File System Handout

Zilogic Systems

1. Files

1.1. Types of Files

- Files in GNU/Linux can be broadly classified as follows:
  1. Regular files
  2. Directories
  3. Links, similar to M$ Windows Shortcuts
  4. Device Files
- In GNU/Linux every device is represented by a file in `/dev` directory. For example `/dev/sda` represents the hard disk, `/dev/ttyS0` represents the serial port, `/dev/input/mice` represents the mouse, ...
- Reading/writing to and from the device file, results in reading from/writing to the device. To get mouse events the following command can be used.

```
# cat /dev/input/mice
```

- The command initial blocks, displaying nothing. But when the mouse is moved, data representing mouse movement events are displayed on the screen.

1.2. Filenames

- The filename can have maximum of 256 characters.
- The filename can consist of any character except `/` — the directory separator.
- Filenames are case sensitive.
- The concept of extensions does not exist at the file system level. A filename can have an extension but it is not mandated by the file system.
- Applications may or may not recognize and use extensions.
- If files do not have extensions, the file type can be identified using the `file` command. Example:

```
$ file my-document
```

- Files that start with a period are hidden files.
- Hidden files can be viewed using the `-a` option to `ls` as shown below.

```
$ ls -a
```

Try Out

- Goto the directory `file-type` under the home directory.
- For each file in the folder find out the type of the file using the `file` command.
- Check if there are any hidden files in the directory.

1.3. Text File and Binary Files

- The `cat` command should be used on the file only when the `file` command it reports it is a text file.
- The behaviour of terminals can be controlled using special sequence of characters.
- For example, certain sequence of characters can be used to change the color of the text displayed in the terminal.
- When a binary file is displayed using `cat`, it might contain the special sequence of characters, and can cause the terminal to get corrupted.
- To restore the terminal the reset command can be used.

Try Out

- Goto the directory `colors` under the home directory.
- `cat` each file to change the color of the terminal.
- Goto the directory `file-type` under the home directory.
- Type `cat myfile4`, to display the contents of the binary file.
- The terminal will get corrupted. Close your eyes :-) and type `reset` to restore the terminal back to its original state.

2. Directory Structure

- Every operating has a way of laying out different categories of files in the filesystem.

Category	**Folder**
User's Files	`C:\Documents and Settings`
Application Programs	`C:\Program Files`
System Programs	`C:\Windows`
Temporary Files	`C:\Windows\Temp`
System Configuration	`C:\Windows\System32\Config`

Category	**Directory**
User's Files	`/home`
Binaries	`/bin`, `/usr/bin`
System Binaries	`/sbin`, `/usr/sbin`
Kernel, Bootloader	`/boot`
Libraries	`/lib`, `/usr/lib`
Temporary Files	`/tmp`
Configuration Files	`/etc`
Help Files	`/usr/share/doc`
Architecture-independent application files	`/usr/share`
Variable data files	`/var`
Device nodes	`/dev`
Processes and Kernel information	`/proc`

Figure 1. Windows XP Layout Tree

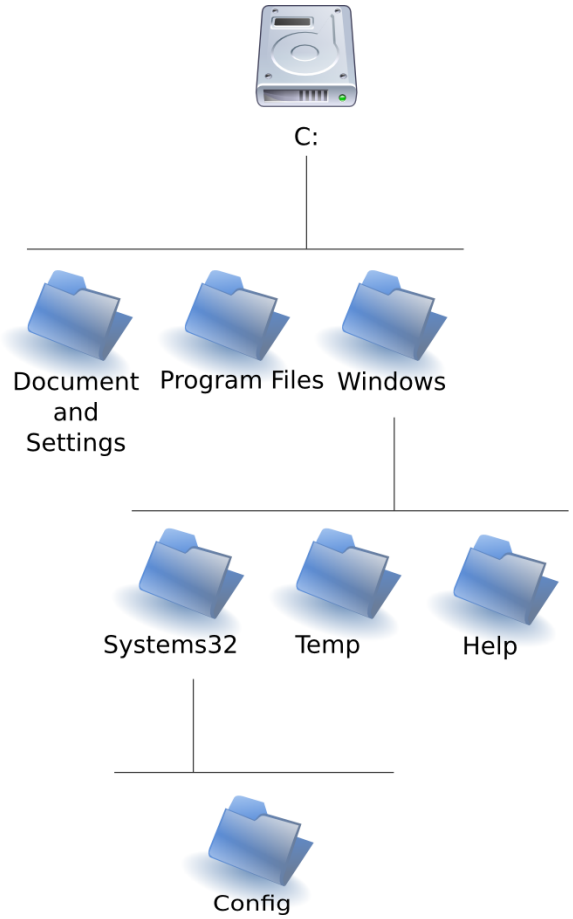
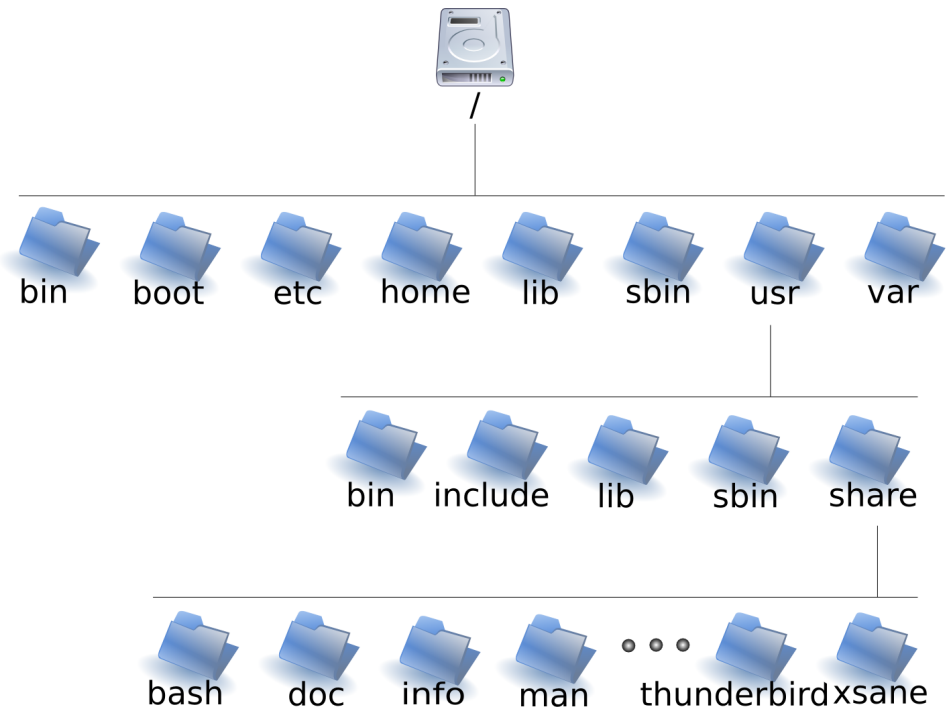


Figure 2. GNU/Linux Layout Tree



2.1. Advantages

- `/usr` - can be mounted from a remote NFS.
- `/usr` - contains only static files, can be in a read-only partition.
- `/var`, `/tmp` - can be in a separate partitions - rest of the disk is not fragmented.
- `/boot` - can be in a separate parition, that is accessible to the boot loader.
- `/home` - can be a separate partition, shared by a group of systems, not affected when the system is re-installed.

2.2. Local software

- `/usr/local` - software that are not managed by the package manager.
- Protected from system software upgrades.

Try Out

- Find out the location of the `cat` and `ls` executable.
- Find out the location of `firefox` executable.
- Find out the location of the C library file `libc.so.6`.
- Find out the location of the icons used by the program `iceweasel`.
- Find out the location of the kernel log file `kern.log`.
- Goto `/proc`. Type `cat cpuinfo`, to get information about the processor exported by the kernel.

3. Pathname

- The pathname specified where in the hierarchy, a file is located.
- Absolute paths, start with a `/`. Example: `/usr/share/firefox`.
- Relative paths, are relative to current working directory.
- Relative paths can go both ways in the hierarchy.
- Relative path `doc/iceweasel` goes down the hierarchy.
- Relative path `../../boot` goes up the hierarchy.
- Example commands:

```
$ ls doc/iceweasel
$ ls ../../boot
```

Try Out

- Create a file called `myfile` under `science/biology/botany`.
- Change working directory to `science/physics`.
- Copy the file created under `botany` to current working directory, using absolute path. Command: `cp /home/xxx/science/biology/botany/myfile .`
- Remove the copied file and repeat using relative path. Command: `cp ../biology/botany/myfile .`
- Goto the `botany` directory.
- Remove the copied file using relative path. Command: `rm ../../physics/myfile`

4. Searching

4.1. Searching Files ...

- The `find` command is used to search for files.
- The `find` command has lot of options for searching and filter.
- In its most commonly used form, it has the following general syntax.

```
find <path> -name <pattern>
```

- The `path` specifies the directory under which the file is to be searched for.
- The `pattern` is an argument to the `-name` option, and specifies a wildcard pattern.
- All files matching the wild card pattern will be printed on the screen.
- For example, to file all JPEG files under `/usr`, the following command can be used.

```
$ find /usr -name "*.jpg"
```

- For more advanced usage, the general syntax is as follows.

```
find <path> [<expression>]
```

- The `expression` is set of tests that specifies the filter criteria.
- `-name` is one of the test that can be performed.
- `-type` checks for the file type. `-type d` matches directories, `-type f` matches files, etc.
- When more than one test is specified then `find` will check if all the tests are satisfied by a file.
- This behaviour can be changed by specifying `-o` option, which indicates that tests have to be logically ORed instead of being ANDed. Logically ANDing can also be explicitly specified using the `-a` option.
- All JPEG and PNG files under `/usr` can be printed using the following command.

```
$ find /usr -name "*.jpg" -o -name "*.png"
```

- All directories with name `doc` can be printed using the following command.

```
$ find /usr -name doc -a -type d
```

Try Out

- Using `find` determine the locations of all PDF files in the system.
- Using `find` determine the location of all directories called `bin` and `sbin` in the system.

4.2. Accelerating Searches

- `find` walks through the file system, searching for files.
- `find` is slow — the data scattered in disk.
- Solution: database of files
- Build, search, update DB
- Usage:

```
locate <pattern>
```

- Searches from root

Try Out

- Using `locate` find out the location of all JPG files in the system.

5. Permissions

5.1. Owners and Groups

- Multi-user OS
- Each user has a username
- Need to simplify user privileges management
- Example: Printer privileges
- Users are placed into groups — `groups` command
- Each user has a main group
- Each file has a owning user and owning group
- By default, creator of file - owning user
- Main group of creator - owing group
- To find the owning user and owning group `ls -l`

Owner Output screenshot.

```
                ❶          ❷
-rw-r-----  1 root        adm     729 2008-01-22 22:54 user.log.2.gz
-rw-r-----  1 root        adm     249 2008-01-03 13:42 user.log.3.gz
-rw-r--r--  1 root        root      0 2007-10-07 04:48 uucp.log
-rw-rw-r--  1 root        utmp 105600 2008-02-11 12:22 wtmp
-rw-rw-r--  1 root        utmp 113664 2008-02-02 07:04 wtmp.1
-rw-r--r--  1 root        root  61308 2008-02-11 14:57 Xorg.0.log
-rw-r--r--  1 root        root  59427 2008-02-06 11:37 Xorg.0.log.old
```

❶ Owning user of the file
❷ Owning group of the file

- `chown` command - to change owner
- only superuser can change owning user
- user quotas is based on file ownership
- the owning user can change owning group to any one of his groups

```
# chown <owner>:<group> <file>
```

5.2. Permissions

- Each file - perms for
  - owning user
  - owning group
  - others

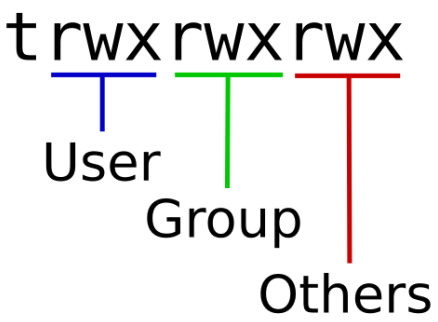
- To see permissions `ls -l`

Permissions Output Screenshot.

```
❶
-rw-r-----  1 root        adm     729 2008-01-22 22:54 user.log.2.gz
-rw-r-----  1 root        adm     249 2008-01-03 13:42 user.log.3.gz
-rw-r--r--  1 root        root      0 2007-10-07 04:48 uucp.log
-rw-rw-r--  1 root        utmp 105600 2008-02-11 12:22 wtmp
-rw-rw-r--  1 root        utmp 113664 2008-02-02 07:04 wtmp.1
-rw-r--r--  1 root        root  61308 2008-02-11 14:57 Xorg.0.log
-rw-r--r--  1 root        root  59427 2008-02-06 11:37 Xorg.0.log.old
```

❶ Permissions bits for the owning user, owning group and others

Figure 3. Permission Bits



- Directories and permissions
  - `r-x` - write protected
  - none - no access
  - other combinations - rarely used
- change perms - `chmod`

`chmod` Examples.

```
$ chmod u+w myfile
$ chmod g+rw myfile
$ chmod ugo+x myfile
$ chmod o-rwx myfile
$ chmod ugo=rw myfile
```

- Only owning user or root can change the permissions

Try Out

- Create two files `out.txt` and `in.txt`, in your home directory.
- Give permission to the two files, such that your friend can read from `out.txt` and can write to `in.txt`.

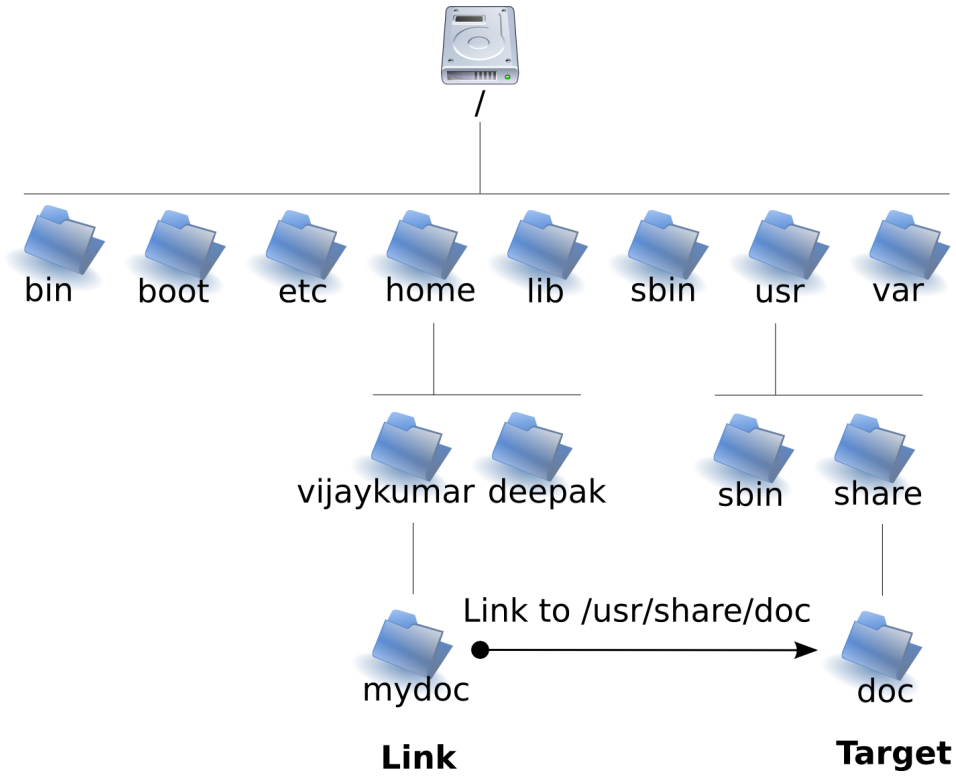
- Write a message to your friend in `out.txt`.
- Go to your friends directory, and read his message in his `out.txt` file. And write a response for him in his `in.txt` file.

6. Links

- Similar to M$ Windows Shortcuts
- Same file in two locations
- Two types of Links
  - Hard Link
  - Symbolic Link
- Hard Links - rarely used by users
- Used internally by OS to implement `.` and `..`
- Limitations on files that can be linked
- Symbolic Links created using `ln`
- Usage:

```
ln -s <target> <link>
```

Figure 4. Example file tree with links



Absolute Links.

```
project
|
+- source
|  +- main.c
|  +- lib.c
```

```
|
+- include
+- doc
   +- link -> /home/vijaykumar/project/source/lib.c
```

Relative Links.

```
project
|
+- source
|  +- main.c
|  +- lib.c
|
+- include
+- doc
   +- link -> ../source/lib.c

project
|
+- source
|  +- subdir
|  |  +- main.c
|  |
|  +- lib.c
|
+- include
+- doc
   +- link -> ../source/lib.c *broken*
```

Try Out

- Create a symbolic link in your home directory to the file `/proc/cpuinfo`.
- Repeat the above using relative path for the link. Command: `ln -s ../../proc/cpuinfo cpu`.
- Move the file to another directory within your home directory.
- Try accessing the file now.

7. Advanced Search

- Search by owning user, `-user`
- Search by owning group, `-group`

8. Mount Points

- M$ Windows - separate root for each drive
- Each drive is identified by separate letter
- GNU/Linux - singly rooted hierachy
- Each drive's tree is grafted on to main tree
- Main tree - root file system
- Point of grafting - mount point

Figure 5. Before mounting

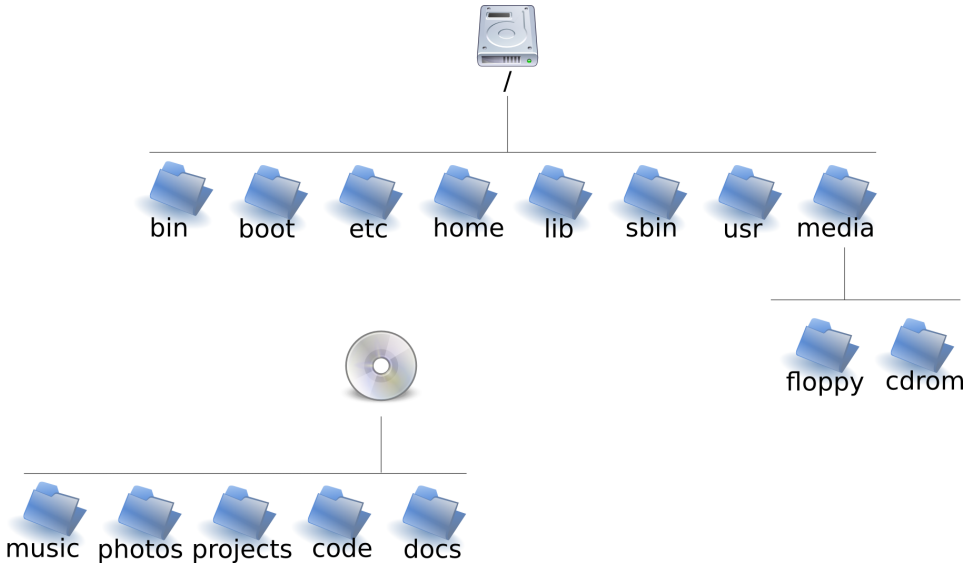
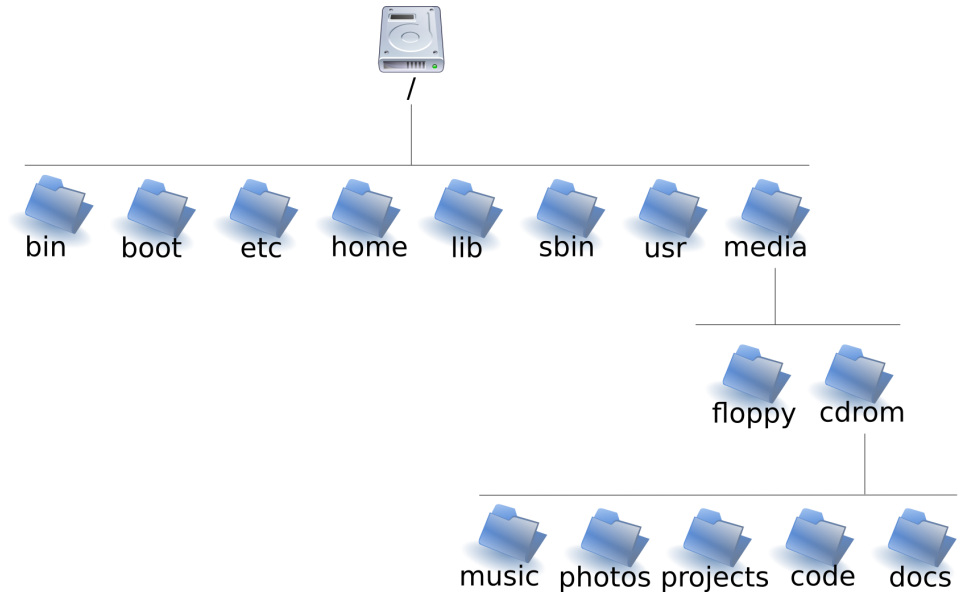


Figure 6. After mounting



9. Disk Space

- Disk Usage, du
- Summary of disk usage of files and dirs

`du` Invocation Example.

```
vijaykumar@trinity:char$ du -h
968K    ./ipmi
564K    ./ip2
500K    ./pcmcia
4.9M    ./drm
124K    ./tpm
1.4M    ./agp
488K    ./rio
```

```
324K    ./hw_random
2.4M    ./watchdog
364K    ./mwave
19M     .
```

- Disk Free, df
- Summary of free space available in file systems

`df` Invocation Example.

```
vijaykumar@trinity:~$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda3             258M  141M  104M  58% /
tmpfs                 498M     0  498M   0% /lib/init/rw
udev                   10M  116K  9.9M   2% /dev
tmpfs                 498M     0  498M   0% /dev/shm
/dev/sda9              31G   12G   18G  40% /home
/dev/sda8             372M   14M  339M   4% /tmp
/dev/sda5             4.6G  2.4G  2.0G  55% /usr
/dev/sda6             2.8G  337M  2.3G  13% /var
```

10. Further Reading

- Debian Reference: GNU/Linux tutorials: Unix-like filesystem - http://www.debian.org/doc/manuals/reference/ch-tutorial.en.html
- Filesystem Hierarchy Standard - http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
- FreeBSD Handbook: Getting Started: Unix Basics: Users and Basic Account Management - https://www.freebsd.org/doc/handbook/users-synopsis.html
- FreeBSD Handbook: Getting Started: Unix Basics: Permissions - https://www.freebsd.org/doc/handbook/users-synopsis.html
- `man 7 hier`