Introduction to Linux

Mulualem Teku

GCIT, Bhutan

Aug 2019

Brainstorm

1. Free and open source software (FOSS)?
2. Linux? Ubuntu/Fedora/RedHat...? Kernel? Shell?

Brainstorm

1. Free and open source software (FOSS)?

2. Linux? Ubuntu/Fedora/RedHat...? Kernel? Shell?

Unix & Linux

- Unix is a multi-tasking, multi-user OS.
- A basis for many OSs:

Example

Berkeley Software Distribution - `BSD` (`NetBSD`, `OpenBSD`, and `FreeBSD`), Sun's `Solaris` (now `Oracle Solaris`) & the open source `OpenSolaris`, `GNU/Linux` , `OS X`, `Android`, etc.

- Linux is a Unix-like open source OS (OSS).
- Linus Torvalds wrote the core component (the kernel).
- Technically, Linux refers to this core part.

Evolution of a Revolution

- 1969 - C developed at AT&T.
- 1973 - `UNIX` rewritten in C & the code shared (to UC, Berkeley too).
- By 1975 - AT&T started selling `UNIX` ($\sim$ half written by others).
- As a result, two versions: AT&T `Unix` and `BSD Unix`.
- In the 80s, companies wrote their own versions.

  e.g. `IBM's AIX` & `SunOS` (later `SunSolaris`)
- Richard Stallman started the `GNU (GNU's Not Unix)` project to distribute free Unix-like software.
- In the 90s, Linus wrote the kernel for his 386 system and shared it online.

GNU

The goal of GNU (GNU's Not Unix):

> *"To create complete UNIX-compatible software systems entirely composed of free software."* *Richard Stallman*

- Unix-like but no unix code (hence GNU).
- The movement created many popular tools (`emacs`, `gcc`, `gdb`...).

GNU/Linux

"There really is a Linux, and these people are using it, but it is just a part of the system they use. Linux is the kernel: the program in the system that allocates the machines resources to the other programs that you run. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux." *Richard Stallman*

Linux: The Whys

- Free! (as in 'free speech')
- Portable
- Prevalence, Scalability *&* Versatility
  - Most leading hosting companies' servers run Linux (source: here)
  - 95.2% of the top fastest supercomputers (source: here)
- Large community base
- Security

Some cons:

- Many distribution choices
- Lag in software support (e.g. Photoshop, games)

Linux: The Whys

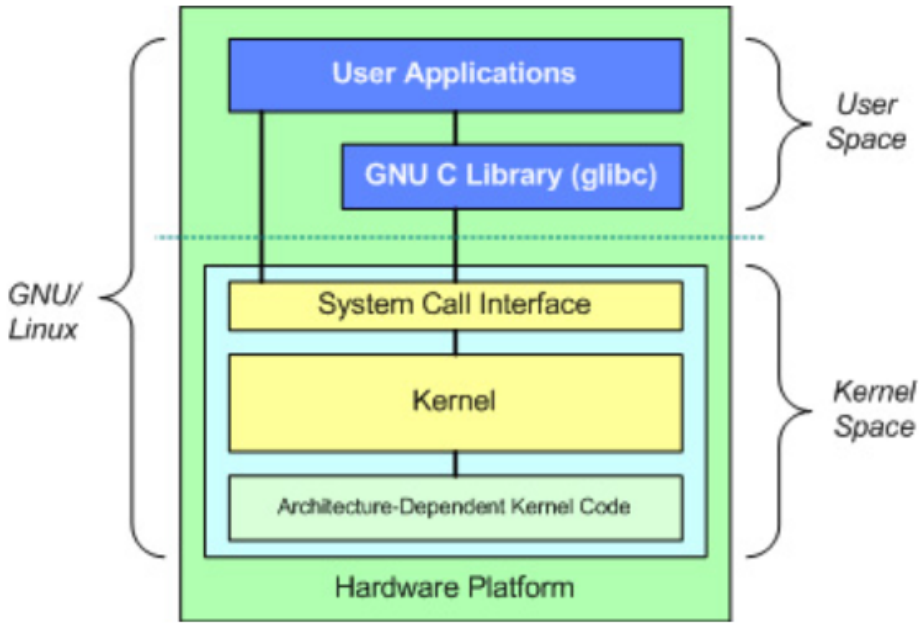
- Free! (as in 'free speech')
- Portable
- Prevalence, Scalability *&* Versatility
  - Most leading hosting companies' servers run Linux (source: here)
  - 95.2% of the top fastest supercomputers (source: here)
- Large community base
- Security

Some cons:

- Many distribution choices
- Lag in software support (e.g. Photoshop, games)

Linux Architecture

The Standard OS Model

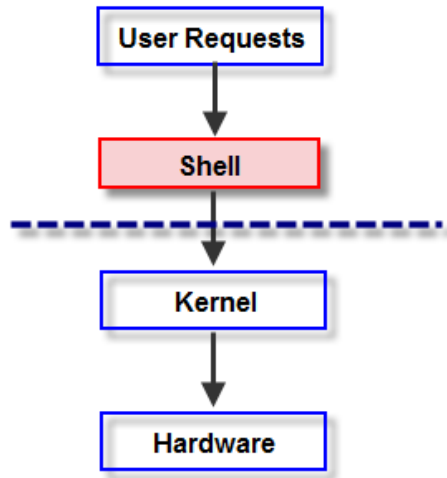


Basic Architecture of the GNU/Linux OS

Linux Architecture: The Shell

A program (a.k.a. command line interpreter) that allows the user to interact with the `UNIX/Linux` system.

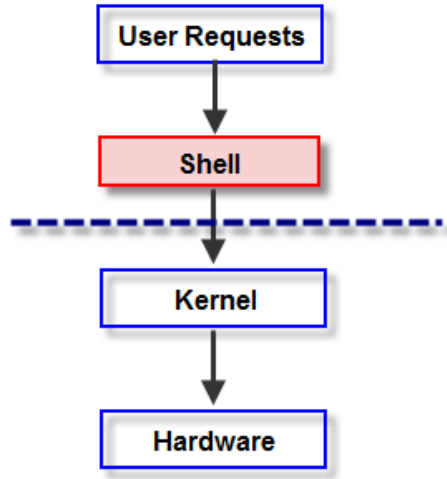
- Reads user's input.
- Parses it (evaluates special characters if any).
- Works with the kernel to execute the command.



Linux Architecture: The Shell

A program (a.k.a. command line interpreter) that allows the user to interact with the `UNIX/Linux` system.

- Reads user's input.
- Parses it (evaluates special characters if any).
- Works with the kernel to execute the command.



Shell Script

A regular text file that contains executable shell or `Linux` commands.

Examples

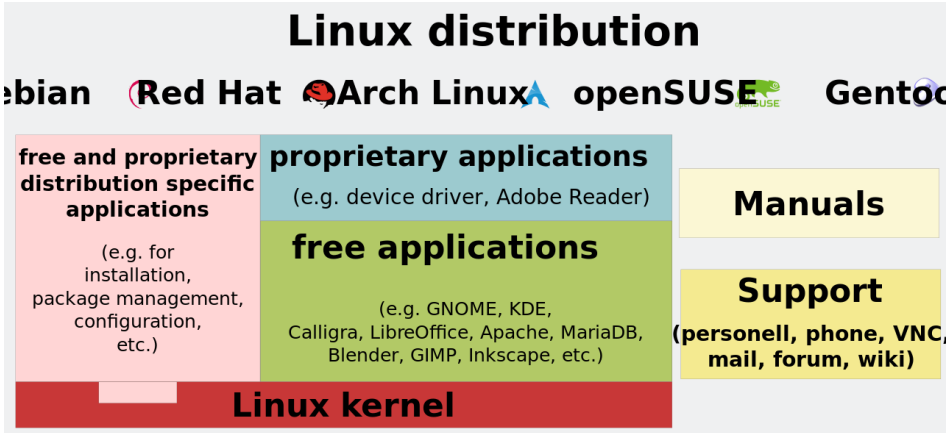
`Bourne shell (sh)`
`Bourne again shell (Bash)`
`C shell (csh, tcsh)`
`Korn shell (ksh)`

Linux Distributions (Distros)

Distro = a collection of software, often open source, on top of a kernel.

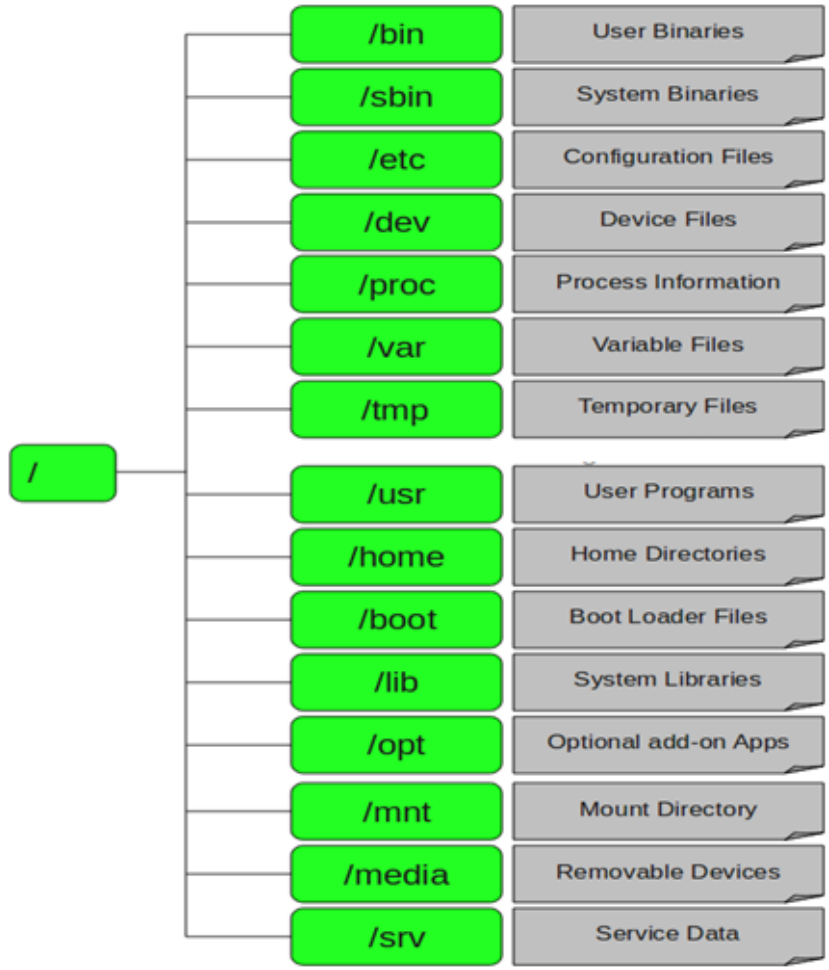
- Different vendors distribute kernel + GNU + non-GNU components (e.g. desktop applications, server software, system management tools, documentation...)

- 300+ active distros; some more popular:



Linux: The File System

- In Linux, everything is a file!
- Hierarchical organization
- Absolute vs Relative paths
  - ~ (tilde) - the home directory
  - . (a dot) - the current directory
  - .. (double dot) - the parent directory



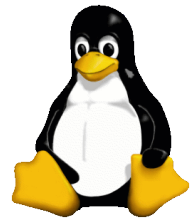
The File System: File Types

1. `Text files`: human-readable
   e.g. documentation, application settings, source code, logs
2. `Binary files`: executables, libraries, media files, ...

The File System: File Types

1. `Text files`: human-readable
   e.g. documentation, application settings, source code, logs
2. `Binary files`: executables, libraries, media files, ...

- `Regular/Ordinary file`: contains printable/non-printable stream of characters.
- `Directory file`: maintains info about files it houses (e.g. name, inode number).
- `Device/special file`: contains attributes of a device (e.g. printer, CD-ROM) used by the kernel.



Basic Commands

General Syntax

```
<SomeCommand> [option 1] [option 2] ...[option n]
```

General Syntax

```
<SomeCommand> [option 1] [option 2] ...[option n]
```

Print Working Directory

`pwd`

- Displays full path of the current directory.

General Syntax

```
<SomeCommand> [option 1] [option 2] ...[option n]
```

Print Working Directory

`pwd`

- Displays full path of the current directory.

The List command

`ls [flags] [file]`

- Lists directory content
- Flags/options: -l, -a, -s, -S, -t ...

Basic Linux Commands (2)

Change Directory

`cd [dir]`

- Changes directory to `[dir]`.
- Defaults to user's home directory if `<dir>` not given.

Basic Linux Commands (2)

Change Directory

`cd [dir]`

- Changes directory to `[dir]`.
- Defaults to user's home directory if `<dir>` not given.

Know Your System

- `echo $SHELL`
- `uname [-a]`
- `whoami`
- `w(ho)`
- `ifconfig [-a]`
- `route`
- `df -h, du -h, free -m`

File & Directory Commands: Creation

Creating Files

`touch [flags] <file>` (easiest way)

- If the file exists, timestamp modified.
- If not, the file is created.

File & Directory Commands: Creation

Creating Files

`touch [flags] <file>` (easiest way)

- If the file exists, timestamp modified.
- If not, the file is created.

Creating Directories

`mkdir [flags] <dir name>`

- Creates a directory with the name `<dir name>`.

View + Concatenate

```
cat <file>
cat <file1> <file2> ...<file n>
od   [flags] <file>        (octal display)
    e.g.  od -bc /bin/ls
```

File & Directory Commands: Display

View + Concatenate

```
cat <file>
cat <file1> <file2> ...<file n>
od   [flags] <file>      (octal display)
    e.g.  od -bc /bin/ls
```

More & Less

`more <filename>`

Scrolls 1 page @ a time (space bar)

`less <filename>`

Scrolls up/down by pages/lines

File & Directory Commands: Display

View + Concatenate

```
cat <file>
cat <file1> <file2> ...<file n>
od   [flags] <file>        (octal display)
    e.g.  od -bc /bin/ls
```

More & Less

`more <filename>`

Scrolls 1 page @ a time (space bar)

`less <filename>`

Scrolls up/down by pages/lines

Head & Tail

`head -[numlines] <filename>`

`tail -[numlines] <filename>`

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 & Directory Commands: 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 & Directory Commands: Deletion

Remove File

```
rm [flags] <file>
rm -i <filename>                (prompts - good idea!)
alias rm="rm -i"                is called aliasing
```

- Be cautious!
- Use wildcards (more about them later) to delete multiple files.

Remove Directory

```
rmdir [flags] <directory>       (empty directory)
rm -r <directory>               (directories + subdirectories)
```

- Be extremely cautious!!

File & Directory Commands: Deletion

Remove File

```
rm [flags] <file>
rm -i <filename>                (prompts - good idea!)
alias rm="rm -i"                is called aliasing
```

- Be cautious!
- Use wildcards (more about them later) to delete multiple files.

Remove Directory

```
rmdir [flags] <directory>          (empty directory)
rm -r <directory>          (directories + subdirectories)
```

Be extremely cautious!!

Getting Help

The `manual` command

`man <command>`

- Displays the manual page (manpage) of `<command>`.
- Use `/<keyword>` to do a keyword search in a manpage
- Make `man` your best friend.

Getting Help

The manual command

`man <command>`

- Displays the manual page (manpage) of `<command>`.
- Use `/<keyword>` to do a keyword search in a manpage
- Make `man` your best friend.

whatis & info

`whatis <command>`

`info <command>`

Getting Help

The manual command

`man <command>`

- Displays the manual page (manpage) of `<command>`.
- Use `/<keyword>` to do a keyword search in a manpage
- Make `man` your best friend.

whatis & info

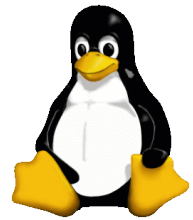
`whatis <command>`

`info <command>`

The apropos command

`apropos <keyword>`

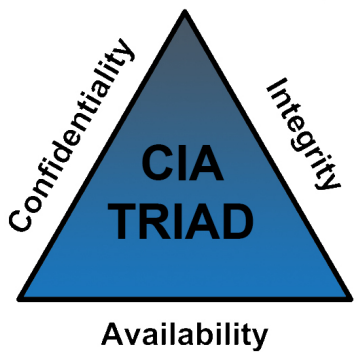
- Finds all commands containing the keyword.



Working with Files

1) File Security

- GNU/Linux is a multi-user OS. Implications?
- Major security goals - the CIA triad
  - Confidentiality
  - Integrity
  - Availability



  - Authentication
  - Authorization
  - Accountability
- File attributes maintained in inode.
  - file type and permissions, links, user and group ownerships, size, timestamp (LMT)

File Permissions

- Different user accounts with different file access privileges/permissions.
- Three-tiered file protection system

Format

`[type]rwxrwxrwx`

- [type]= - (ordinary) or d (directory) or l (link)
- user's permissions
- Group's permissions
- Others' (world's) permissions
- r=read, w=write, x=execute

Changing Permissions: chmod

chmod (change mode)

```
chmod [-R] <mode> <file>
```

`<mode>` has three fields:

- *user category*: `u, g, o or a`
- *operation* : `+, - or =`
- *permissions*: any/combination of `r, w or x`
- Can be done using octal numbers too (read=4, write=2, execute=1)
- The `umask` command reveals default permissions. But it can be set!

- Relative vs Absolute permission assignment

- Directory permissions

Changing [Member|Owner]ship

`chgrp` (change group)

`chgrp <group> <file>`

- Changes the group membership of `<file>` to a new group, `<group>`.

Changing [Member|Owner]ship

`chgrp` (change group)

```
chgrp <group> <file>
```

- Changes the group membership of `<file>` to a new group, `<group>`.

`chown` (change owner)

```
chown <user>[:group] <file>
```

- Assigns to `<user>` the ownership of `<file>` (`[group]` is optional).
- For the root/super user

2) File Compression/Archival

`tar` (tape archiver)

```
tar [options] <archive name> <files>
```

- A utility to archive multiple files together.
- No compression!
- Common options: `-c` (create), `-x` (extract), `-t` (list), `-f` (filename)

```
tar -cvf  Myarchive.tar file1 file2 file3        (creates)

tar -xvf  Myarchive.tar                          (extracts)

tar -tvf  Myarchive.tar                    (displays contents)
```

gzip/gunzip, bzip/bunzip, bzip2/bunzip2

`gzip [options] <file>`

`gzip -d <file.gz> / gunzip <file.gz>` (decompression)

- Compression/decompression tools.
- Outputs a compressed file of `.gz` ext; original file removed.

```
gzip hello.c hello.html hello.sh

gzip -l hello.html.gz                    (amount of compression)

gzip -d hello.c.gz hello.html.gz

gunzip hello.c.gz hello.html.gz

gzip .                                   (?)
```

1. Compressed Archives using `zip/unzip`

zip/unzip

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

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

```
zip lectures.zip lecture1.pdf lecture2.pdf

zip -r backup.zip .                    (recursive compression)

unzip lectures.zip                     (the -v flag?)
```

2. Compressed Archives using `tar`:
   - With `-z` option, `tar` compresses using `gzip` (`tar -cvzf file.tar.gz`)
   - With `-j` option, `tar` compresses using `bzip2` (`tar -cvjf file.tar.gz`)

1. Compressed Archives using `zip/unzip`

zip/unzip

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

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

```
zip lectures.zip lecture1.pdf lecture2.pdf

zip -r backup.zip .                    (recursive compression)

unzip lectures.zip                     (the -v flag?)
```

2. Compressed Archives using `tar`.
   - With `-z` option, `tar` compresses using `gzip` (tar -cvzf file.tar.gz)
   - With `-j` option, `tar` compresses using `bzip2` (tar -cvjf file.tar.gz)

3) Remote File Access

ssh (secure shell)

```
ssh [options] [username@]<remote-machine-name/IP address>
```

- `ssh` daemon (`sshd`) must be listening on some port (often port 22).
- Remote machine be configured to accept incoming `SSH` connections.
- Can be used to execute remote commands.
- Common options: `-X/-Y` (imports `X11` - graphical window), `-f` (puts `ssh` into the background before executing the remote command).

```
ssh me@example.com

ssh -X me@example.com firefox        (run Firefox remotely)
```

scp (secure copy)

```
scp [-r] <file> username@remote machine:          (a)
scp username@remote machine:<file> <target>       (b)
```

- (a) copies `<file>` *to* the remote machine over an encrypted channel.
- Notice the colon (:) It is necessary.
- (b) copies `<file>` *from* the remote machine to `<target>`.

```
scp -r MyDocuments me@abc.org:                    (export)

scp me@abc.org:myfile .                           (import)
```

sftp (Secure File Transfer Protocol)

```
sftp username@remote machine
```

- Transfers files between local and remote machines securely.
- Uses an interactive console.
- Same connection settings as `ssh`.

- Common commands include:
  - `help`
  - `get -` download from remote machine
  - `put -` upload to remote machine
  - `cd / pwd / ls -` (on remote machine)
  - `lcd / lpwd / lls -` (on local machine)

Other network-related commands/diagnostic tools you may find useful:

- `ping`
- `host`
- `traceroute`
- `wget`
- `curl`

Consult `man` for more. Again, make `man` your best friend.

File Operations: Editing

- `Vi/Vim` (Vi improved) is a lightweight but powerful text editor.
- Other common text editors: `pico, nano, emacs, gedit ...`
- Uses 3 modes to speed up editing:
  1. *Normal/Command mode* (shortcut key: `esc`)
     - `Vi(m)` starts in this mode.
     - To view the text but not edit it.
     - Also to issue a command.
  2. *Insert/Input mode* (shortcut key: `i`)
     - To type text into the file (buffer)
  3. *Visual mode* (shortcut key: `v`)
     - To highlight text and perform operations on selected text

Vi/Vim Commands

Vi Help

```
:help
```

Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Vi/Vim Commands

Vi Help

```
:help
```

Save (write) file

```
:w <filename>
```

Open another file

```
:e <filename>
```

Vi/Vim Commands

Vi Help

`:help`

Save (write) file

`:w <filename>`

Open another file

`:e <filename>`

Editing commands

Copy (yank) $\Longrightarrow$ `y`	(try: `yy`, `yw`, {n}`yy`)
delete $\Longrightarrow$ `d`	(try: `dd`, `dw`, {n}`dd`)
Paste $\Longrightarrow$ `p`	{n} is No of lines
undo $\Longrightarrow$ `u`	
redo $\Longrightarrow$ `ctrl + R`	

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Searching

`/pattern`	(search forward)
`?pattern`	(search backward)
`n`	(Repeat the last pattern search)

Moving between lines

`0 (zero)`	(beginning of line)
`$`	(end of line)
`<n>|`	(move to the nth column)
`<n>G`	(Go to line number `<n>`)

Searching

`/pattern`	(search forward)
`?pattern`	(search backward)
`n`	(Repeat the last pattern search)

Useful Turn-ons

`:set spell`	(spell check)
`:set number`	(line number)
`:syntax on`	(syntax highlighting)

Modifying Environment

`:sp`	(horizontal split)
`:vsp`	(vertical split)
`ctrl+w`	(move around)

Modifying Environment

`:sp`	(horizontal split)
`:vsp`	(vertical split)
`ctrl+w`	(move around)

Quit

`:q`	
`:q!`	(Quit without saving)
`:wq` or `:x`	(Save and quit)

For more on `Vi(m)`, checkout the built-in vimtutor!

Checkpoint!

- Notion of permission for regular files and directories ?

Assume a system with `umask` value set to 022. Create a file inside a new directory.

1. case 1: -w for the directory only
2. case 2: -w for the file only
3. case 3: -w for both

Now, do `rm`/`mv` on the file. What happens?

- Sources/inputs & destinations/outputs of Linux commands?

Checkpoint!

- Notion of permission for regular files and directories ?

  Assume a system with `umask` value set to 022. Create a file inside a new directory.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

  Now, do `rm/mv` on the file. What happens?

- Sources/inputs & destinations/outputs of Linux commands?

Checkpoint!

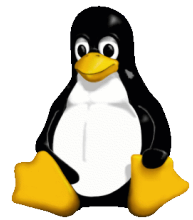
- Notion of permission for regular files and directories ?

  Assume a system with `umask` value set to 022. Create a file inside a new directory.

  1. case 1: -w for the directory only
  2. case 2: -w for the file only
  3. case 3: -w for both

  Now, do `rm/mv` on the file. What happens?

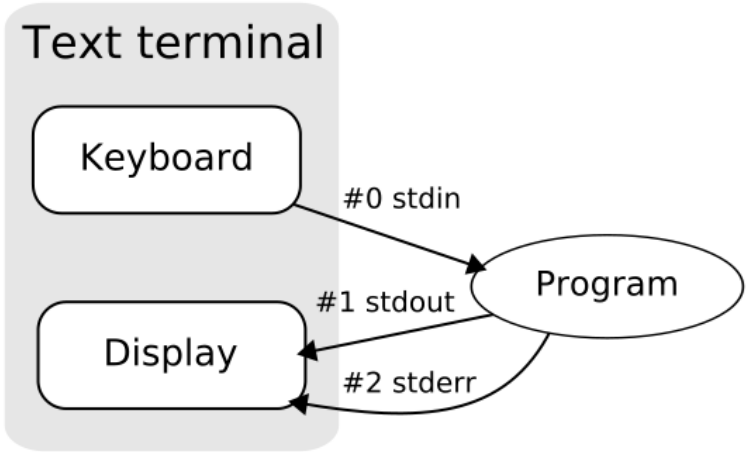
- Sources/inputs *&* destinations/outputs of Linux commands?



I/O Redirection *&* Piping

I/O Redirection

- The shell reads input *&* writes output as a stream of characters. (stream = sequence of bytes)
- Command outputs: result or status/error messages. Sent to?
- The shell provides 3 special files @ login, each with a unique file descriptor value.
- Each is associated with a default device
  1. `#0 - Standard Input stream (STDIN)`: input to commands.
  2. `#1 - Standard Output stream (STDOUT)`: output from commands.
  3. `#2 - Standard error stream (STDERR)`: errors from commands.



I/O Redirection

- A way of unhooking a stream from its default device.
- Changing where input comes from/output goes to.
- The operators:
  1. Input redirection: `0< or just <`
  2. Output redirection: `> or >>`
  3. Error redirection: `2> or 2>>`

Examples

```
wc -l < /usr/share/dict/words
```

```
cat < input.txt > output.txt                    (try: cat, cat << END)
                                                (a.k.a. Here-document)
```

```
ls -l IExist.txt IExistNot.txt > output.txt 2>> log.txt
```

```
ls -l IExist.txt IExistNot.txt &> output.txt
```

```
ls -l IExist.txt IExistNot.txt 2>&1                (any difference?)
```

```
cat /etc/passwd > /dev/null
```

Piping (|)

`command1 | command2 | ...| command n`

- Output of a command piped into input for another.
- `STDOUT` $\rightarrow$ `STDIN`
- Length of the pipe can be "indefinite"
- Redirection vs Piping (`> vs |`)? E.g. No of files in ./ ?

Example

```
who | wc -l

history | head -10 | tail -5

ls -l | sort -k 8 > output.txt
```

Piping (|)

`command1 | command2 | ...| command n`

- Output of a command piped into input for another.
- `STDOUT` $\rightarrow$ `STDIN`
- Length of the pipe can be "indefinite"
- Redirection vs Piping (`> vs |`)? E.g. No of files in ./ ?

Example

```
who | wc -l

history | head -10 | tail -5

ls -l | sort -k 8 > output.txt
```