 4. The UID of the owner.
 5. The GID of the group owner.
 6. File size in bytes.
 7. Date and time of last modification.
 8. Date and time of last access.
 9. Date and time of the last change of the inode.
 10. An array of pointers that keep track of all disk blocks used by the file.

- 
- Observe that neither the name of the file nor the inode number is stored in the inode. It's the directory that stores the inode number along with the filename. Every file system has a separate portion set aside for storing inodes, where they are laid out in a contiguous manner. This area is accessible only to the kernel. The inode number is actually the position of the inode in this area. The kernel can locate the inode number of any file using simple arithmetic.
 - **Since a UNIX machine usually comprises multiple file systems, you can conclude that the inode number for a file is unique in a single file system.**
 - Thus **ls** command reads the inode to fetch a file's attribute, and it can list most of them using suitable



One of them is the `-l` (inode) option that tells you the inode number of a file :

```
$ ls -l tulec05
```

```
9059 -rw-r--r- kumar metal 51813 Jan 31 11:15 tulec05
```

The file `tulec05` has the inode number 9059. No other file. No other file in the same file system can have this number unless the file is removed. When that happens , the kernel will allocate this inode number to a new file.

Blocks and File System

Every file system has four components –

- **The Boot Block** – this block contains a small boot program and the partition table.
- **The Super Block** – it contains global information about the file system. Additionally it also maintains a free list of inodes & data blocks that can immediately allocated by the kernel when creating a file.
- **The Inode Blocks** – this region contains the inode for every file of the system.
- **The data blocks** - All data and programs created by users reside in this area.

The Boot Block :

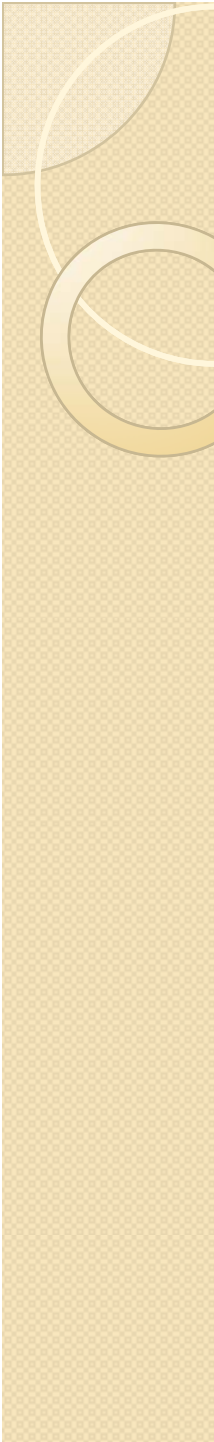
- The first block of a file system is known as the Boot Block, containing a small boot strap program – referred to as the “**Master Boot Record**”.
- This program is loaded into memory when the system is booted (hence the name).
- It may in turn, load another program from the disk, but eventually it loads the kernel into memory.
- However, the boot-strapping program is read in from the Boot Block of only one file system (the main one, called the root file system).
- For other file system, this block is simply kept blank.

The Super Block :

- The boot block is followed by the “super block, the balance sheet of every UNIX file system.
- It contains global file information about disk usage and availability of data blocks and Inodes.
- Its information should, therefore, be correct for healthy operation of the system.

This mainly contains –

- The size of the file system.
- The block size used by the file system.
- The number of free data blocks available & a partial list of immediately allocated free data blocks.
- Number of free inodes available & a partial list of immediately usable inodes.
- Last time of updating.
- The state of the file system (whether clean or dirty).

- 
- UNIX refuses to boot if the super block is corrupt to overcome this problem, many systems like (Solaris & Linux) have multiple super blocks written on different areas of disk.
 - If one super block is corrupt, the system can be directed to use another.

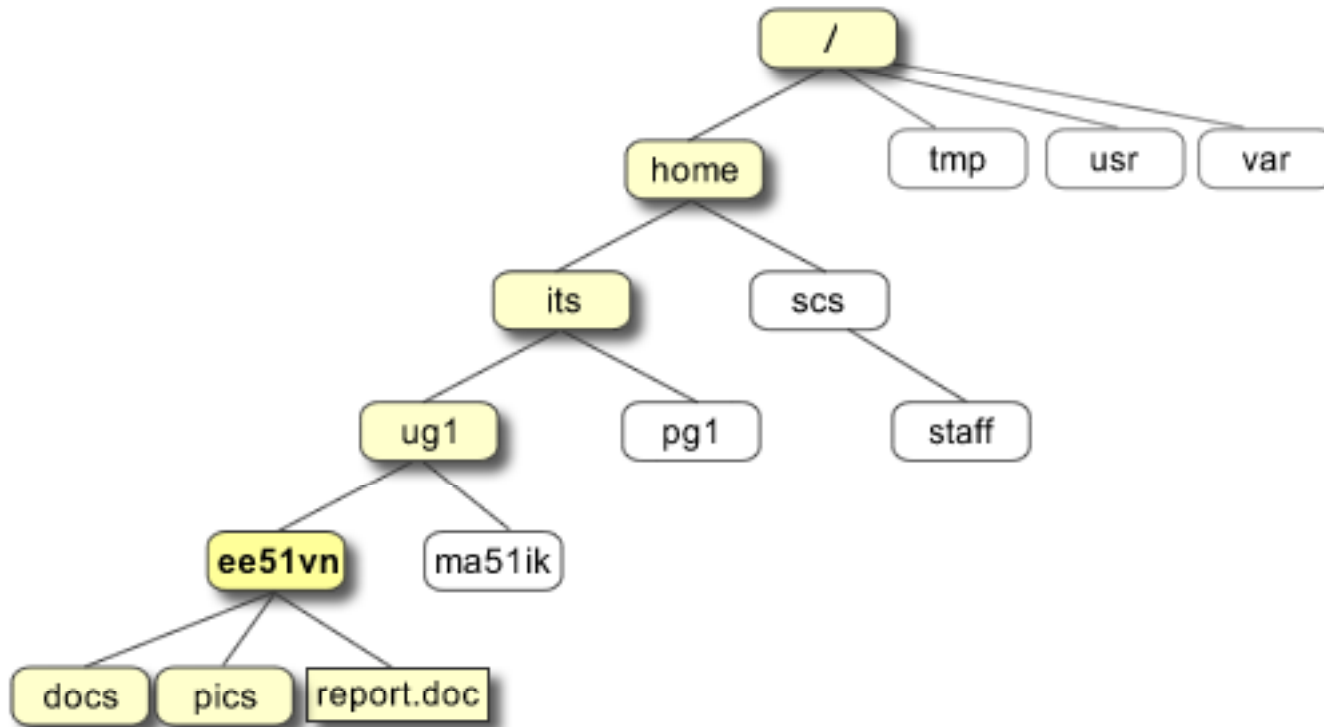
Inode Blocks :

- The Inode block contains all inodes. When a file is created, an inode is allocated here. Even though the inode contains all file attributes (except its name) ;it doesn't store the inode number.
- Since the inodes are laid out sequentially, the kernel uses simple arithmetic to identify the inode by using the inode number as the index in this list.
- The ls command looks up directory to know the inode number of a file; and then reads the

Data Blocks :

- All the file data are stored in the data blocks, which occupy most of the file system.
 - The data blocks are normally called as physical block size – Unix tools assume a block size of 512 bytes (1024 in Linux).
-
- We call this the logical size block, and it is set at the time of installation of the system.
 - This block size also determines the largest file size that the system can accommodate.
 - When a file expands the kernel may not find its adjacent blocks free.
 - The remaining data then have to be written to the next free block; wherever it may be.

Directory Structure



The Unix file system tree

/home/its/ug1/ee51vn/docs

Unix Commands used in Directory Structure

(1) THE HOME DIRECTORY

- When you login to the system, UNIX automatically places you in a directory called the **home directory**.
- It is created by the system when a user account is opened.

(2) **pwd** : CHECKING YOUR CURRENT DIRECTORY:

- In UNIX, just like a file, a user is placed in a specific directory of the file system on logging in.
- You can move around from one directory to another, but at any point of time, you are located in only one directory. This directory is known as **current directory**.
- At any time, you should be able to know what your current directory is. The **pwd** (print working directory) command tells you that :

```
$ pwd
```

```
/home/kumar
```

(3) cd : CHANGING THE CURRENT DIRECTORY

- You can move around the file system by using the **cd** (change directory) command. When used with an argument, it changes the current directory to the directory specified as argument, for instance, progs :

```
$ pwd
```

```
/home/kumar
```

```
$ cd progs
```

progs must be in current directory.

```
$ pwd
```

```
/home/kumar/progs
```

The command **cd progs** here means this : “Change your subdirectory to progs under the current directory” Using a pathname causes no harm either; use **cd /home/kumar/progs** for the same effect

(5) mkdir : MAKING DIRECTORIES

- directories are created with the **mkdir** (make directory) command. The command is followed by names of the directories to be created.
- A directory patch is create under the current directory like this :

\$mkdir patch

- You can create a number of subdirectories with one mkdir command:

\$ mkdir patch dbs doc

rmdir: REMOVING DIRECTORIES

- The rmdir command is used to delete directories.
- The format of this command is:
% rmdir directory_name
- For eg. Lets store files for project work in a directory called proj.
- When the project has been completed we deletes the directory using the command:
% rmdir proj
- ***Note that the directory must be empty before it can be deleted.***

Consider a directory path :- /home/kumar/pis

```
$ cd progs
```

```
$ pwd
```

```
/home/kumar/pis/progs
```

```
$ rmdir /home/kumar/pis/progs
```

Trying to remove the current

directory

```
rmdir: directory "/home/kumar/pis/progs " : Directory does not exist.
```

To remove this directory, you must position yourself in the directory above progs, i.e. pis & then remove it from there:

```
$ cd /home/kumar/pis
```

```
$ pwd
```

```
/home/kumar/pis
```

```
$ rmdir progs
```

Applications

Compilers, interpreters and programming tools

- csh command language interpreter (C-shell scripts)
- ksh command language interpreter (Korn-shell scripts)
- sh command language interpreter (Borne-shell scripts)
- f77 Fortran 77 compiler
- f2c convert fortran source code to C source code
- gcc GNU C compiler
- g++ GNU C++ compiler
- dbx command-line symbolic debugger for compiled C or Fortran
- make recompile programs from modified source
- cflow generate C flow graph

Research

- CCR's computing resources are primarily Linux based and therefore using them requires a basic understanding of the Unix operating system. Some basic commands are provided below.
- **Basic Unix Commands**
- CCR Reference Card for Linux/UNIX commands [pdf](#)
- Show pathname of current directory: pwd
- List files: ls
- Make a directory: mkdir directory-name
- Change directory: cd directory-name
 - Change directory back to home directory: cd
- Copy a file: cp old-filename new-filename
- View a file:
 - cat filename
 - more filename
 - less filename
- Edit a file:
 - emacs filename
 - vi filename
- Delete a file: rm filename
 - Delete a directory (recursively): rm -R directory-name
 - All files and subdirectories are deleted
- Move a file: mv old-filename new-filename
- Change permissions:
 - Arguments to chmod command: ugo+-rwx
 - where ugo are user, group and other; rwx are read, write and execute
 - Add execute permission for yourself: chmod u+x filename
 - Remove read, write and execute for group and other from a directory its contents: chmod -R go-rwx directory-name