File and Directory command: movement

Copy

cp [flags] <file> <destination>

- Copies the file <file> to a location <destination>.
- Use -r flag to copy an entire directory.

Move

mv [flags] <source> <destination>

mv [flags] <oldname> <newname> (rename)

- Moves a file or directory from <source> to <destination>.
- Recurses for directories automatically (unlike cp).

File and Directory command: movement

Remove" File
rm [flags] <file>
rm -i <filename> (interactive deleting - good idea!)
alias rm="rm -i" is called aliasing

- Be cautious!
- Use wildcards to delete multiple files.

Remove Directory
rmdir [flags] <directory>
(empty directory)
rm -r <directory>
(directories + subdirectories)
Be extremely cautious!!

File Systems - Pathnames

- In Unix case is significant (in Windows abc == Abc)
- Filenames can contain almost any character but some, such as space, require quoting - avoid doing this.
- A file is associated with a unique file pathname
- A pathname is either absolute or relative to the current working directory.
  - absolute path: */dir1/dir2/dir3/filename*
  - relative path: *../dir2/dir3/filename*

File Systems - Security

- File attributes maintained in inode (aka Index node).
- Some metadata: file type and permissions, links, user and group, ownerships, size, timestamp (LMT), etc.
- GNU/Linux is a multi-user OS. Implications?
- Major security goals - the CIA triad

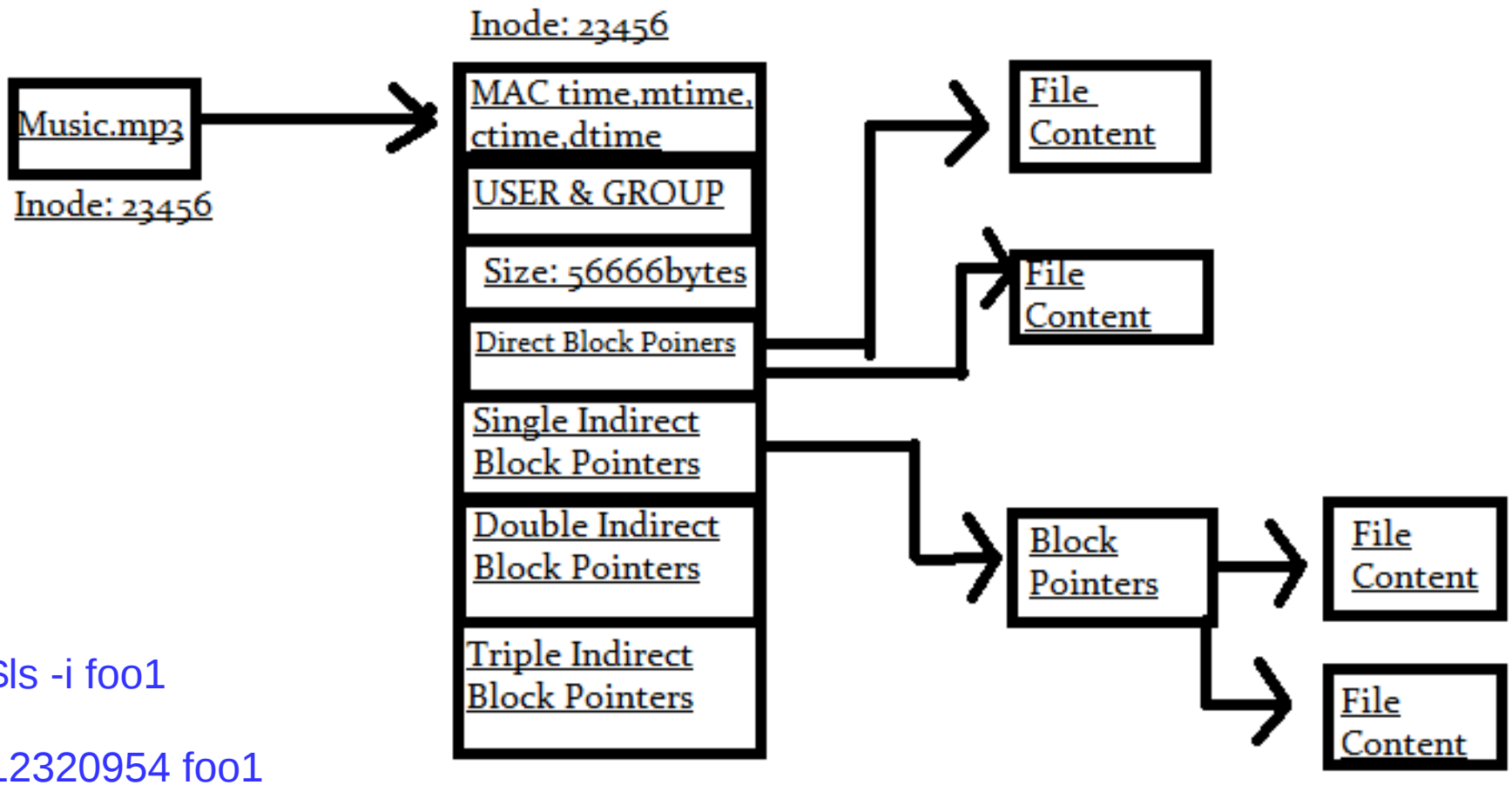
  Data
  - Confidentiality
  - Integrity
  - Availability

  User
  - Authentication
  - Authorization
  - Accountability

- Three-tier file protection system

File Systems - inode

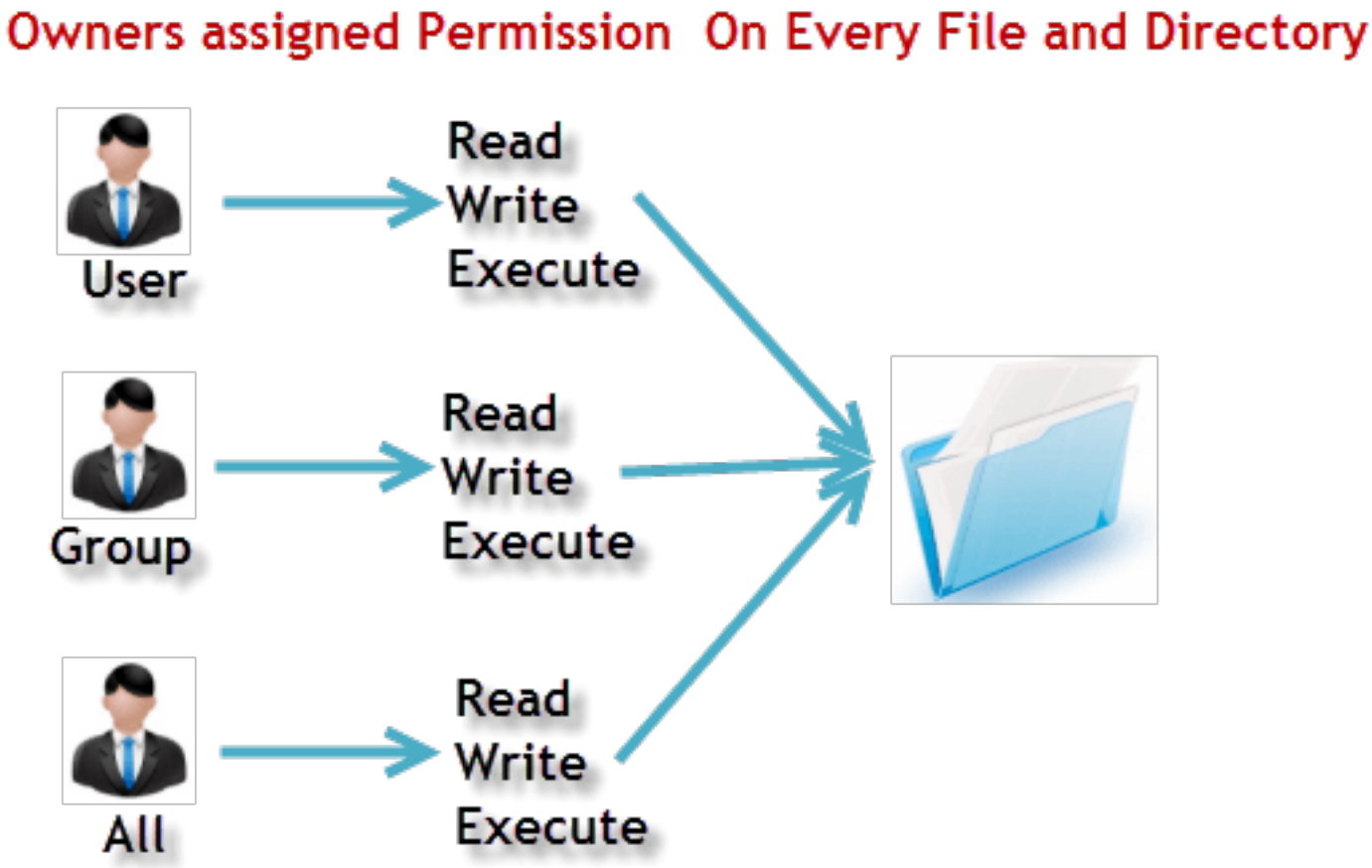
MAC – modification, access, change(permissions/group/owner) times



$ls -i foo1

12320954 foo1

File Systems - Security

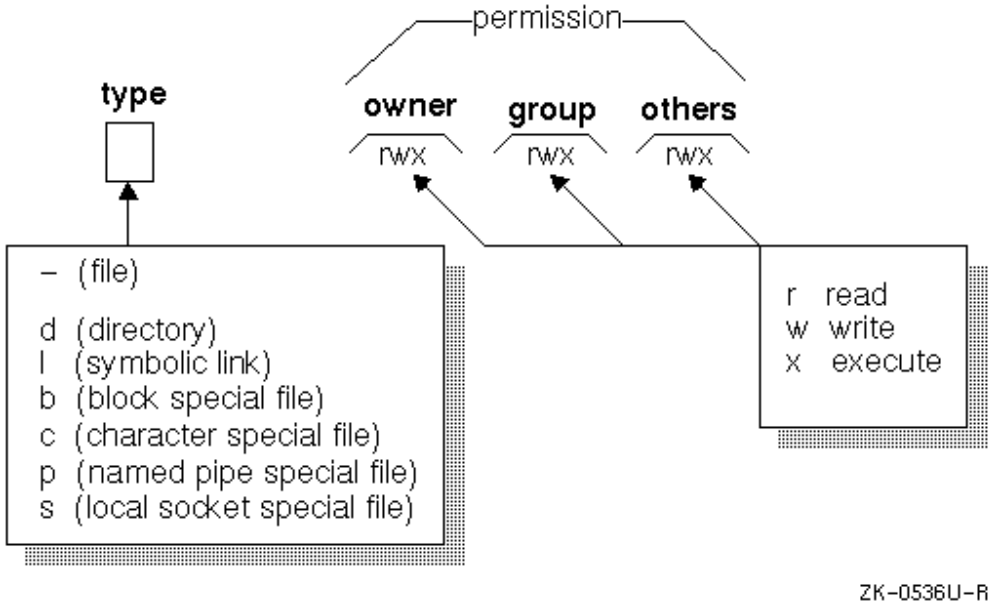


File Systems - Protection

- The basic Unix protection system consists of
  - File ownership by user and group
  - File access permissions by user, group, and other
- The ownership of a file is set when it is created, many systems do not allow a user to change it
- The access permissions are set at creation to a default and can be changed at any time
- Assess permissions are frequently presented as a 3 digit octal number
  - First octal digit is owner access
  - Second octal digit is group access
  - Third octal digit is other access

File Systems - Protection

- Each bit of the octal digit represents, from left to right,
  - read access
  - write access
  - execute access



- Examples
  - 711 => rwx--x--x
  - 644 => rw-r--r--
  - 755 => rwxr-xr-x
  - 700 => rwx------

- Use chmod command to change permissions

- Use ls -l command to view permissions

-rw-rw-r-- 1 rekha rekha 29 Jul 29 23:40 test1

File Systems - chmod

Changing Permissions: chmod

Relative vs Absolute permission assignment

chmod mode <file>

<mode> has three fields:

- user category : **u, g, o or a**
- operation : **+, - or =**
- permissions: any combination of **r, w or x**

File Systems - chmod

- A “group” is a set of users
- Each file or directory is owned by one user and one group
- Users are listed in the file /etc/passwd
- Users can be added by using adduser command from root
- Groups are listed in the file /etc/group
- Groups can be added using addgroup command from root
- Users can be added/appended to groups using usermod

```
usermod -a -G groupname username
```

Some examples of file protection modes

```
chmod 664 foo1

chmod  g-w foo1

chmod a+w foo1

chmod ug=rx foo1

chmod u=rw,g=r,o= foo1

chmod -R u+w,go-w test1
```

Default permissions

- Bitwise compliment of mask is applied to the default permissions using a operation logical And.

- The result is that the umask tells the operating system which permission bits to "turn off" when it creates a file.

- Umask
  system's current umask value
  0002
- Umask -S
  symbolic representation of umask
  u=rwx,g=rwx,o=rx

Mask to turn off the write permission for group and others

$ umask 022

$ touch new1

$ ls -l new1
-rw-r--r-- 1 rekha rekha 0 Jul 31 23:02 new1

Permission encoding used by umask		
Octal	Binary	Permissions
0	000	rwx
1	001	rw-
2	010	r-x
3	011	r--
4	100	-wx
5	101	-w-
6	110	--x
7	111	(none)

Default permissions

umask Value Octal (xyz)	Default File Permissions	666 - xyz	Default Directory Permissions	777 - xyz
000	rw-rw-rw	666	rwxrwxrwx	777
002	rw-rw-r--	664	rwxrwxr-x	775
022	rw-r--r--	644	rwxr-xr-x	755
026	rw-r-----	640	rwxr-x--x	751
046	rw--w----	620	rwx-wx--x	731
062	rw----r--	604	rwx--xr-x	715
066	rw-------	600	rwx--x--x	711
222	r--r--r--	444	r-xr-xr-x	555
600	---rw-rw-	066	--xrwxrwx	177
666	---------	000	--x--x--x	111
777	---------	000	---------	000

Change owner and group owner

chown : change file owner and group

$sudo chown user1 foo1

$ ls -l foo1

-rw-r----- 1 user1 rekha 2095 Jan 3 2018 foo1

$sudo chown rekha:user1 foo1

$ ls -l foo1

-rw-r----- 1 rekha user1 2095 Jan 3 2018 foo1

chgrp : change group ownership

$sudo chgrp user2 foo1

$ ls -l foo1

-rw-r----- 1 rekha user2 2095 Jan 3 2018 foo1

Permissions on directory?

What does x - eXecuting a directory mean?

Being allowed to "enter" a dir and gain possible access to sub-dirs.

```
$mkdir new
$ cd new
$ ls
$ cd ..
$ ls -ld new
drwxrwxr-x 2 rekha rekha 4096 Aug  3 10:09 new
$ chmod a-x new
$ ls -ld new
drw-rw-r-- 2 rekha rekha 4096 Aug  3 10:09 new
$ cd new
bash: cd: new: Permission denied
$ chmod a+x new
$ cd new
```

File compression/decompression

Gzip:

- -r: Each file in the directory foo is individually compressed/decompressed.
- Original files are replaced.

For Compression:

sudo gzip -r foo

For Decompression:

sudo gzip -dr foo

File compression/decompression

Zip:

- First argument of zip be the compressed file name.
- Doesn’t overwrite existing compressed file but updates/appends.

For Compression:

zip <output-file> <files-to-be-compressed>

zip -r test.zip junk foo

For Decompression:

unzip <files-to-be-decompressed>

unzip test.zip

File compression/decompression

Tape Archiver:

Collection of files are archived into a single large file before compression.

For compression:

tar –czvf file.tar.gz ./dirname

-c: Create an archive.

-z: Compress the archive with gzip.

-v: “verbose” mode.

-f: filename of the archive.

tar -czvf foo.tar.gz junk

For Decompression:

tar –xzvf file.tar.gz (-x instead of -c)

-x: extract an archive

tar -xzvf foo.tar.gz

For Display:

tar –tvf file.tar.gz

-t: display an archive

tar -tvf foo.tar.gz

File compression/decompression

Zip:

- First argument of zip be the compressed file name.
- Doesn’t overwrite existing compressed file but updates/appends.

For Compression:

zip <output-file> <files-to-be-compressed>

zip -r test.zip junk foo

For Decompression:

unzip <files-to-be-decompressed>

unzip test.zip

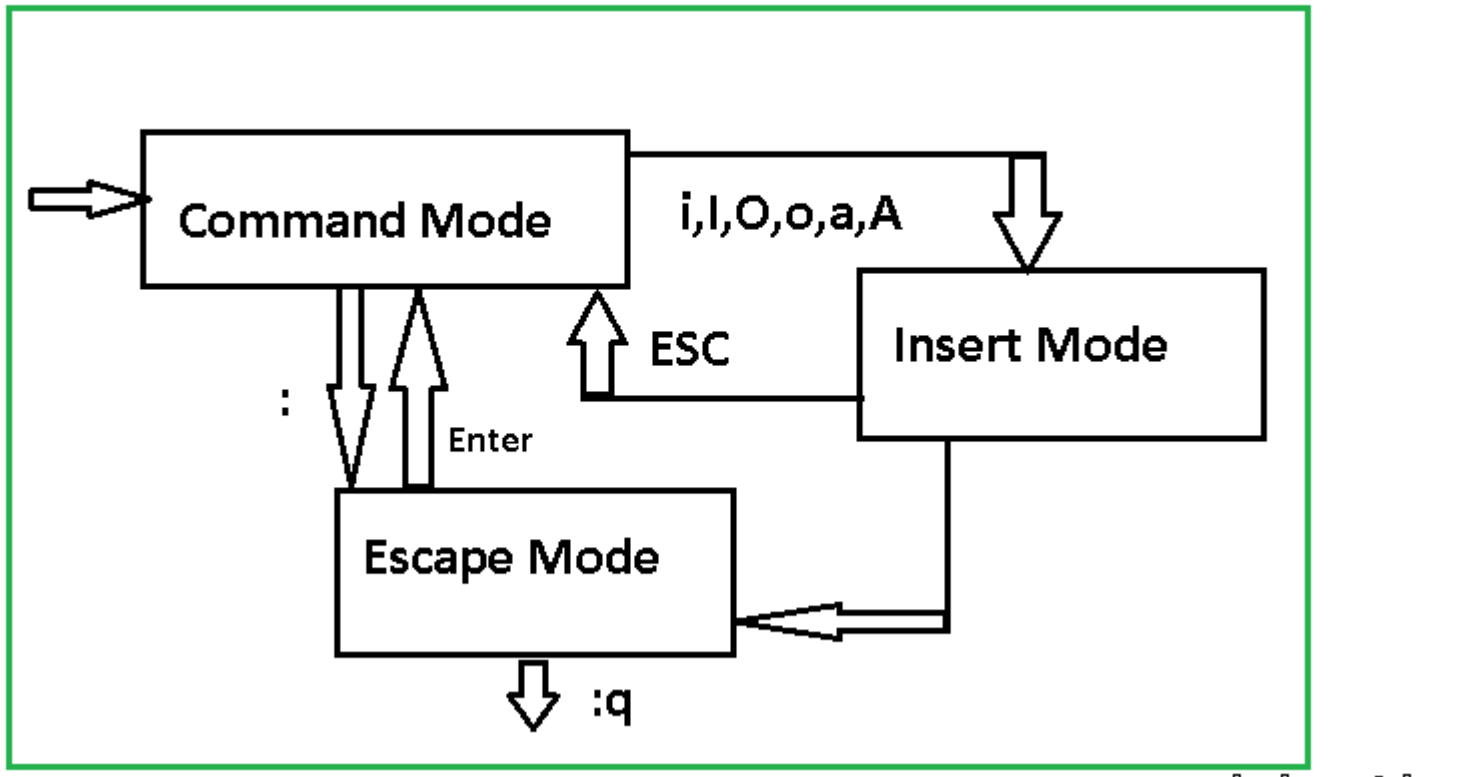
File Editing : nano

- nano is a simple text editor

  nano filename

Editing - vi

- Normally operates in 3 modes (insert text, command, escape)
- Move around screen using cursor keys
- Case-sensitive



File Editing : vi

- Why vi, fast and easy
- Basic modes- edit and command,

  ‘esc’ for command mode

  ‘i’ for insert and a’ for append mode
- Other commands using colon- :q,:w,:q!,:e

  :q for quit, :w for write, :q! quit without save

  :e open another file for editing, :wq write and quit
- Searching using ‘/’

  In command mode use ‘/’ then write the word you want to search

  ‘n’ for forward search, ‘N’ for backward search
- Search and replace

  :s/ram/mohan - will search string “ram” and replace with “mohan”
- Advanced vi – **vim**(vi improve) and **gvim**(gnuvim)

File text processing

- wc filename

  counts number of characters, words and lines in a file

- sort filename

  to sort data in a file

- uniq filename

  to omit repeated lines,Typically used after sorting

- cut -d delim -f fields file

  cuts given fields from a file with fields delimited by delim

```
$cut -d : -f 6 /etc/passwd

....
/home/user1
/home/user2
```

File text processing

- paste file1 file2

  merges lines in file1 with file2

- grep pattern file

  prints lines from file matching a pattern

- file

  to know the type of a file looking at the contents

- sed

  for performing text translations

- awk

  a programming language for scanning and processing patterns

- diff

  to show differences between two files

Autocompletion of commands

TAB key to finish a half-entered command

- Up and down arrows to browse command history

- ! to run previous command
  - !command will repeat previous invocation of command

- history command to see the earlier typed commands

  Control-R to search a previously issued command

Redirection of input: <

- command < filename
  redirects the contents of filename as input to command

$wc -l <outfile

Redirection of output : >

- command > outfile
  stores the output of command into outfile
- outfile is completely overwritten

$ls > outfile

- Can be used for creating a new file

$cat > newfile
....
ctrl D

Redirection of output : >>

- command >> outfile stores the output of command into outfile
- outfile is appened to instead of overwritten

$ls >> outfile
$cat outfile

pipes

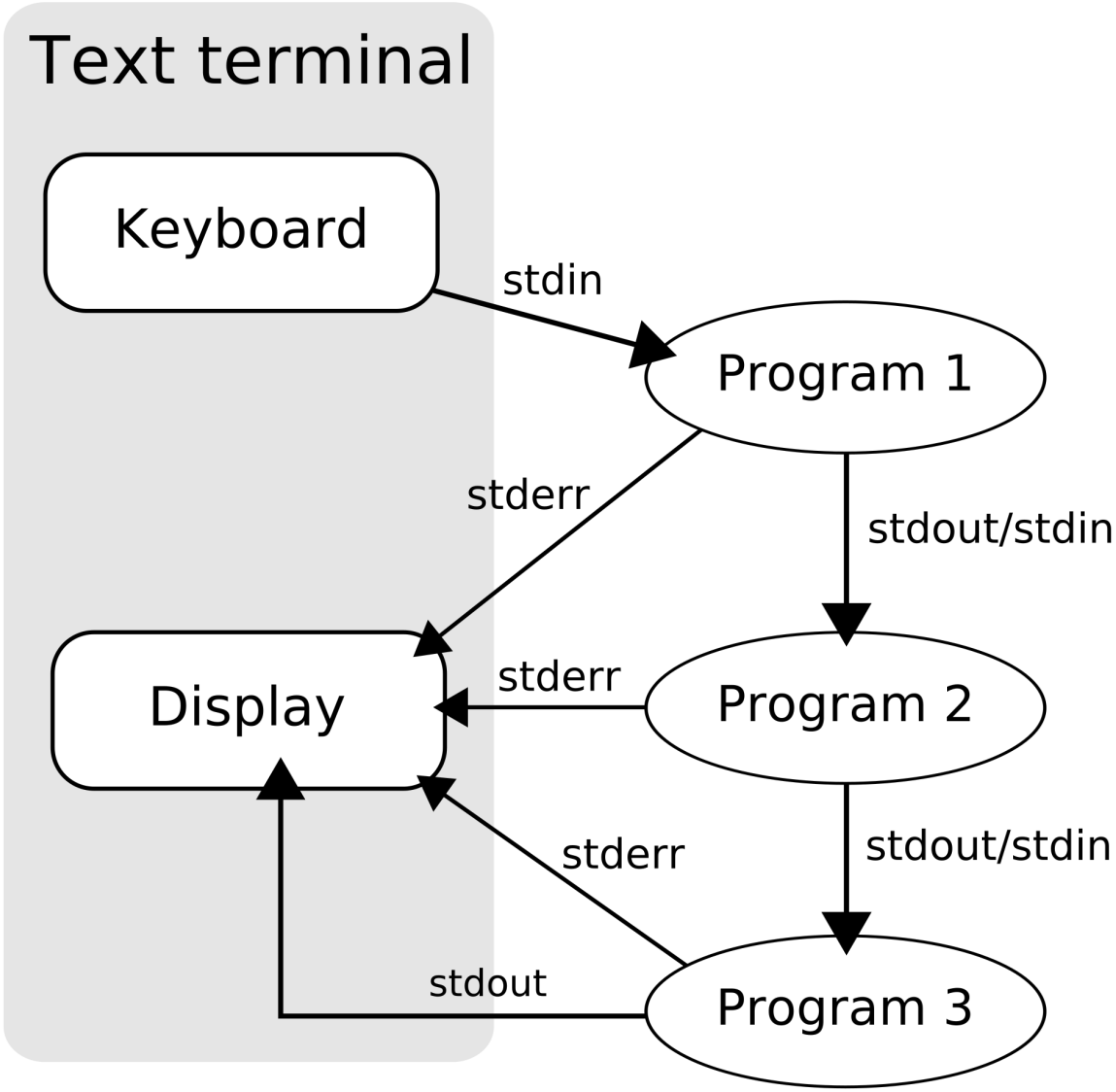
command1 | command2 | ...| command n

Output of a command piped into input for another. Redirects the output of command1 as input to command2 and output of command2 is input to comand3...Output from command n goes to STDOUT.

- STDOUT → STDIN
- The Shell does the setup, the command unaware.
- Length of the pipe can be “indefinite”

sort inventory | uniq
cat inventory | sort | uniq
history | head -20 | tail -5

Piping


Text terminal
Keyboard
stdin
Program 1
stderr
stdout/stdin
Display
stderr
Program 2
stderr
stdout/stdin
stdout
Program 3