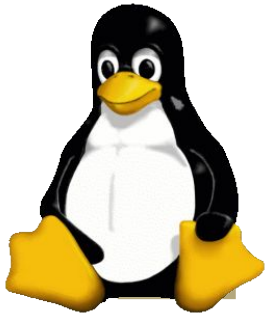


Linux commands

Maruthi S. Inukonda

[01]9th Jan 2019

Agenda

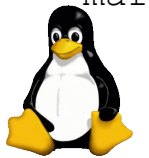
- Login, Session, Logout
- Manual pages
- Bash shell
- Users and Groups
- Process, Jobs
- File system basics
- Discretionary Access Control
- File operations
- Searching files and directories
- Redirection, Pipes and Filters

Login, Session, Logout

Login

- For secure CLI login to a remote Unix/Linux system, use
  `ssh <username>@<fqdn>` Or `ssh <username>@<ipaddr>`
- For insecure GUI login to a remote Linux system, use vnc
  Not part of this workshop.
- Insecure remote login commands are deprecated. Never use them.
  `telnet, rsh, rlogin`

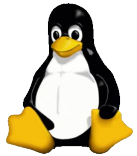
```
$ ssh maruthisi@192.168.136.10
maruthisi@192.168.136.10's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.15.0-36-generic x86_64)
...
maruthisi@godavari:~$
```



uname

- Use `uname -a` command to know basic information about system.
  - `uname -o` : operating system.
  - `uname -r` : kernel version.
  - `uname -m` : hardware architecture.
  - `uname -n` : node/host name.

```
$ uname -a
Linux godavari 4.15.0-29-generic
#31~16.04.1-Ubuntu SMP Wed Jul
18 08:54:04 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux

$ uname -o
GNU/Linux

$ uname -r
4.15.0-42-generic

$ uname -m
x86_64
$ uname -m
aarch64

$ uname -n
godavari
```



Session identification

- To identify a login session, use `tty`
  pts = pseudo terminal session (ssh and Terminal in GUI)
  tty = tele terminal session (CLI)
  ttyS = Serial console session. (CLI)

```
$ tty
/dev/pts/6

$ tty
/dev/tty1

$ tty
/dev/ttyS0
```

Other sessions

- To find out other login sessions, use `who -u` or `w` command

```
$ who -u
maruthisi pts/8        2019-01-09 05:02   .          1265 (172.16.0.100)
owner    pts/9         2019-01-09 05:05   .          1349 (172.16.0.100)

$ w
 05:06:00 up 4 min,  2 users,  load average: 0.04, 0.24, 0.14
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
maruthis pts/8    172.16.0.100     05:02    0.00s  0.13s  0.01s w
owner    pts/9    172.16.0.100     05:05    4.00s  0.08s  0.08s -bash
```

Session history

- To see history of login sessions, use `last` command

```
$ last
owner    pts/9        172.16.0.100     Wed Jan  9 05:05   still logged in
maruthis pts/8        172.16.0.100     Wed Jan  9 05:02   still logged in
reboot   system boot  4.15.0-29-generi Wed Jan  9 05:01   still running
...
owner    tty7         :0               Wed Jan  9 03:41 - crash  (00:54)
reboot   system boot  4.15.0-29-generi Tue Jan  8 06:11 - 09:56  (03:44)

wtmp begins Tue Jan  8 06:11:45 2019
```

Logout

- To logout from a remote Unix/Linux system, use `logout` or `exit` or `ctrl+d`
- During logout all commands launched in the session are killed.

```
$ logout
Connection to localhost closed.
```

Manuals

Manual pages

- Historically documentation of Unix was in set of books. Each book is a Section. Each section has manual pages
- Frequently used sections are
  - command (section 1)
  - system call (section 2)
  - library function (section 3)
  - file format (section 5)
- Use man man to know about sections.
- Use man [<section>] <argument> command to see a manual page.
  - Use arrows to navigate.
  - Use / to search for a word while viewing.
  - Press q to stop viewing.

```
$ man man

$ man ls
$ man passwd

$ man fork

$ man printf

$ man fstab

$ man 5 passwd
```

Apropos

- Apropos is search engine in Unix manual pages.
- It works even without the Internet.
- Use apropos <search_string> command to search a string in all manual pages

```
$ apropos uname
arch (1)              - print machine hardware name (same as uname -m)
oldolduname (2)       - get name and information about current kernel
olduname (2)          - get name and information about current kernel
uname (1)             - print system information
uname (2)             - get name and information about current kernel

$ apropos "get name"
getpeername (2)       - get name of connected peer socket
oldolduname (2)       - get name and information about current kernel
olduname (2)          - get name and information about current kernel
uname (2)             - get name and information about current kernel
```

Bash shell

Shell

- First process after login
- Interprets and launches commands keyed-in at the command prompt
  `<username>@<hostname>:<pwd>{#|$}`
- Most commonly used shell is bash
- Types of shells
  - root vs non-root shell (# vs $ prompt)
  - login vs non-login shell (-bash vs bash)
- Interpreter for shell scripts

```
maruthisi@godavari:~ $ _
```

```
root@godavari:~ # _
```

Commands

- Programs that are keyed-in by user and launched by shell
- Types of commands
  - Internal (builtin) commands
  - External commands

```
$ type cd
cd is a shell builtin

$ type ls
mkdir is /bin/ls
```

Internal Commands

- Internal (built-in) commands are programs implemented by the shell itself

  `echo, fg, bg, cd, umask, ...`

```
$ type cd
cd is a shell builtin

$ which cd
$ which type

$ whereis cd
cd:
```

External Commands

- External commands are programs (mostly c) in /bin or /usr/bin or /usr/sbin directory
- To know the location of a command use which or `whereis` command

```
$ type mkdir
mkdir is /bin/mkdir

$ type ls
mkdir is /bin/ls

$ which ls
/bin/ls

$ which git
/usr/bin/git

$ whereis ls
ls: /bin/ls
/usr/share/man/man1/ls.1.gz
```

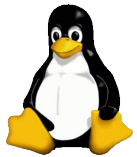
Commands General Syntax

- Commands have a general syntax
  - Command name
  - Options (short or long)
  - Options with arguments
  - Arguments

```
$ command [OPTION]... [OPTION OPTION_ARGUMENT]... [ARGUMENT]...
```

- Typically library functions `getopt()`, `getopt_long()` are used simplify implementation command line options received via `argc`, `argv`.

```
$ ls
$ ls -l
$ ls -l -t -r
$ ls -ltr
$ ls -l abc.txt
$ ls --color abc.txt
$ ls --color=none abc.txt
$ bash --rcfile mybashrc
```



Shell wildcard characters

- Zero or more character match `*`
- One character match `?`
- Character range `[0-9]` `[a-z]` `[A-Z]`
- Character enumeration `[abc]`
- Negation of range/enumeration `[^0-9]` `[^xyz]`
- Escape sequence `\`
- Numeric range `{start..end}`
- Word enumeration `{expr1, expr2}`
- Home directory `~`

```
$ touch file{000..999}.txt

$ ls *.txt

$ ls file*.txt

$ ls file00?.txt

$ ls file0[0-1][0-9].txt

$ ls file??[01].txt

$ ls file{000..999}.txt

$ ls *.{doc,pdf}

$ ls ~
```

Shell environment variables (1/5)

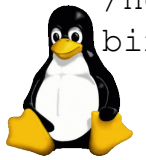
- Every process contains set of variables called environment variables.
- Child processes (commands) inherit a copy of these variables.
- One process cannot modify other processes' environment variables.
- Some of standard env variables (many more are there)

`$HOSTNAME`	–	Hostname
`$HOME`	–	Home directory location
`$SHELL`	–	Default shell
`$PWD`	–	Present working directory
`$PATH`	–	Directory search order for commands (separated by :)
`$LD_LIBRARY_PATH`	–	Directory search order for shared libraries(separated by :)
`$TERM`	–	Terminal type
`$DISPLAY`	–	GUI display name, port (used by vnc, X11 forwarding)
`$?`	–	Exit status of previous command (0: success, ≠0 : failure)

Shell environment variables (2/5)

- To display value of variable use `echo`

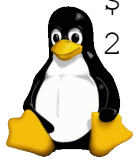
```
$ echo $HOSTNAME
godavari

$ echo $HOME
/home/maruthisi

$ echo $SHELL
/bin/bash

$ cd /
$ echo $PWD
/

$ echo $PATH
/home/maruthisi/bin:/home/maruthisi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/s
bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```



Shell environment variables (3/5)

- Exit status = 0 if previous command succeeded, and ≠ 0 else.

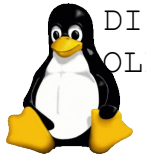
```
$ echo $TERM
xterm-256color

$ echo $DISPLAY
:0

$ cd ~
$ ls .bashrc
.bashrc
$ echo $?
0

$ ls abcd.txt
ls: cannot access 'abcd': No such file or directory
$ echo $?
2
```



Shell environment variables (4/5)

- To display all variables use `env` or `export`

```
$ env
...
TERM=xterm-256color
...
SHELL=/bin/bash
...
USER=maruthisi
...
PATH=/home/maruthisi/bin:/home/maruthisi/.local/bin:/usr/local/sbin:/usr/local/bin:/
usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
...
PWD=/home/maruthisi
...
HOME=/home/maruthisi
...
DISPLAY=:0
OLDPWD=/
```



Shell environment variables (5/5)

- To assign a value to a variable
  - In current shell and future child processes, use `export`
  - In current shell only, assign without export.

```
$ export PATH=/usr/local/cuda-10.0/bin:/opt/ros/kinetic/bin:$PATH

$ echo $PATH
/usr/local/cuda-10.0/bin:/opt/ros/kinetic/bin:/home/maruthisi/bin:/home/maruthisi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

$ PATH=/usr/local/cuda-9.0/bin:/opt/ros/kinetic/bin:$PATH
```

RC scripts

- Enlist variable assignments in .bashrc to assign variables at login time.

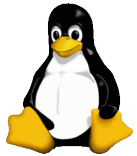
```
$ vi .bashrc
...
export PATH=/usr/local/cuda-9.0/bin:/opt/ros/kinetic/bin:$PATH
```

Users and Groups

Users

- User is an account for
  - A super user (root)
  - A person
  - A service

- An integer number (UID) is assigned to each user account (/etc/passwd file)
- Each user
  - Must belong to one primary group (preferably solo group)
  - May belong to many supplementary groups



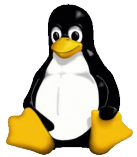
Groups

Group is

- A logical collection of user accounts
- An integer number (GID) is assigned to each group (`/etc/group` file)

Eg:

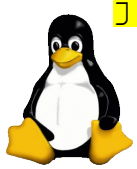
- All Ph.D CSE students from 2018 January
  `cs18resch01`
- All sudo users
- A solo group
  `maruthisi`



Users

- Username, uid, gid, home directory, default login shell are stored in `/etc/passwd` file
- To know the uid, gid use `id` command

```
$ id maruthisi
uid=20201(maruthisi) gid=20201(maruthisi) groups=2051(cs18resch01),27(sudo)

$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...
amits:x:10001:10001::/home/amits:/bin/nologin
sohailm:x:11001:11001::/home/sohailm:/bin/sh
davidk:x:20001:20001::/home/davidk:/bin/bash
maruthisi:x:20201:20201::/home/maruthisi:/bin/bash
jyothin:x:20202:20202::/home/jyothin:/bin/sh
```



Passwords

- Passwords are stored in encrypted form in `/etc/shadow` file
- Change password using `passwd` command

```
$ passwd

$ cat /etc/shadow
...
amits:$6$IRU45oem$RcHnDVg459/1GXNwRJmz7wqsyyfzb95k.6WEMV2Du04yf/lz0:17564:0:99999:7:::
sohailm:$6$FKFYEysx$xvwpzSRJPq1hLtlH577YQZJKLHX9.RCp01KKry6A2guclV0:17564:0:99999:7:::
jyothin:$6$NU9mvrF4$bgHrQxIV24lMlHynK2Mxefbo1UC9gMpcKNzCVaK/8mA9IS.:17564:0:99999:7:::
maruthisi:$6$WT483SxE$bzsX290lzla/nb8NZ6X2c3u0OFdhpcv.BynwFqP5.UPr1:17564:0:99999:7:::
```

Groups

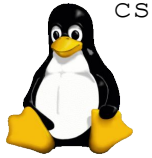
- Group name, gid are stored in `/etc/group` file
- To know the uid, gid use `id` command

```
$ id maruthisi
uid=20201(maruthisi) gid=20201(maruthisi) groups=20201(maruthisi)

$ cat /etc/group
root:x:0:
daemon:x:1:
...
maruthisi:x:20201:
...
```

Supplementary Groups

- Supplementary group name, gid, members are stored in `/etc/group` file
- To know the uid, gid, gids use `id` command

```
$ id maruthisi
uid=20201(maruthisi) gid=20201(maruthisi) groups=20201(maruthisi),2051(cs18resch01),27(sudo)

$ cat /etc/group
root:x:0:
daemon:x:1:
sudo:x:27:owner,maruthisi
...
maruthisi:x:20201:
...
cs18resch01:x:2051:maruthisi,jyothin
```

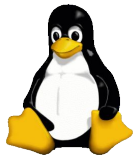


Home Directories

- A directory in /home created one for each user
- Starting directory after login.
- Special character is ~

- Permissions play a key role for securing files from other users, other groups, others in the world.
- UMASK in `/etc/login.defs` should be set to 077.

- Per user limits on storage space usage can be enforced using quotas.



Home Directories

```
$ ls -l /home/
total 36
drwx------  2 amits     amits       4096 Feb  2 10:50 amits
drwx------  2 davidk    davidk      4096 Feb  2 10:51 davidk
drwx------  2 jyothin   jyothin     4096 Feb  2 10:59 jyothin
drwx------  2 maruthisi maruthisi   4096 Feb  2 10:52 maruthisi
drwx------ 17 owner     owner       4096 Feb  2 11:19 owner
...
```

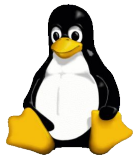
Processes, Jobs

Processes

- A running program
- An Integer number (PID) is used for uniquely identifying a process.

- Processes which run in background are called daemons
- Processes which run in foreground are called interactive

- Each process is launched from an user account.
- Daemons are launched from service accounts or administrator account.



Process hierarchy

- All processes in a system form a hierarchy.
- The first process in the system is
  - init (in older systems, prior to 2016)
  - systemd (in current distros)
- Each process (except init/systemd) also has unique parent (PPID)

```
# pstree
systemd-+-NetworkManager-+-dhclient
        |                |-{NetworkManager}
        |                |-{gdbus}
        |                `-{gmain}
        |-acpid
        |-avahi-daemon---avahi-daemon
        |-crond
        |-login---bash---pstree
        |-smartd
        |-sshd
        |-systemd---(sd-pam)
        |-systemd-journal
        |-systemd-logind
        |-systemd-udevd
        `-wpa_supplicant
```

Process hierarchy (ubuntu)

- Historically, `init` is the first process with pid 1
- All other processes are descendents of the `init`
- Processes with PPID 1 are daemons.
- Processes with PPID 0 dont have parent.
- Command name displayed in `[]` are kernel threads, else userspace processes.

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 05:46 ?        00:00:01 /sbin/init splash
root         2     0  0 05:46 ?        00:00:00 [kthreadd]
root        27     2  0 05:46 ?        00:00:01 [kswapd0]
...
maruthi+  1313     1  0 05:47 ?        00:00:00 /lib/systemd/systemd --user
...
maruthi+  2834  1367  0 05:51 ?        00:00:07 /usr/lib/gnome-terminal/gnome-terminal
maruthi+  2841  2834  0 05:51 pts/4    00:00:00 bash
maruthi+  4971  2841  0 06:37 pts/4    00:00:00 ps -ef
...
```

Process hierarchy (fedora)

- In recent distros, `systemd` is the first process with pid 1
- All other processes are descendents of the `systemd`
- Processes with PPID 1 are daemons.
- Processes with PPID 0 dont have parent.
- Command name displayed in `[]` are kernel threads, else userspace processes.

```
$ ps -ef
UID          PID    PPID    C STIME TTY          TIME CMD
root           1       0    0 03:34 ?        00:00:02 /usr/lib/systemd/systemd --switched-root ...
root           2       0    0 03:34 ?        00:00:00 [kthreadd]
root          53       2    0 03:34 ?        00:00:06 [kswapd0]
...
maruthi+    2730          1      0 03:39 tty2   00:00:09 /usr/libexec/gnome-terminal-server
maruthi+    2734    2730   0 03:39 pts/0  00:00:00 bash
maruthi+    4431    2734   0 06:58 pts/0  00:00:00 ps -ef
...
```

Listing processes

- List processes in current login session, use `ps -f` or `ps -fH` for hierarchy.
- List all processes in the system, use `ps -ef` or `ps -efH` for hierarchy.
- `pstree` could also be used to see processes (without pids) as a hierarchy.

```
$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
maruthi+    5075  5057  0 02:18 pts/17   00:00:00 bash
maruthi+    5848  5075  0 02:55 pts/17   00:00:00 ps -f

$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 05:46 ?        00:00:01 /sbin/init splash
root           2     0  0 05:46 ?        00:00:00 [kthreadd]
root          27     2  0 05:46 ?        00:00:01 [kswapd0]
...
owner       1313        1      0 05:47 ?        00:00:00 /lib/systemd/systemd --user
...
owner       2834  1367  0 05:51 ?        00:00:07 /usr/lib/gnome-terminal/gnome-terminal
owner       2841  2834  0 05:51 pts/4    00:00:00 bash
owner       4971  2841  0 06:37 pts/4    00:00:00 ps -ef
...
```

Listing threads

- List threads in current login session, use `ps -Lf`
- List all threads in the system, use `ps -Lef`
- List all threads (without pids) in the system in hierarchy, use `pstree` notice `n*[]`

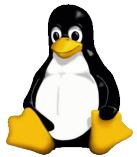
```
$ ps -Lf
UID          PID  PPID   LWP  C NLWP STIME TTY          TIME     CMD
maruthi+  5075  5057  5075  0    1 02:18 pts/17   00:00:00 bash
maruthi+  5808  5075  5808  0    1 02:48 pts/17   00:00:00 ps -Lf
```

```
$ ps -Lef
UID          PID  PPID   LWP  C NLWP STIME TTY          TIME     CMD
...
maruthi+  4627  3631  4627  4   62 02:18 ?        00:01:20 /usr/lib/firefox/firefox
maruthi+  4627  3631  4649  0   62 02:18 ?        00:00:00 /usr/lib/firefox/firefox
...
```

Finding process

- To find process id from program name use `psgrep` or `pidof`

```
$ pgrep bash
8279
8354

$ pidof bash
8354 8279
```



Sending signals to processes

- Signals are asynchronous messages to processes.
- Shell sends signals to processes. Eg.
  - `SIGHUP` – During logout to all sub processes.
- Kernel sends signals to processes. Eg.
  - `SIGSEGV` – During illegal memory access.
  - `SIGFPE` – During illegal operands (like divide by zero, operation on NAN)
  - `SIGBUS` – Memory access alignment error.
  - `SIGCHLD` – Child process stopped/terminated.
- User could also send signals (see man 7 signal) to process(es)
  - With pid(s) using `kill -<signaltype>`
  - With program name using `pkill -<signaltype>`

Terminating/Killing processes

- To terminate process(es) gracefully (default signal)
  - Use `SIGTERM : 15`
- To terminate process(es) abruptly/forcibly
  - Use `SIGKILL : 9`
- To terminate/kill an interactive process use `ctrl+ c`

```
$ pidof firefox
8279 8354

$ kill 8279
$ kill -SIGTERM 8279
$ kill -15 8279

$ kill -SIGKILL 8354
$ kill -9 8354

$ pkill firefox
$ pidof firefox
```

Stopping/Continuing process

- Processes could also be stopped (paused. Not exited) and continued (resumed) using signals
  - `SIGSTOP` : Stop process
  - `SIGCONT` : Resume stopped process.

```
$ pidof firefox
8279 8354

$ kill -SIGSTOP 8354

$ kill -SIGCONT 8354
```

Other signals

- User could send other signals. Eg.
  - `SIGQUIT` – To dump core and terminate process(es).
  - `SIGUSR1, SIGUSR2` – To user-defined behavior.

```
$ pkill -SIGQUIT sample.out

$ ls
coredump
```

Jobs

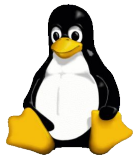
- Interactive shell associates a job with each pipeline.
- Each job is assigned a job id, which is unique within the shell.
- Use `&` at end of command line to start a job in background
- Without `&`, job is started in foreground

```
$ sleep 10 &

$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
maruthi+  5075  5057  0 02:18 pts/17   00:00:00 bash
maruthi+  6057  5075  0 03:27 pts/17   00:00:00 sleep 10
maruthi+  6058  5075  0 03:27 pts/17   00:00:00 ps -f

$ jobs
[1]+  Running                 sleep 10 &

$ sleep 10
```

Switching jobs between foreground/background

- In an interactive shell,
  - `Ctrl + C` could be used to terminate a job.
  - `Ctrl + Z` could be used to stop a job and push it to background.
  - `fg` command could be used resume a job and bring it foreground.
  - `bg` command could be used to resume a job and run it in background.

```
$ cat
^Z
[1]+  Stopped                 cat
$ cat
^Z
[2]+  Stopped                 cat

$ jobs
[1]-  Stopped                 cat
[2]+  Stopped                 cat

$ fg 1
cat
```



File System Basics

File System

- A subsystem in OS kernel
- Logical organization of disk sectors/blocks into files and directories.
- Does accounting of free/used space
- Provides quotas at user/group level
- Provides security using ownership, permissions, access controls (ACL).
- Addresses limitations of disk drives.
  - Logical blocking
  - Caching
- Different File systems types
  - xfs
  - ext4
  - iso9660

File System

- List all file systems using `df -h`
- List file systems along with types using `df -hT`

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/sda3        15G   13G  2.5G  84% /
tmpfs           1.9G   40K  1.9G   1% /tmp
/dev/sdb10      725G  315G  411G  44% /mnt/MySpare
/dev/sdb9       1.9T  763G  1.1T  41% /mnt/MyDrive
/dev/sda5       200G  8.0G  192G   4% /home
...

$ df -hT
Filesystem      Type       Size  Used Avail Use% Mounted on
...
/dev/sda3       ext4        15G   13G  2.5G  84% /
tmpfs           tmpfs      1.9G   40K  1.9G   1% /tmp
/dev/sdb10      xfs        725G  315G  411G  44% /mnt/MySpare
/dev/sdb9       xfs        1.9T  763G  1.1T  41% /mnt/MyDrive
/dev/sda5       xfs        200G  8.0G  192G   4% /home
...
```

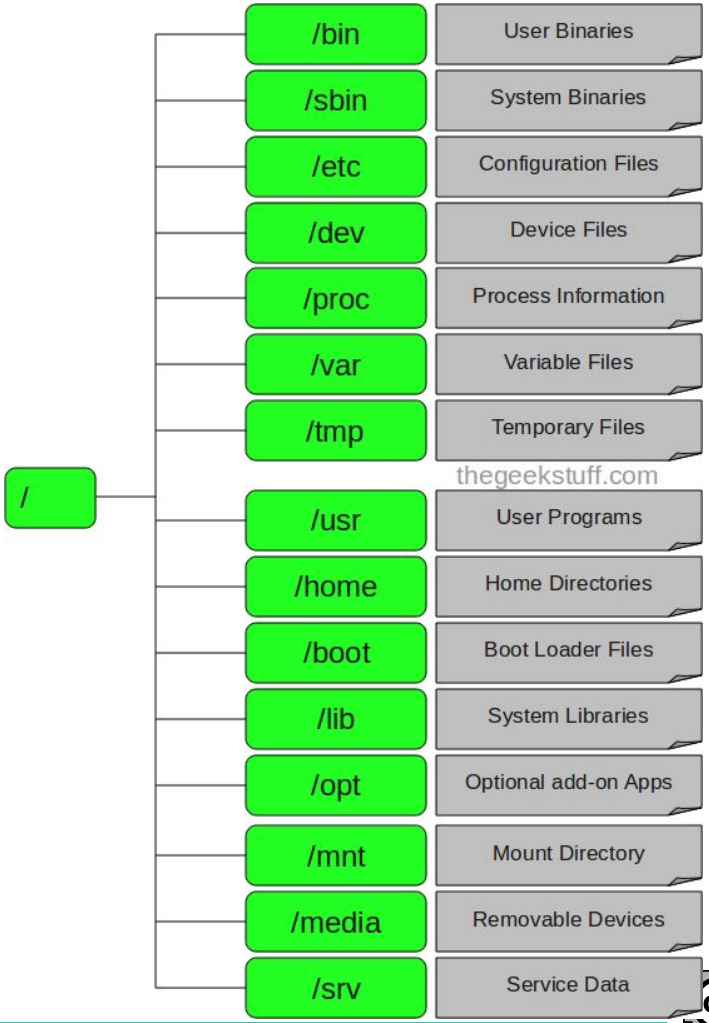
File Systems Hierarchy

- Multiple file-systems organized in a tree structure
- Top most is / (root directory).

NOTE: root is overloaded term. If not explicitly qualified, it could refer to

- root directory
- root user
- root user's home directory

based on context.



File Types

- In Unix/Linux, everything in file-system is a file.

- There are many types of files:

Regular File

```
-rw-r--r-- 1 root root 35913142 Feb  3 04:34 initrd.img-4.4.0-31-generic
```

Directory

```
drwxr-xr-x 5 root root     4096 Nov 14  2016 grub
```

Block (buffered) device special file

```
brw-rw---- 1 root disk     8, 0 Feb  2 10:40 /dev/sda
```

Character (unbuffered) device special file

```
crw--w---- 1 owner tty   136, 0 Feb  3 04:36 /dev/pts/0
```

Symbolic Link (aka soft link)

```
lrwxrwxrwx 1 root root 19 Nov 14  2016 /etc/mtab -> ../proc/self/mounts
```

Socket special file

```
srw-rw-rw-. 1 root root 0 Feb  3 03:34 /run/cups/cups.sock
```

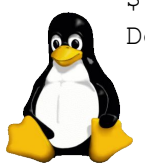
Named Pipe special file

```
prw------- 1 root root 0 Feb  2 10:41 /run/systemd/inhibit/6.ref
```

File Types

- Know file type use `ls -l` command, notice the first letter in output.
- Regular files can be further differentiated based on content.
- Know file type based on its content using `file` command. Works based on magic number stored in `/usr/share/misc/magic.mgc` and `/etc/magic`

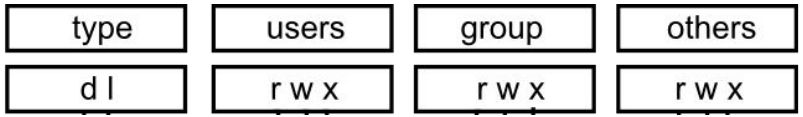
```
$ ls -l
-rw-r--r-- 1 root root 35913142 Feb  3 04:34 initrd.img-4.4.0-31-generic

$ file Documents
Documents: directory

$ file .bashrc
.bashrc: ASCII text
$ file sizesof.c
sizesof.c: C source, ASCII text
$ file Documents/ds-3808.pdf
Documents/ds-3808.pdf: PDF document, version 1.4
```



Discretionary Access Control

User types

- Three different categories of users
  - User (Self)
  - Group
  - Others (World)
- root user can read/write/delete everyone's files.




```
$ ls -l /share/
...
drwxr-xr-x  9 maruthisi    maruthisi        4096 Feb 19 17:29 public
drwxr-x---  9 maruthisi    cs18resch01           4096 Feb 19 17:29 cs18resch01
...
```

File Ownership

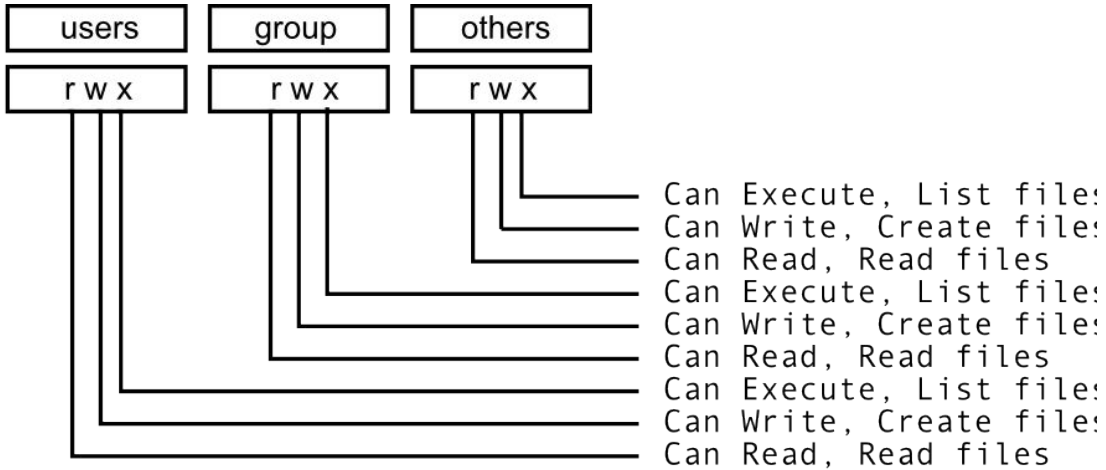
- Each file has (user and group)
  - User ownership
  - Group ownership

- Use `chown` to change user ownership of any file
- Use `chgrp` to change group ownership of any file

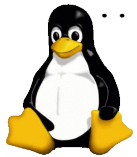
```
$ ls -l /share/
...
drwxr-xr-x  9 maruthisi    maruthisi       4096 Feb 19 17:29 public
drwxr-x---  9 maruthisi    cs18resch01          4096 Feb 19 17:29 cs18resch01
...
```

Permissions

- For directories
  - read (r), write (w), explore (x)
- For other files
  - read (r), write (w), execute (x)




```
$ ls -l /share/
...
drwxr-xr-x   9 maruthisi        maruthisi       4096 Feb 19 17:29 public
dwxr-x---    9 maruthisi        cs18resch01     4096 Feb 19 17:29 cs18resch01
...
```



Default permissions

- To see user's file-creation mask use `umask` command.
- mask is displayed in
  - octal format by default
    0000
  - symbolic format with -S option.
    u=rwx,g=rwx,o=rwx

```
$ umask
0002
$ umask -S
u=rwx,g=rwx,o=rx

$ touch abc.txt
$ ls -l
-rw-rw-r-- 1 maruthisi maruthisi 0 Jan 19 04:23 abc.txt
```

Changing default permissions

- To set user's file-creation mask use `umask` command.
- mask could be in octal format or symbolic format.
- To set user's file-creation mask permanently set it in `.bashrc`

```
$ umask -S
u=rwx,g=rwx,o=rx

$ touch abc.txt
$ ls -l
-rw-rw-r-- 1 maruthisi maruthisi 0 Jan 19 04:23 abc.txt

$ umask u=rwx,g=,o=
$ umask 0077
$ umask -S
u=rwx,g=,o=

$ touch xyz.txt
$ ls -l
-rw-rw-r-- 1 maruthisi maruthisi 0 Jan 19 04:23 abc.txt
-rw------- 1 maruthisi maruthisi 0 Jan 19 04:24 xyz.txt
```

Change permissions

- To explicitly change permissions use `chmod`
- mask could be in
  - octal format or symbolic format.
  - absolute (=) or relative (+ or -)

```
$ ls -l
-rw-rw-r-- 1 maruthisi maruthisi 0 Jan 19 04:23 abc.txt
-rw-rw-rw- 1 maruthisi maruthisi 0 Jan 19 04:34 wuv.txt
-rw------- 1 maruthisi maruthisi 0 Jan 19 04:24 xyz.txt

$ chmod u=rw,g=,o= wuv.txt xyz.txt
```

or

```
$ chmod o-rw wuv.txt
$ chmod g+rw xyz.txt

$ ls -l wuv.txt xyz.txt
-rw-rw---- 1 maruthisi maruthisi 0 Jan 19 04:34 wuv.txt
-rw-rw---- 1 maruthisi maruthisi 0 Jan 19 04:24 xyz.txt
```

File operations

Listing files

- To list files in present working directory, use `ls` with options
  - -l : long listing
  - -h : sizes in human readable format (KB, MB, GB…)
  - -t : sort by time (oldest last)
  - -r : reverse the order
  - -R : recursive listing of directory tree
- `total` is the size of metadata and data of directory being listed.

```
$ ls -hltr /boot
total 351M
drwx------ 3 root root 4.0K Jan  1  1970 efi
-rw-r--r-- 1 root root 181K Jan 28  2016 memtest86+_multiboot.bin
-rw-r--r-- 1 root root 181K Jan 28  2016 memtest86+.elf
-rw-r--r-- 1 root root 179K Jan 28  2016 memtest86+.bin
...
-rw------- 1 root root 3.9M Nov 19 21:02 System.map-4.15.0-42-generic
drwxr-xr-x 5 root root 4.0K Dec 21 07:34 grub
-rw-r--r-- 1 root root  59M Dec 21 08:58 initrd.img-4.15.0-42-generic
```

Paths

- Path is a sequence of file names separated by /
  - Path component: Each filename is called path component.
    Eg. home, usr
  - Absolute path: Sequence of path components starting with the root directory.
    Eg. /home/maruthisi , /bin/ls, /usr/local/cuda-10.0/bin
  - Relative path : Sequence of path components starting with present working directory.
    Eg. Documents/cv.docx, Pictures/passport.jpeg

```
$ ls -l /home/maruthisi/Documents/TA.pdf
-rw-rw-r-- 1 maruthisi maruthisi 83432 Jul 30 08:49 /home/maruthisi/Documents/TA.pdf

$ ls -l Documents/TA.pdf
-rw-rw-r-- 1 maruthisi maruthisi 83432 Jul 30 08:49 Documents/TA.pdf
```

Splitting path

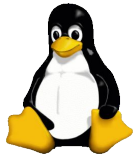
- To find base directory of a given path, use dirname
- To find leaf level path component, use basename

```
$ ls -l /home/maruthisi/Documents/TA.pdf
-rw-rw-r-- 1 maruthisi maruthisi 83432 Jul 30 08:49 /home/maruthisi/Documents/TA.pdf

$ basename /home/maruthisi/Documents/TA.pdf
TA.pdf

$ dirname /home/maruthisi/Documents/TA.pdf
/home/maruthisi/Documents
```

Creating directories

- To create directories use `mkdir` command
- Use `-p` option to create a deeply nested directory and all its path components.

```
$ mkdir demo

$ mkdir -p demo/dir1/dir11 demo/dir1/dir12
```



Deleting directories

- To delete directories use `rmdir` command
- Only empty directories can be deleted with rmdir
- Use `-p` option to delete a ancestors of a nested directory. Only empty ancestors will be deleted.

```
$ rmdir demo
rmdir: failed to remove 'demo': Directory not empty

$ rmdir demo/dir1/dir11 demo/dir1/dir12
$ rmdir -p demo/dir1
$ rmdir demo
rmdir: failed to remove 'demo': No such file or directory
```

Home, Current, Parent directory

- Every user has a home directory
  - Denoted using ~

- By default two hidden directory entries (dirent) exist in every directory
  - Current directory .
  - Parent directory ..

```
$ ls -la ~
drwx------. 62 maruthisi maruthisi    8192 Jan 19 05:15 .
drwxr-xr-x.  6 root      root           83 Dec 10 04:38 ..
...

$ ls -la
drwxrwxrwx   3 maruthisi maruthisi   51 Jan 19 05:46 .
drwx------. 62 maruthisi maruthisi 8192 Jan 19 05:15 ..
...
```

Knowing present working directory

- At any time a shell works in one directory, viz “present working directory”.
- To know present working directory, use `pwd`

```
$ pwd
/home/maruthisi/demo
```

Changing present working directory

- To change present working directory, use `cd`
  - To change to home directory, use `cd` or `cd ~`
  - To change to any other directory, use `cd <directory>`
  - To go back to previous working directory, use `cd -`
  - To go to parent directory, use `cd ..`

```
$ pwd
/home/maruthisi

$ cd demo
$ cd dir1
$ pwd
/home/maruthisi/demo/dir1

$ cd -
/home/maruthisi/demo
```

```
$ pwd
/home/maruthisi/demo

$ cd ~
$ pwd
/home/maruthisi

$ cd ..
$ pwd
/home
```

Hidden files

- Any filename that starts with . is a hidden file/directory
- To see hidden files use `-a` option of `ls`

```
$ ls -la ~
...
-rw-rw-r--   1 maruthisi maruthisi 1081 Dec 10 20:37 .bashrc

$ ls -la
total 16
drwxrwxrwx   3 maruthisi maruthisi   51 Jan 19 05:46 .
drwx------. 62 maruthisi maruthisi 8192 Jan 19 05:15 ..
...
-rw-rw-rw-   1 maruthisi maruthisi    0 Jan 19 05:46 .mno.txt
```

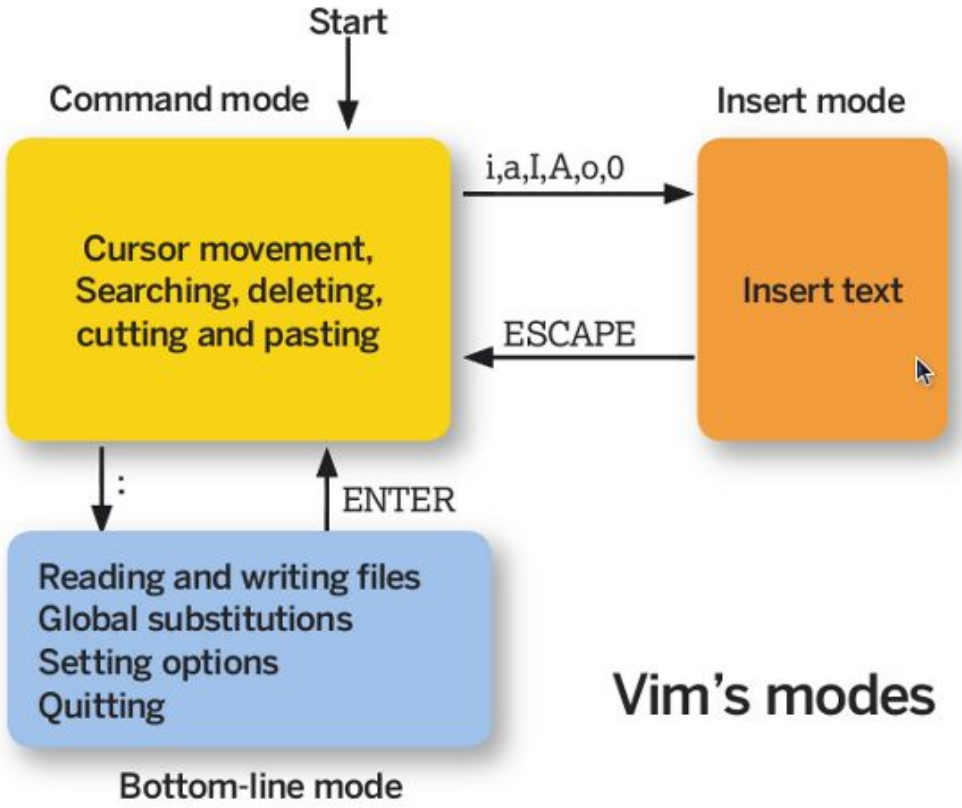
Creating regular files (1/2)

- To create regular files of zero length, use `touch` command.
- To create regular files of desired length, use `truncate -s` command.
  Files created with truncate will be sparse (no storage blocks allocated).
  Use KiB, MiB, .. for 1024 based units. KB, MB, … for 1000 based units.

```
$ touch abc.txt def.txt
$ ls -l
-rw-rw-rw- 1 maruthisi maruthisi 0 Jan 19 08:49 abc.txt
-rw-rw-rw- 1 maruthisi maruthisi 0 Jan 19 08:49 def.txt

$ truncate -s 20TB xyz.txt
$ truncate -s 20TiB uvw.txt
$ ls -lh
-rw-rw-rw- 1 maruthisi maruthisi   0 Jan 19 08:49 abc.txt
-rw-rw-rw- 1 maruthisi maruthisi   0 Jan 19 08:49 def.txt
-rw-rw-rw- 1 maruthisi maruthisi 20T Jan 19 08:51 uvw.txt
-rw-rw-rw- 1 maruthisi maruthisi 19T Jan 19 08:49 xyz.txt
```

Creating regular files (2/2)

- To create a regular file with desired content, use `vim` editor.
- Vim is the widely available and powerful editor.
- It operates in three modes. Command mode, Insert mode, Bottom-line mode.
- In Bottom-line mode
  - `w` to write/save
  - `q!` to quit without saving
  - `wq` to write and quit

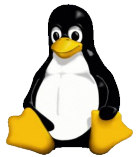



```
$ vi abc.txt
hello world
```

Displaying regular file content

- To display content of a regular file having ASCII text, use `cat`
- Using `cat` to display a regular file having binary data may freeze login session.
- Vim could also be used to view content of a regular file having ASCII text.

```
$ cat abc.txt
hello world
$
```



Copying files

- To copy files, use `cp`
  - -r : recursively
  - -p : preserve permissions and ownership
  - -i : interactive (queries yes/no)
  - -f : force overwrite (no querying)
- Equivalent to copy+paste in GUI.

```
$ cp abc.txt ABC.txt

$ ls -l
-rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:10 abc.txt
-rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:29 ABC.txt
```

Moving files

- To rename/move files, use `mv`
- Equivalent to cut+paste in GUI.
- Move within file-system does not involve data movement (just a metadata operation).
- Move across file-systems involves data movement operation. Time consuming for large files.

```
$ mv abc.txt ABC.txt

$ ls -l
-rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:10 abc.txt
-rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:29 ABC.txt
```

Deleting files

- To delete files, use `rm`
  - -i : interactive (queries yes/no)
  - -f : force overwrite (no querying)
- Equivalent to shift+delete in GUI.

CAUTION:

- There is no recycle-bin in Unix/Linux. Files deleted from command line usng rm are permanently deleted.
- lost+found is not recyle bin.

```
$ rm ABC.txt
$ ls -l ABC.txt
ls: cannot access 'ABC.txt': No such file or directory
```

Deleting directories and files recursively

- To delete files recursively, use `rm`
  -r : recursively
- Equivalent to shift+delete in GUI.

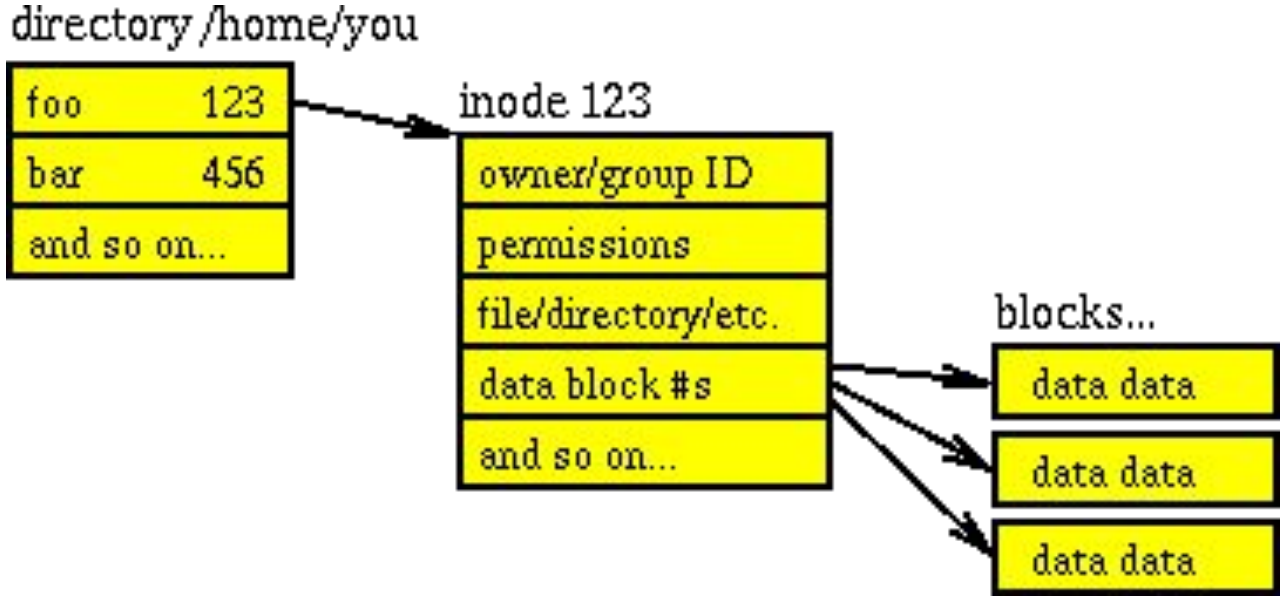
CAUTION:

- `rm -rf` is a double-edged sword. Extra caution.

```
$ rm -rf demo
$ ls -l demo
ls: cannot access 'demo': No such file or directory
```

Index node (i-node) and Directory Entry (dirent)

- Every file has few ondisk data structures for metadata.
- Index node (inode) with a unique number (ino). Filename is not part of inode.
- Directory entries (dirent) which stores the file name, and inode number.



Knowing file metadata (1/2)

- To know file's metadata use `stat` command.

```
$ ls -li
  6894908 drwxrwxrwx 3 maruthisi maruthisi 16 Jan 19 05:20 dir1
421776638 -rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 05:15 file1.txt

$ stat dir1
  File: 'dir1'
  Size: 16              Blocks: 0          IO Block: 4096   directory
Device: 807h/2055d      Inode: 6894908     Links: 2
Access: (0777/drwxrwxrwx)  Uid: ( 1001/maruthisi)   Gid: ( 1001/maruthisi)
Access: 2019-01-19 05:19:54.122106408 +0530
Modify: 2019-01-19 05:20:02.774092126 +0530
Change: 2019-01-19 05:20:02.774092126 +0530
 Birth: -

$ stat file1.txt
  File: 'file1.txt'
  Size: 12              Blocks: 8          IO Block: 4096   regular file
Device: 807h/2055d      Inode: 421776642   Links: 1
...
```

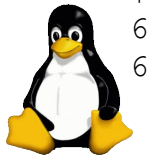
Knowing file metadata (2/2)

- Attributes from dirent:
  - File: name
- Attributes in inode:
  - Size: apparent file size
  - Blocks: allocated file size (in 512 byte blocks)
  - IO Block: Unit of I/O size by underlying block device driver.
  - Device: id of device on which this inode exists.
  - Inode: inode number
  - Links: number of dirents pointing to this inode.
  - Access: permissions
  - Uid: user owner's id
  - Gid: group owner's id
  - Access: the last time the file was read
  - Modify: the last time the file was modified (content has been modified)
  - Change: the last time meta data of the file was changed (e.g. permissions)
  -

Symbolic link

- To create a symbolic link (symlink), use `ln -s`
  -f : force recreation
- Symlinks can be created within and across file-systems.
- Separate inode and dirent is created for symlink.
- A symlink can become dangling link if target is deleted.
- `ls -l` shows “l” in the file type column.
  -L : to traverse symlink. Useful for detecting broken links.

```
$ ln -s abc.txt pqr.txt
$ ls -li
6894904 -rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:48 abc.txt
6894915 lrwxrwxrwx 1 maruthisi maruthisi  7 Jan 19 09:49 pqr.txt -> abc.txt

$ ls -liL
6894918 -rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:56 abc.txt
6894918 -rw-rw-rw- 1 maruthisi maruthisi 12 Jan 19 09:56 pqr.txt
```



Hard link

- To create a link (hardlink), use `ln`
  -f : force recreation
- Hardinks must be created within file-system.
- A new dirent is created for each hardlink. Inode is shared across all hardlinks of the inode.
- A hardlink cannot become dangling link if target is deleted. Link count reduces.
- `ls -l` shows “-” in the file type column. Link count could be used to differentiate.

```
$ ln abc.txt stu.txt
$ ls -li
6894913 -rw-rw-rw- 2 maruthisi maruthisi 12 Jan 19 09:59 abc.txt
6894913 -rw-rw-rw- 2 maruthisi maruthisi 12 Jan 19 09:59 stu.txt
```

Searching files and directories

Finding files

- Searching files could be done using find
  - file extension
  - file type
  - file name

```
$ find . -name "*.mp3"
./Music/Ringtones/bhajare.mp3

$ find . -type d -name "Pictures"
./Pictures
```

Finding sizes

- File size could be found from `ls -lh` or more precisely `ls -l`
- Directory size could be found from `ls -ldh` more precisely `ls -ldh`
- Directory and its contents size could be found using `du -sh` or more precisely `du -s`

```
$ ls -lh ./Music/Ringtones/bhajare.mp3
-rwxr-----.  1 maruthisi maruthisi 610K Aug 11  2016 ./Music/Ringtones/bhajare.mp3

$ ls -ldh Downloads/
drwxr-xr-x. 17 maruthisi maruthisi   4K Feb  3 14:05 Downloads/

$ du -sh Downloads
1.1G    Downloads
```

Finding inside files

- Searching inside files could be done using `grep`
  - Inside a file
  - Inside all files in recursively in a directory

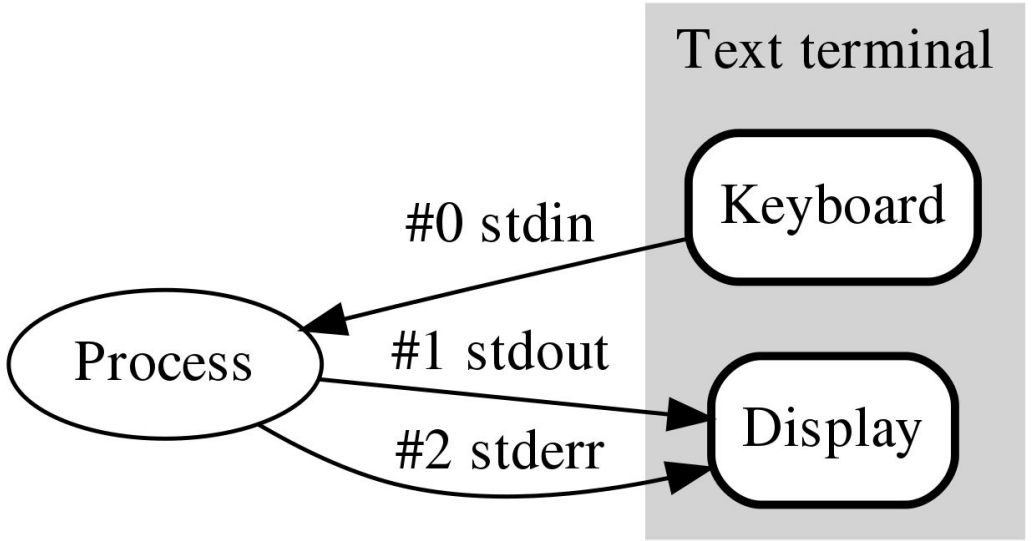
```
$ grep -e "compare" Prog1.cu
Prog1.cu:int compare_print_result(int *h_a, int *h_b, int *h_c, int *hd_c, int size);
Prog1.cu:    if (compare_print_result(h_a, h_b, h_c, hd_c, size) != 0) {
Prog1.cu:int compare_print_result(int *h_a, int *h_b, int *h_c, int *hd_c, int size)
```

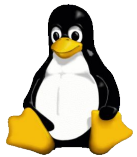
```
$ grep -R -e "compare" .
./Prog1.cu:int compare_print_result(int *h_a, int *h_b, int *h_c, int *hd_c, int size);
./Prog1.cu:    if (compare_print_result(h_a, h_b, h_c, hd_c, size) != 0) {
./Prog1.cu:int compare_print_result(int *h_a, int *h_b, int *h_c, int *hd_c, int size)
...
```

Redirection, Pipes and Filters

Process input outputs

- Three files are opened by default for every process
  - Standard Input (file descriptor #0)
  - Standard Output (file descriptor #1)
  - Standard error (file descriptor #2)
- Command line parameters
- Environment variables
- Shared memory
- Message queues





Input redirection

- The process of feeding standard input from a file instead of keyboard.
- Use `command [options] [args] < filename`

```
$ cat < /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...
$
```

Output redirection

- The process of sending standard output to a file instead of monitor.
- Use `command [options] [args] > filename`
  or
- Use `command [options] [args] 1> filename`

```
$ ls -l > ls.txt
$
```

Error redirection

- The process of sending standard errors to a file instead of monitor.
- Use `command [options] [args] 2> filename`

```
$ ls -l abcd.txt
ls: cannot access 'abcd.txt': No such file or directory

$ ls -l abcd.txt 2> efg.txt

$ cat efg.txt
ls: cannot access 'abcd.txt': No such file or directory
```

Infinite source and sink files

- Infinite sources
  - /dev/zero
  - /dev/urandom
  - /dev/random
  - yes
- Infinite sinks
  - /dev/null

```
$ yes
y
...
$ cat /dev/zero
$ cat /dev/random              # caution may make current shell use-less
...
$ cat /dev/urandom             # caution may make current shell use-less
...
$ cat /dev/urandom > /dev/null
```

Pipes

- The process of sending standard output of one process to standard input of another process.
- Use `command1 | command2`

```
$ ls -l | cat
```

Filters

- Programs that modify input from stdin in some way and produce output on stdout.
- Use `command [options] [args] | filter1 [options] [args]`
- Commonly used filters
  `head, tail, less, more, wc, grep, sort, uniq, awk, sed`

```
$ cat /etc/passwd | grep maruthisi
maruthisi:x:1001:1001::/home/maruthisi:/bin/bash
$

$ ls -l | sort -k 5 -n
<output of ls sorted by size>

$ cat /etc/passwd | awk -F: '{print $7}'  | sort | uniq | wc -l
6
```

Command substitution

- The process of sending standard output of one process to command line arguments of another process.
- To run a command and substitute its output as command line input, use backticks `` `command` ``

```
$ kill `pidof firefox`
```

Arguments' injection

- The process of sending standard output of one process to command line arguments of another process using a filter.
- To run a command and substitute its output as command line input, use `xargs` filter

```
$ sudo find /lib -name "*.ko" | xargs du -c | tail -1
41512    total
```

References

References

- Linux manual pages

Q & A