# GNU/Linux foundation Commands

Ver 2.8.2

## *ILG Labs*

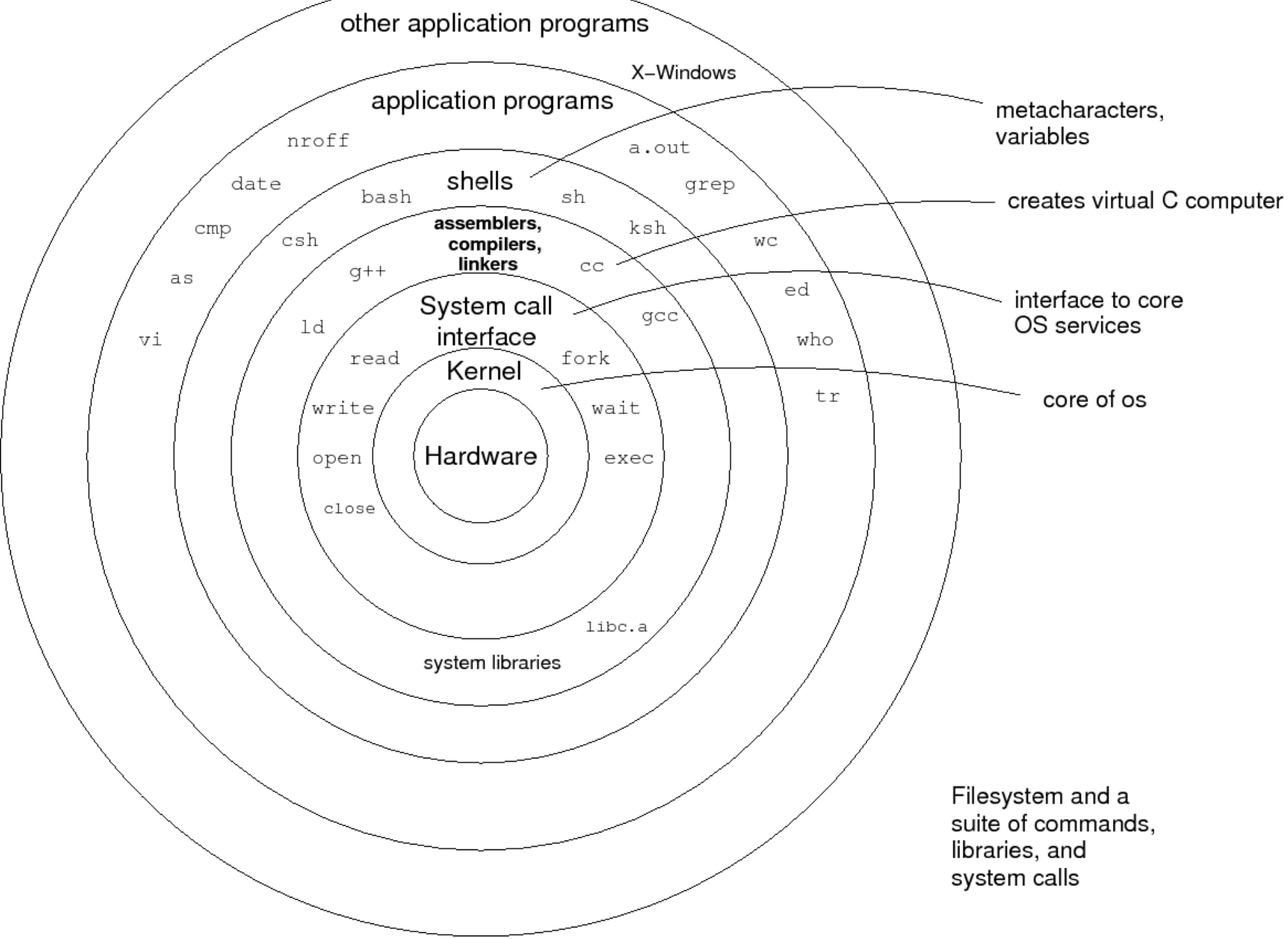
## Insight GNU/Linux Group

*Reinventing the way you,*
*Think,*
*Learn,*
*Work*

## GNU IT Solutions & Services

*Discovering infinite possibilities*
*Www.gnugroup.org*

*Practice is the not the thing you do once you are good.*
*It is the thing you do that makes you good*



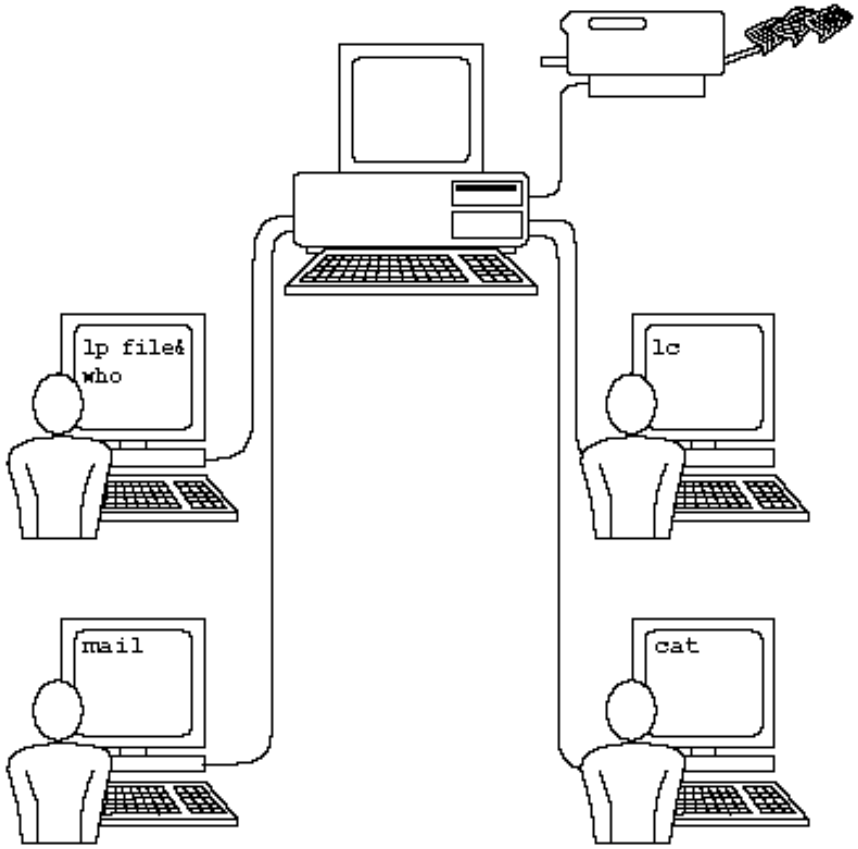

Conceptual Architecture of UNIX SYSTEMS

"The only thing standing between you and your goal—is the bullshit story you keep telling yourself as to why you can't achieve it."

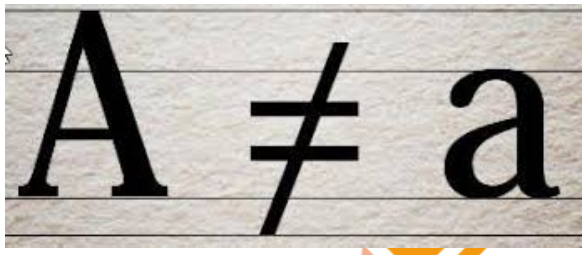
— Jordan Belfort

# WARNING!

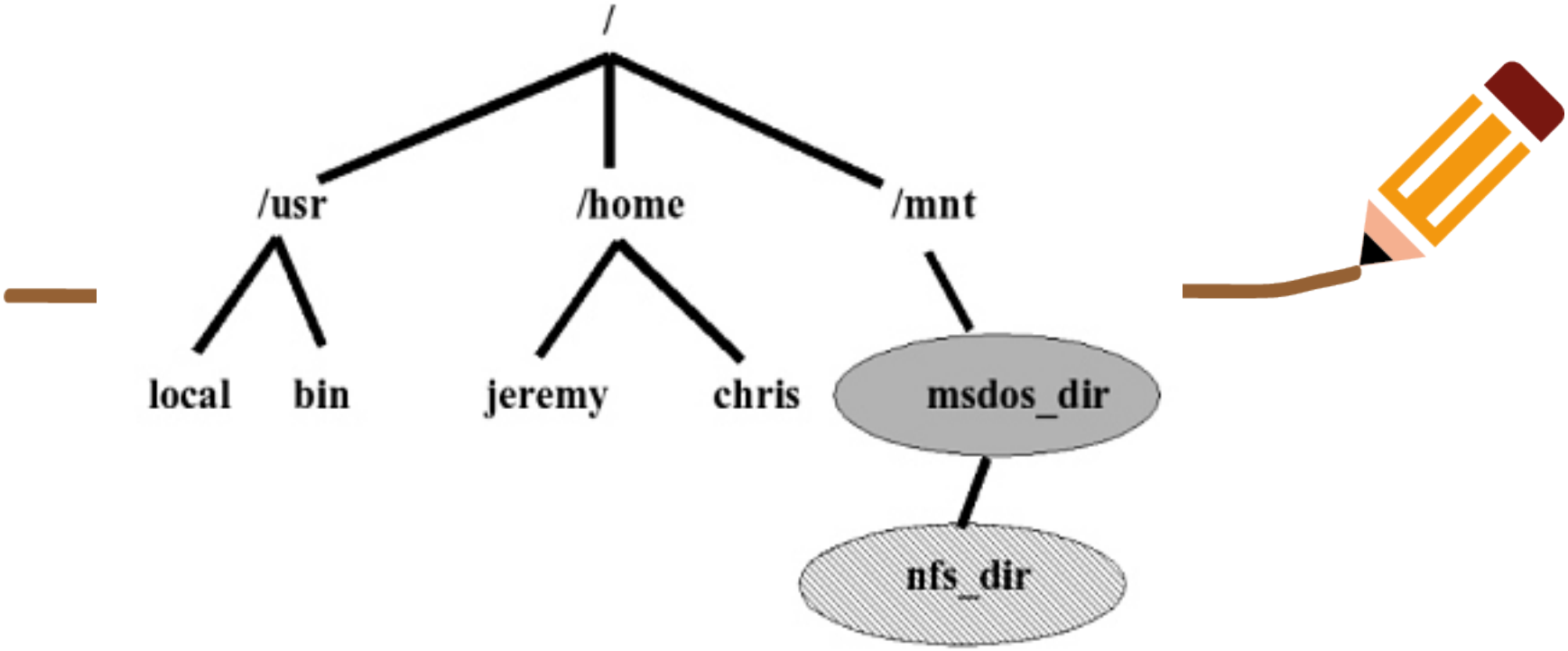
1) UNIX/Linux is a multiuser & multitasking OS.



2) All of Unix is case sensitive.

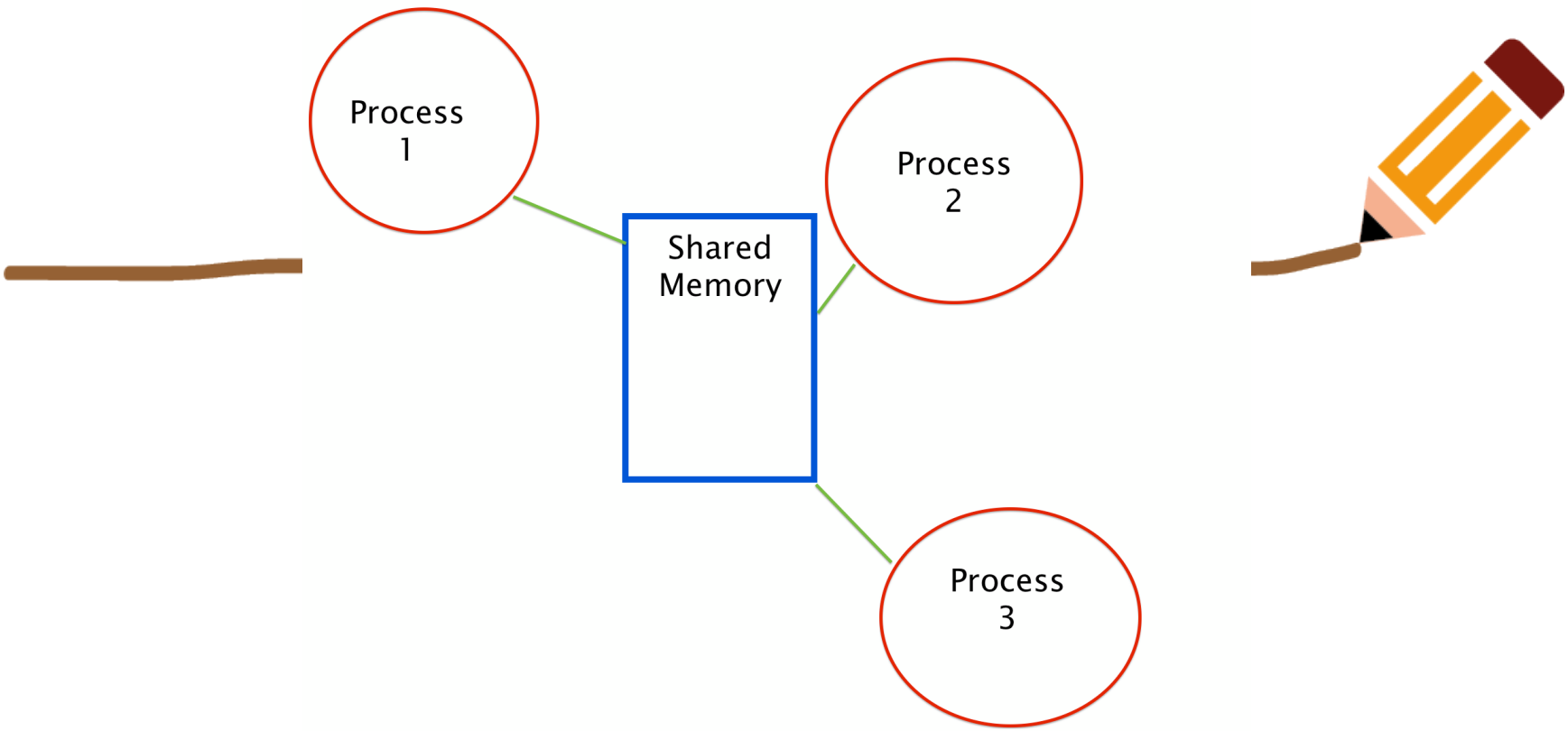



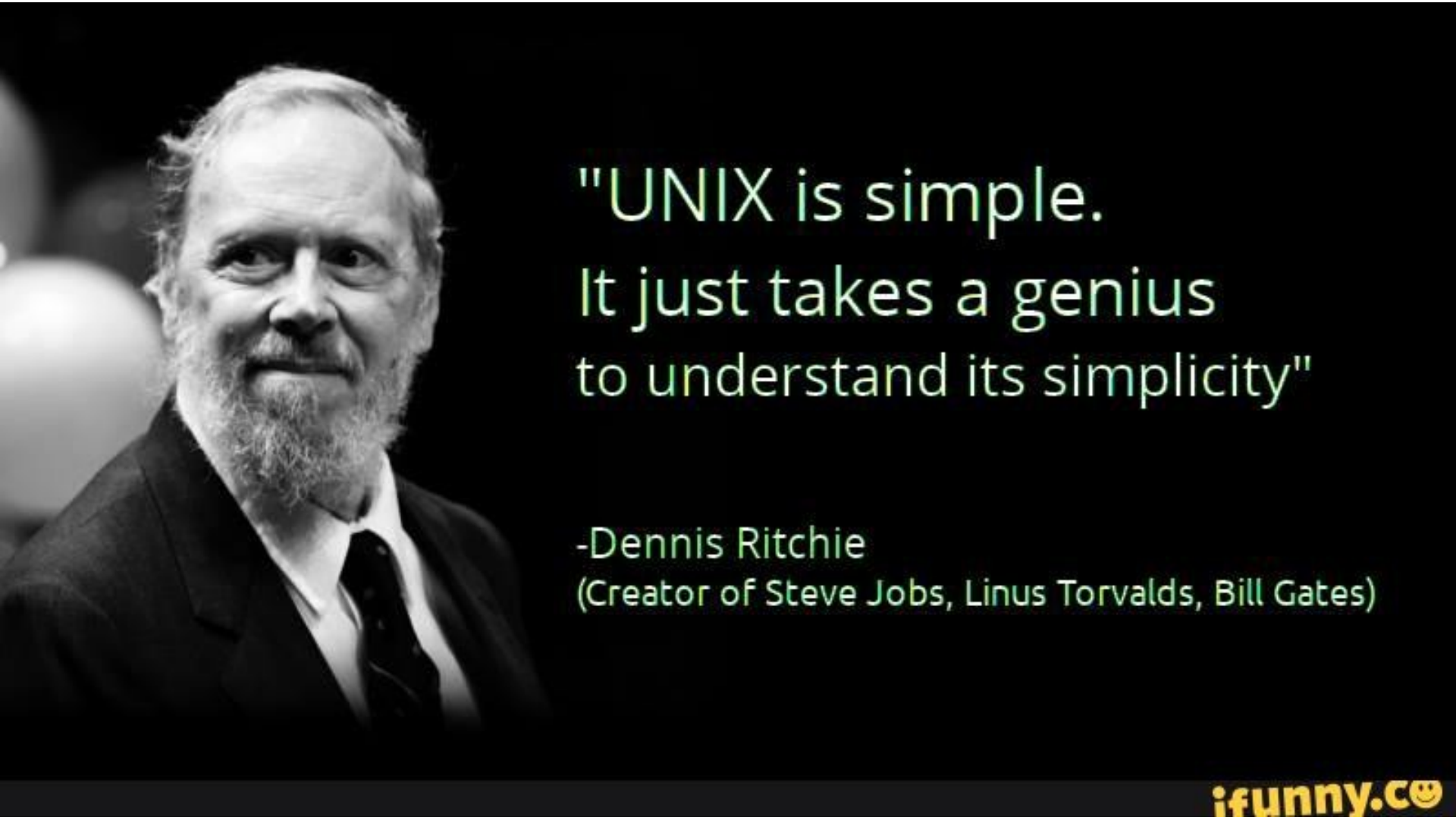
3) It does not bother about filename or file extensions



# 4) **Everything** is a **file** or a **process**

# 5) **IPC** - Interprocess communication.




"UNIX is simple.
It just takes a genius
to understand its simplicity"
-Dennis Ritchie
(Creator of Steve Jobs, Linus Torvalds, Bill Gates)
ifunny.co

# Shell Prompt

Now that you have logged in, you will see a **shell prompt**.

**[root@localhost /root]#**

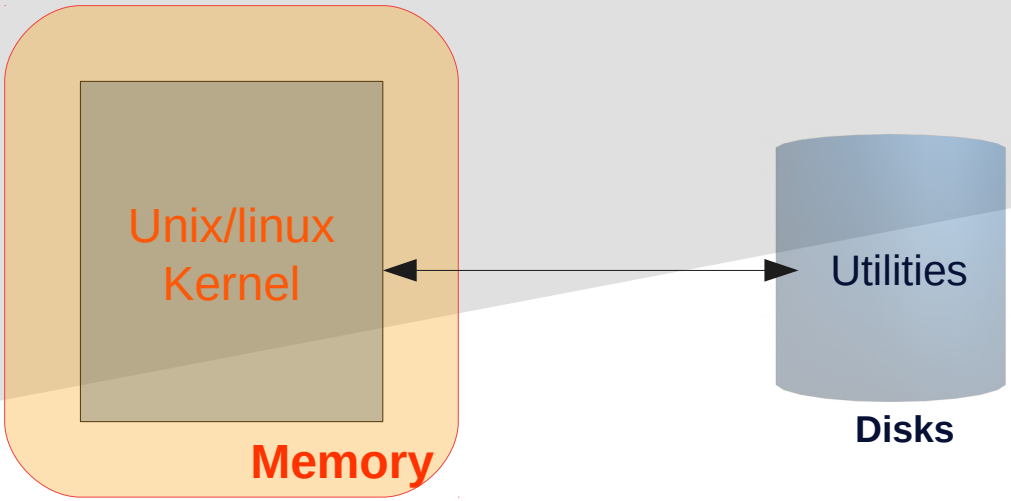
This is where you will spend most of your time as *system administrator*.

# What is a Shell ?

Unix/Linux is logically divided into 2 sections

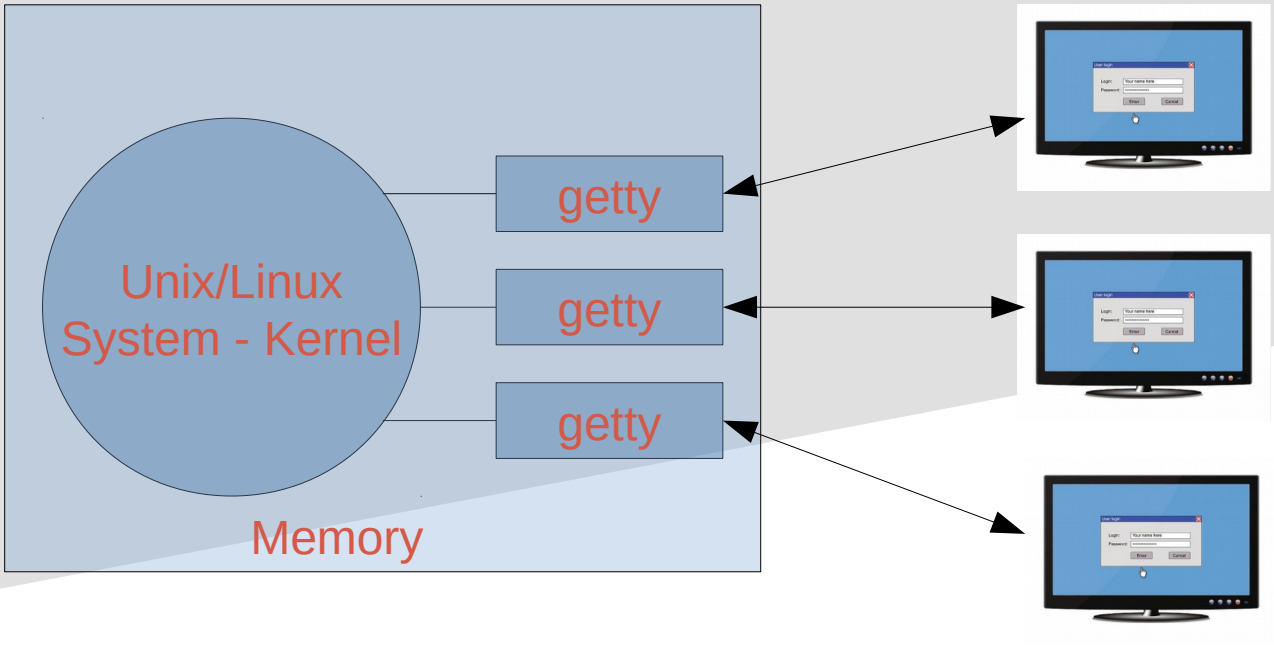
1) Kernel

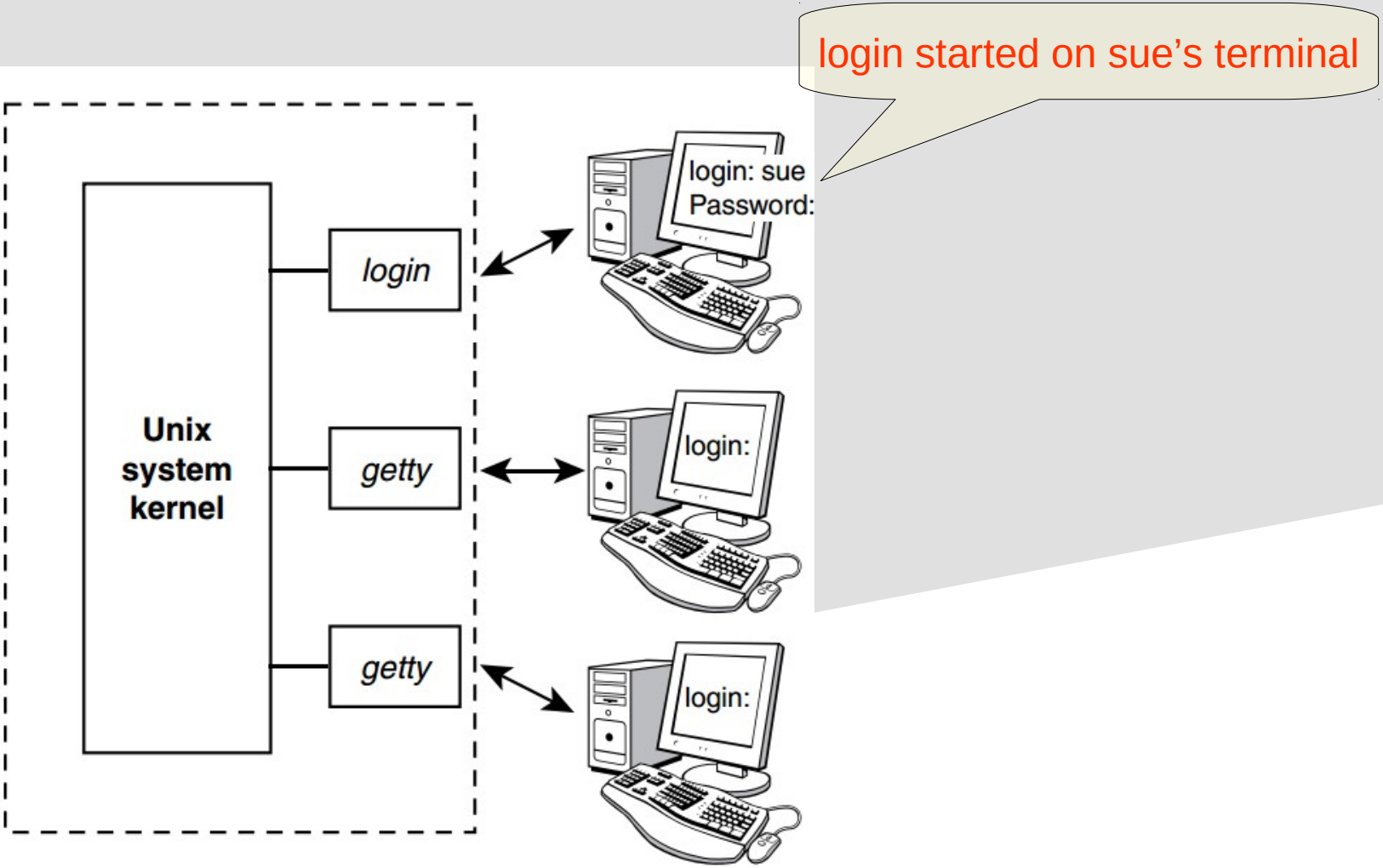
2) Utilities i.e commands



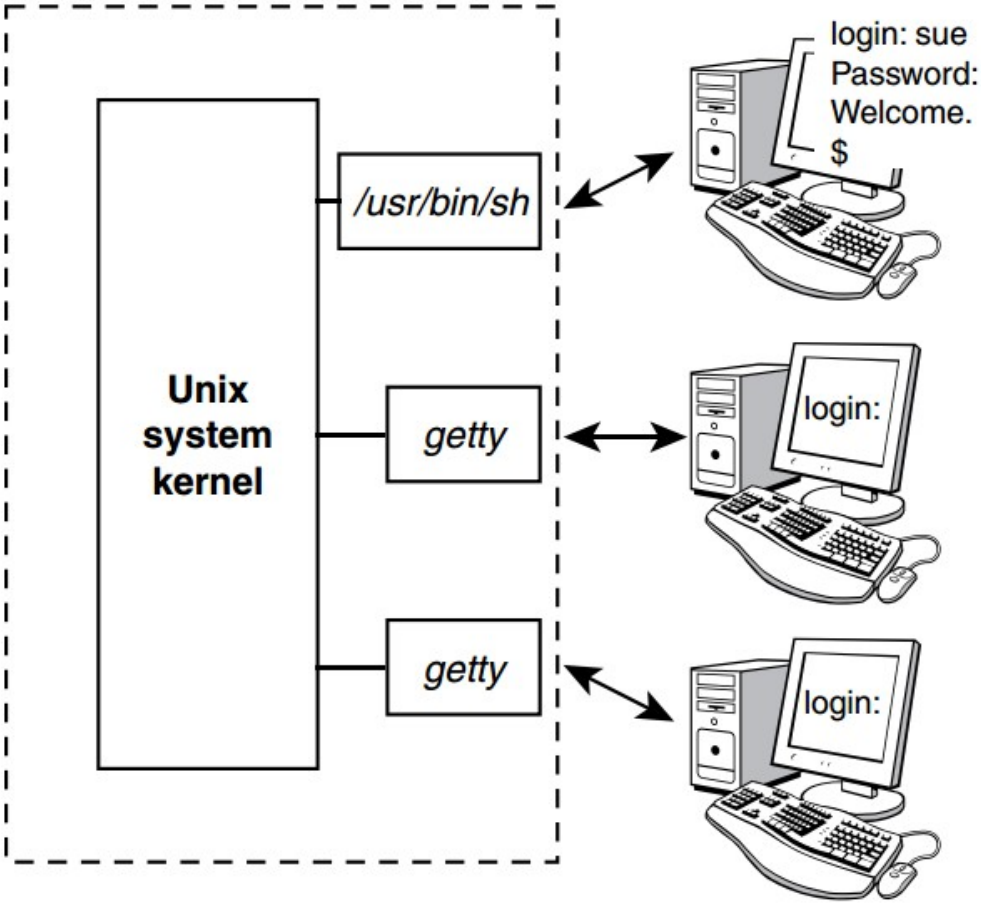
# Login Shell

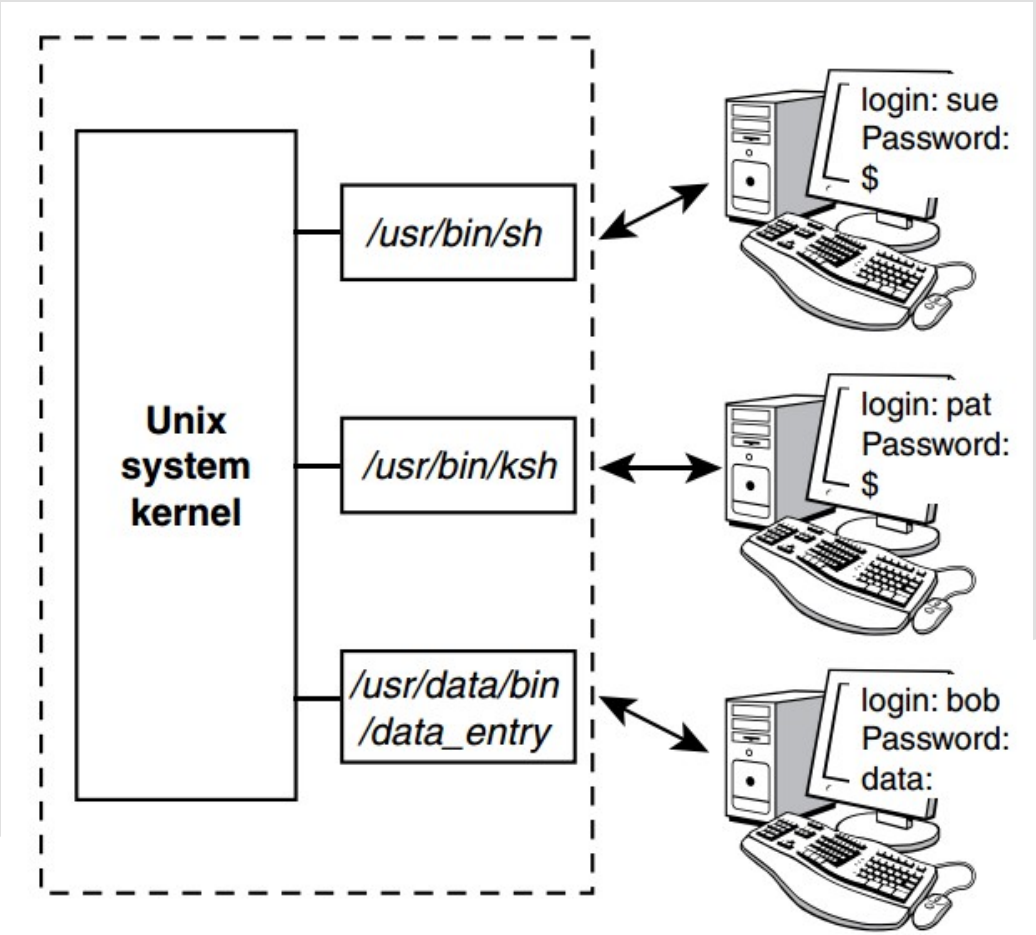
Terminal, xterm




login started on sue's terminal
Unix system kernel
login
getty
getty
login: sue
Password:
login:
login:

login executes /usr/bin/sh






Three users logged in

# Logging Out of Root

**Just type exit at the prompt, as in:**

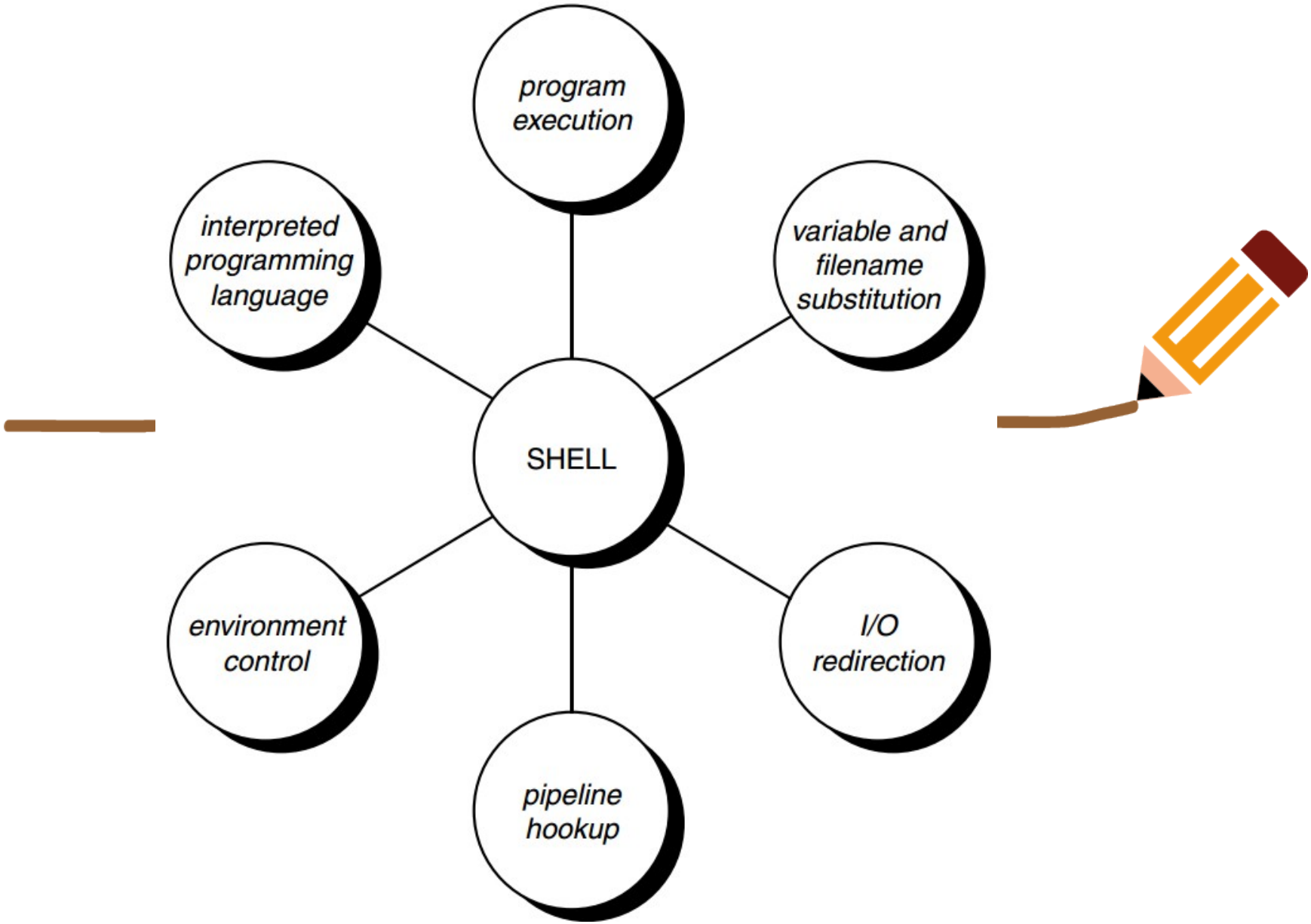
```
[root@localhost /root]# logout
```

**or by using the key combination of**

```
[Ctrl]-[D]
```

**Or just type logout at the prompt:**

```
[root@localhost /root]# exit
```


program execution
interpreted programming language
variable and filename substitution
SHELL
environment control
I/O redirection
pipeline hookup

# System Shutdown

## The Need To Shutdow

The Linux operating system keeps the more current versions of the "table of contents", or inode table, in memory to speed disk access.

If the system is not shutdown properly the inode table stored in memory is not written to the disk so the table of contents will not be correct and files will be lost.

Never, under any circumstances, shutdown your Linux system simply by pressing the power button

# The Three Finger Salute - vulcan pinch

<CTRL><ALT><DEL>

**Shutting down in this matter will forcibly log off any other users who will lose whatever their working on**

# The shutdown Command

**#shutdown -h now**

**The shutdown command is the best option for shutting down a system with users currently logged on.**

# halt Command

```
#halt
```

Since they are based on the UNIX operating system, some versions of Linux allow you to use the commands "fasthalt" or "haltsys" to immediately bring the system down in a safe and orderly fashion.

# Rebooting The System

**The reboot Command**

```
#reboot
```

**The "shutdown -r" Command for rebooting the system**

```
#shutdown -r now
```

# WARNING!

**Make certain that you've saved your work before halting or restarting your system from the shell prompt. Running applications will be closed and you won't have the option of saving your work or your session.**

# History of UNIX

Work on UNIX started way back in 1969, when Ken Thompson, Dennis Ritchie and others started working on the “little-used PDP-7 in a corner” at Bell Labs and gradually the product got to be known as UNIX

Ken Thompson

Dennis Ritchie

# Changing your Password

## Exercise to change your password?

1. Type the command **passwd**.
2. You will then be asked for a **new password**.
3. And then asked to **confirm that password.**
4. Then you will arrive **back in the shell.**
5. The password you have chosen will take effect immediately,
6. Replacing the previous password that you used to log in.

# Listing Files (ls)

- Type in the command.

```
[root@localhost /root]#ls
```

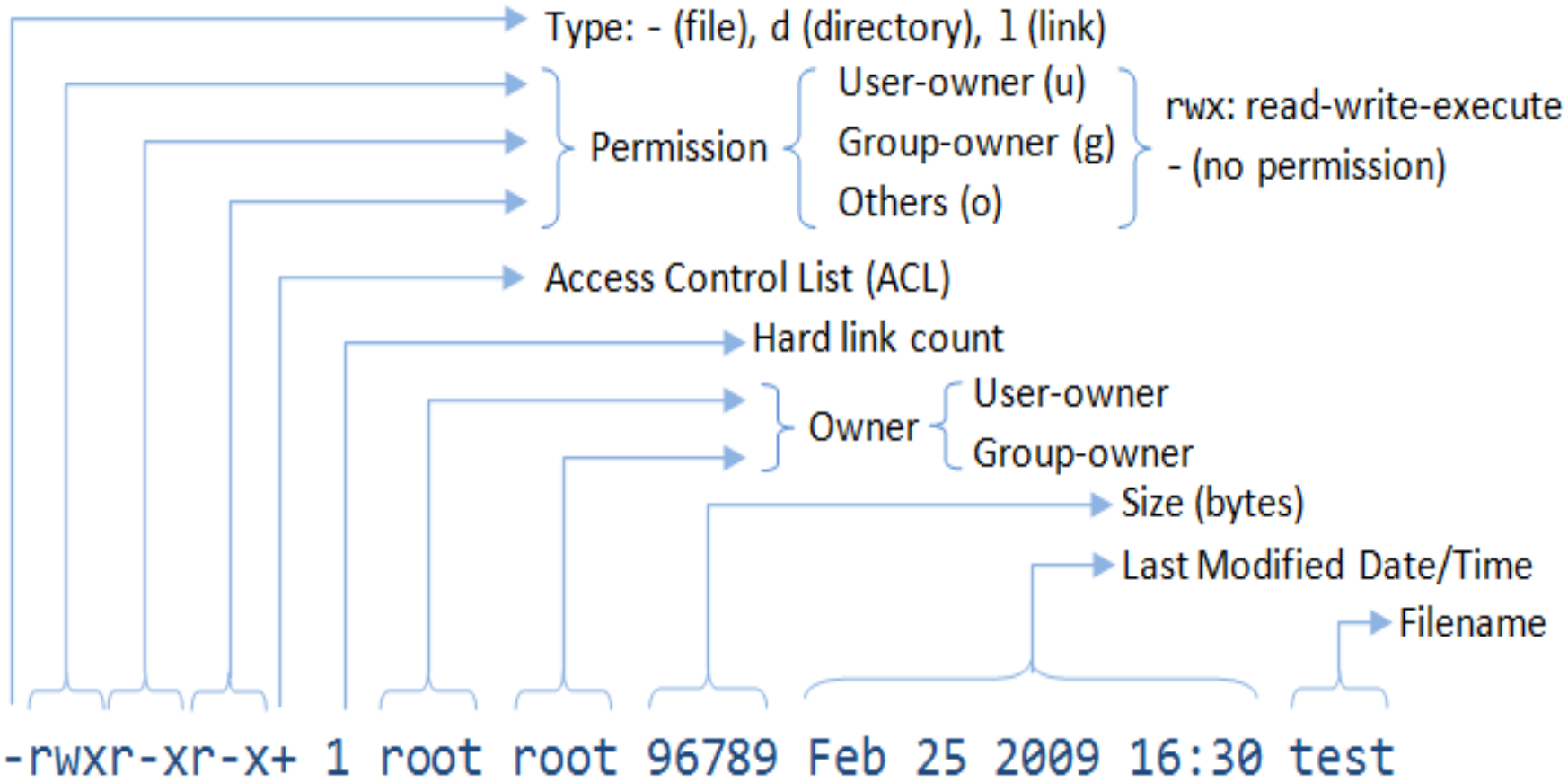
If there were files, you would see their names listed in columns with no indication of what they are for.

- To see a **hidden file** you have to use the command

```
[root@localhost /root]#ls -a
```

- Another variant ls -l which lists the contents in long format.

```
[root@localhost /root]#ls -l
```


Type: - (file), d (directory), l (link)
Permission
User-owner (u)
Group-owner (g)
Others (o)
rwx: read-write-execute
- (no permission)
Access Control List (ACL)
Hard link count
Owner
User-owner
Group-owner
Size (bytes)
Last Modified Date/Time
Filename
-rwxr-xr-x+ 1 root root 96789 Feb 25 2009 16:30 test

# ls (cont.)

They can be strung together in any way that is convenient for example ls -a -l, ls -l -a or ls -al | either of these will list all files in long format.

```
[root@localhost /root]#ls –a -l
```

```
[root@localhost /root]#ls –l -a
```

```
[root@localhost /root]#ls -al
```

# Linux file types

| 1st column | Meaning |
| --- | --- |
| - | Plain text |
| d | Directory |
| c | Character driver |
| b | Block driver |
| l | Link file |
| s | Socket file |
| = or p | FIFO file |

Regular files

Named Pipes

EMERTXE

# System manual pages

You should now use the man command to look up the manual pages for all the commands that you will learn.

Type

```
# man cp
```

```
# man mv
```

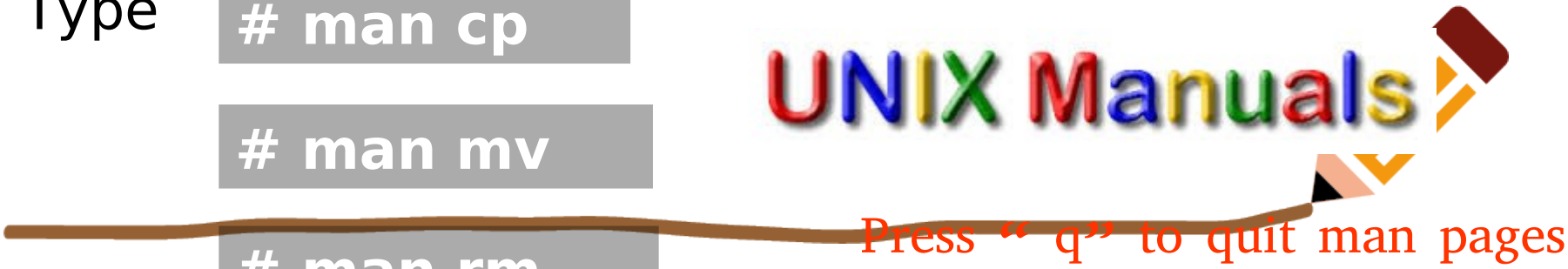
```
# man rm
```

```
# man mkdir
```

```
# man rmdir
```

```
# man passwd
```

```
# man man
```



# System info pages

You can also type info <command> for help on many basic commands.

**Some packages will however not have info pages.**

# Directories [pwd]

The command pwd stands for present working directory (also called the current directory) and is used to tell you what directory you are currently in.

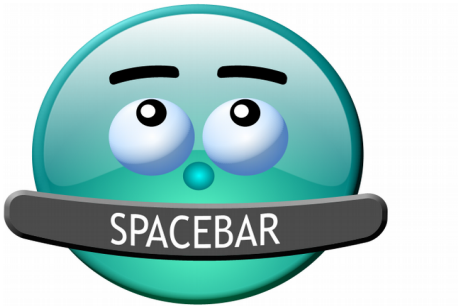
## #pwd

# The "more" Command – A pager

```
# ls -l /bin | more
```

Will show the information one page at a time.

Press “space bar” to go to next page

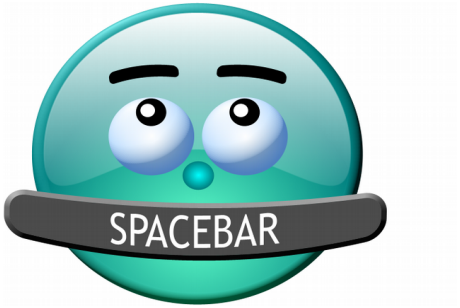


# The "less" Command – A pager

**#ls -l /sbin | less**

Will show the information one page at a time.

Press “space bar” to go to next page



# Make directories [mkdir]

```
#mkdir java
```

```
#mkdir -p java/javaservers/apachi
```

What does -p option mean in above command ?

*Hint: check the man page*

# “touch” command -
## *You wanna create a zero byte file, if it does not exist*

This command updates the timestamps of a file or directory.

If the named file does not exists, it will be created empty.

```
# touch  file or directory
```

Note: 'touch' does not create directories.

# Manipulating directories

## cd — change directories

The cd command is used to take you to different directories.

```
#cd directory1/directory2
```

And similarly you can get back to where you were with

```
#cd ..
```

By simply typing cd you get back to your home directory no matter where ever you are

```
#cd
```

# Directories [rmdir]

**rmdir—Remove empty directories**

```
#rmdir -p dir1/dir2/dir3
```

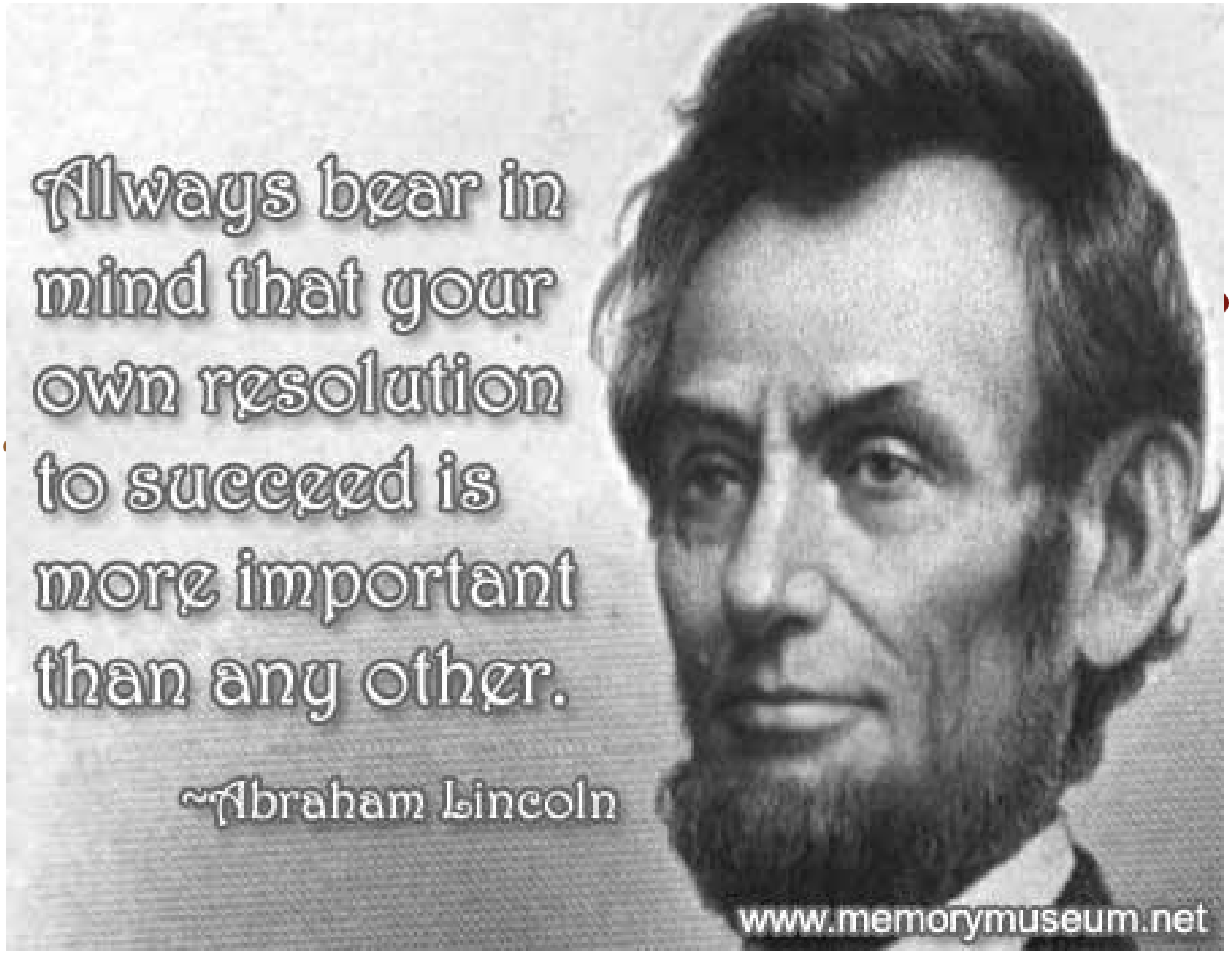
What does -p option mean in above command ?

*Hint: check the man page*

# Directories [rm]

rm—Remove files


**Both commands are dangerous to use as a root user**


Always bear in mind that your own resolution to succeed is more important than any other.
~Abraham Lincoln
www.memorymuseum.net

# Directories [cp] ----- taking photo copy

**SYNOPSIS**

cp [options] source dest

**OPTIONS**

–p, —preserve Preserve the original files' owner, group, permissions, and timestamps.

-r Copy directories recursively, copying all non directories as if they were regular files.

-i Copy files in an interactive way, i.e will ask whether to overwrite the file, if it's already existing in the target/dest location

To copy files, you use the cp command. The following will copy file to file2. Note that if file2 doesn't exist, it'll be created, but if it exists, it'll be overwritten:

```
$ cp file file2
```

There aren't any undo commands in the Linux CLI, so accidentally overwriting an important file would probably make you pull your head off. The risk of doing so is smaller if you use the -i option ("interactive") with cp. The following does the same as the above, but if file2 exists, you'll be prompted before overwriting:

```
$ cp -i file file2

cp: overwrite `file2'? n
```

So it's a good idea to use the -i option whenever you're dealing with important files you don't want to lose!

If you want to copy file into directory dir1:

```
$ cp file dir1
```

The following would do the same as the above, copy file into dir1, but under a different name:

```
$ cp file dir1/file2
```

You can also copy multiple files into one directory with a single command:

```
$ cp file1 file2 file3 dir1
```

Note that if the last argument isn't a directory name, you'll get an error message complaining about it.

# mv or rename command

**The mv command can be used for moving or renaming files. To rename a file, you can use it like this:**

```
$ mv file file2
```

If file2 doesn't exist, it'll be created, but if it exists, it'll be overwritten. If you want to be prompted before overwriting files, you can use the -i option the same way as with cp:

```
$ mv -i file file2
```

To move the file into another directory:

```
$ mv file dir1
```

If you want to rename the file to file2 and move it into another directory, you probably already figured out the command:

```
$ mv file dir1/file2
```

# ln - make links between files

Creating a soft link

```
ln -s   foo   foo-sl
```

Creating a hard link

```
ln   foo   foo-hl
```

Do the listing 'ls -li ' and check for **inodes** of hard linked file ?.

What is your observation?.

# Some useful commands [clear]

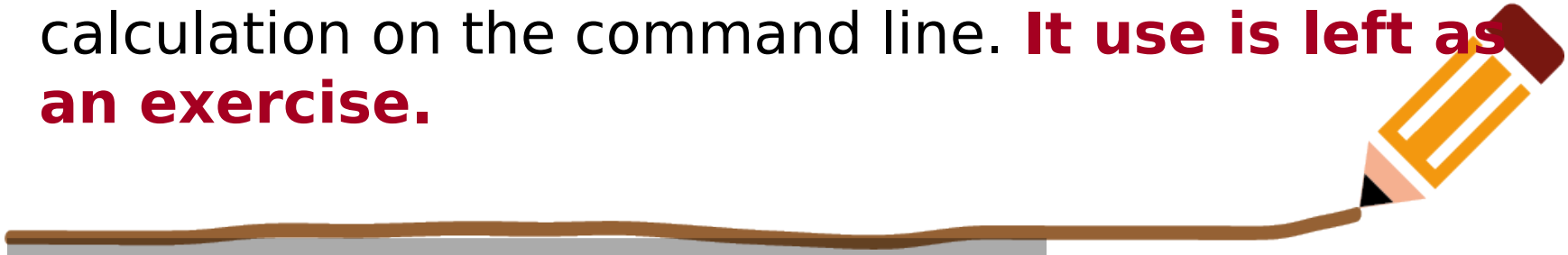
The clear command clears your terminal and returns the command line prompt to the top of the screen.

```
# clear
```

**Note: or press ctrl + l (ell)**

# bc

A calculator program that handles arbitrary precision (very large) numbers. It is useful for doing any kind of calculation on the command line. **It use is left as an exercise.**



```
[root@localhost /root]# bc
```

**Ctrl + d to exit**

# cal [[0-12] 1--9999]

Prints out a nicely formatted calendar of the current month, or a specified month, or a specified whole year.

```
# cal  1947

# cal 9 1752
```

# whoami

Prints out your login name.

```
# whoami
# who am i
```

**What's the difference ?**

# date --- wanna date ?

Prints out the current date and time.

```
[root@localhost /root]#date
```

# df Stands for disk free

This tells you how much free space is left on your system.

**[root@localhost /root]# df -h**

**[root@localhost /root]# df  -Th**

```
ilg@Insight ~/rnd $ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/sda1      ext4      9.1G  366M  8.3G   5% /
none           tmpfs     4.0K     0  4.0K   0% /sys/fs/cgroup
udev           devtmpfs  2.9G  4.0K  2.9G   1% /dev
tmpfs          tmpfs     588M  1.4M  586M   1% /run
none           tmpfs     5.0M     0  5.0M   0% /run/lock
none           tmpfs     2.9G  1.5M  2.9G   1% /run/shm
none           tmpfs     100M   16K  100M   1% /run/user
/dev/sda10     ext4      3.7G  9.9M  3.4G   1% /tmp
/dev/sda7      ext4       46G  925M   43G   3% /var
/dev/sda11     ext4      363G  326G   18G  95% /home
/dev/sda5      ext4      922M   43M  816M   5% /boot
/dev/sda8      ext4      9.1G  5.9G  2.7G  69% /opt
/dev/sda9      ext4       23G  4.6G   18G  21% /usr
ilg@Insight ~/rnd $
```

# free Prints out available free memory.

You will notice two listings: swap space and physical memory.

## # free

Check man page for various options like

-m

-k

What is the option for Tera Byte ?

# uname

Prints out the name of the Unix operating system you are currently using.

```
# uname -a
```

-r  option ?
-n option ?

# wc - I want to count words, lines, char

**wc [-c] [-w] [-l] <filename>**

Counts the number -

characters/bytes (with -c),

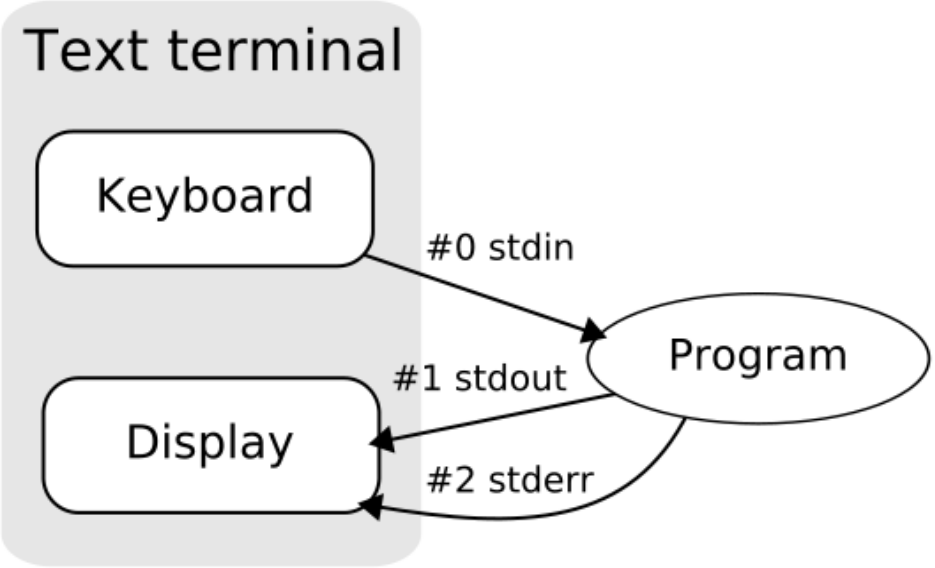
words (with -w) or

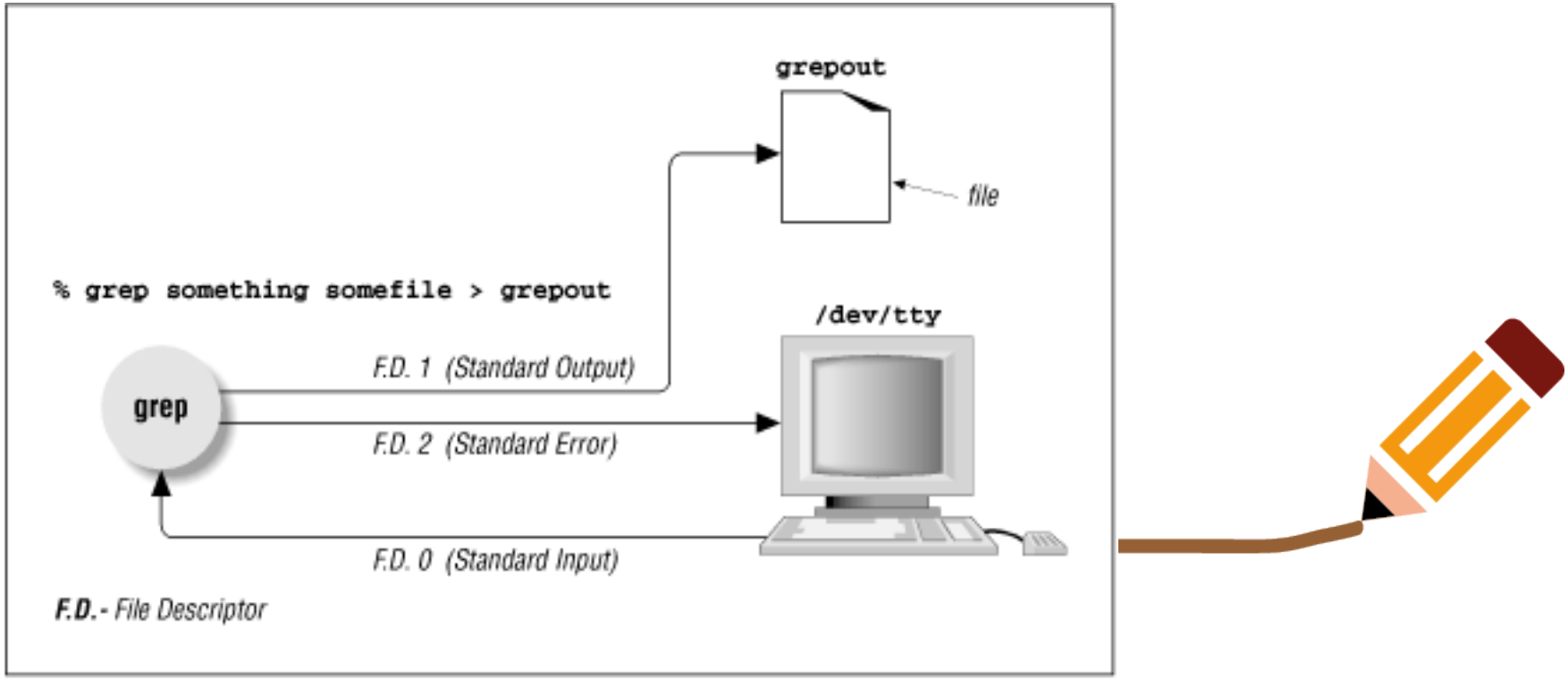
lines (with-l) in a file.

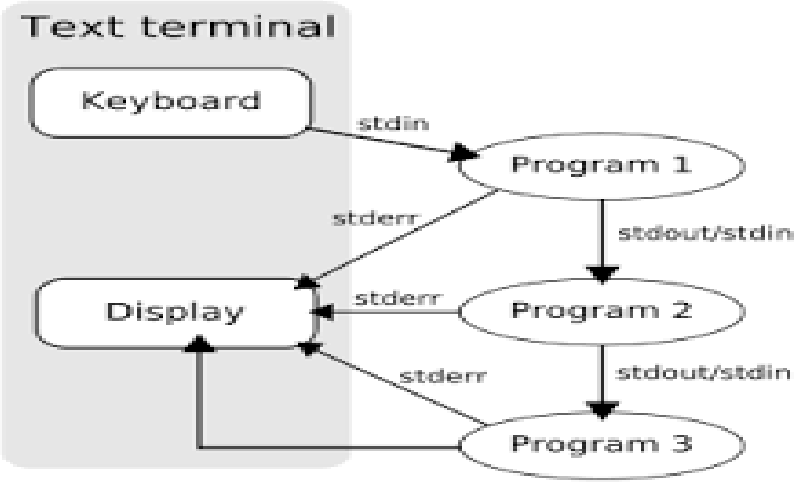
```
# wc  /etc/passwd
```

# Stdin,stdout,stderr

- In computer programming, standard streams are pre connected input and output channels between a computer program and its environment (typically a text terminal) when it begins execution.
- The three I/O connections are called
  - standard input (stdin) – 0 (zero)
  - standard output (stdout) – 1
  - standard error (stderr) - 2

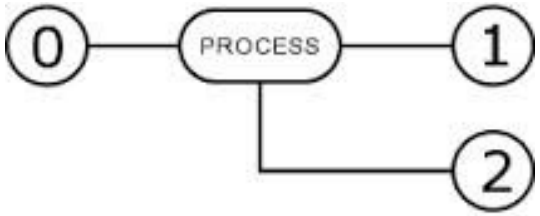



grepout
file
% grep something somefile > grepout
/dev/tty
grep
F.D. 1 (Standard Output)
F.D. 2 (Standard Error)
F.D. 0 (Standard Input)
F.D.- File Descriptor

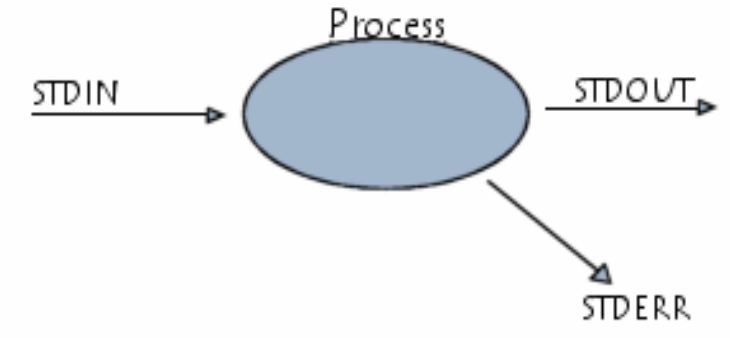

Text terminal
Keyboard
stdin
Program 1
stderr
stdout/stdin
Display
stderr
Program 2
stderr
stdout/stdin
Program 3

## Standard input (stdin 0)

**$ mail info@gnugroup.org < /etc/hosts**



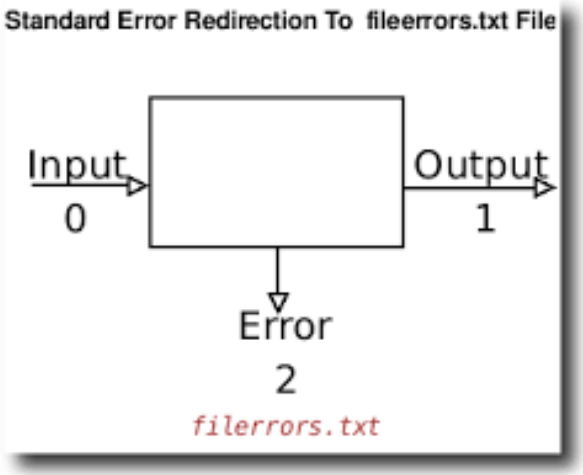

## Standard output (stdout 1 )




**$ ls -l /tmp /tmnp > out.txt**

## Standard error (stderr 2)

**$ ls -l /tmp tmnp 2> err.txt**




Wow !!!
GOOD achievement

# Using cat command to create files

Start cat to see what this means. At the shell prompt, type:

```
# cat
```

**The cursor moves to a blank line. Now, in that blank line, let's type:**

```
stop by sneaker store
```

**and press the [Enter] key. Your screen will look like:**

```
# cat
stop by sneaker store
stop by sneaker store
```

**To quit cat now, press the [Ctrl] and [D] keys at the same time.**

# Cat
# Standard Input & Standard Output

But cat has just demonstrated the definition of standard input and standard output.

Your input was read from the keyboard (standard input), and that input was then directed to your terminal (standard output).

# Using Redirection

**Redirection** means causing the shell to change what it considers standard input or where the standard output is going.

To redirect standard output, we'll use the **>** symbol. Placing **>** after the cat command

# Let’s try Redirection.

```
# cat >sneakers.txt
buy some sneakers
then go to the coffee shop
```

Now press **[Enter]** to go to an empty line, and use the **[Ctrl]-[D]** keys to quit **cat.**

You can even use cat to read the file, by typing at the prompt.

```
#cat sneakers.txt
```

# Caution

You can easily overwrite an existing file! Make sure the name of the file you're creating doesn't match the name of a pre-existing file, unless you want to replace it.

# Exercise

Create another file named **home.txt** having the following contents

**bring the coffee home**

**take off shoes**

**put on sneakers**

**make some coffee**

**relax!**

**Check the file using cat command ?**

# Joining Files and Redirecting Output

```
[user@localhost /user]# cat sneakers.txt
home.txt > myfile
```

Now it’s time to check our handiwork. Type:

```
[newuser@localhost /newuser]# cat myfile
```

# Appending Standard Output

when you use **>>**, you're *adding* information, rather than replacing it.

Type

```
#cat home.txt >> sneakers.txt
```

Now let's check the file by typing:

```
#cat sneakers.txt
```

# Redirecting Standard Input

**Just type:**

```
#cat < sneakers.txt
```

# Using Output Redirection with Other commands

**Type**

```
$ date > date.dat
$ cat date.dat
$ ls > list.dat
$ cat list.dat
```

Now combine these two files in file name combo

# The tee Utility

You can use the **tee** utility in a pipe to send the output of a command to a file while also sending the output to standard output.

The utility takes a single input and sends the output in two directions.

```
$ ls  -l | tee who.out
```


"Talk
To Yourself
Once In A Day...
Otherwise
You May
Miss Meeting
An
Excellent Person
in this World"

# which command

To locate the exact path of a program, you can use the **which** command

Type

```
# which hostname

/bin/hostname
```

# head

Syntax:
head [*-count* | *-n number*] *filename*

This command will display the first few lines of a file.

By default, the first 10 lines of a file are displayed.

However, you could use the preceding options to specify a different number of lines.

```
# head -2 doc.txt
```

# tail – display last 10 lines of file

```
# tail -n -50 doc.txt
# tail  doc.txt
```

PUSH YOURSELF BECAUSE, NO ONE ELSE IS GOING TO DO IT FOR YOU.

# locate

***locate <filename>***.

This searches through a previously created database of all the files on the system, and hence finds files instantaneously.

Its counterpart

***# updatedb***

is used to update the database of files used by locate.

On some systems updatedb runs automatically every day at 04h00.

# Grep – global regular expn print

**[root@localhost /root]#** grep [-viw] *pattern file(s)*

The grep command allows you to search for one or more files for particular character patterns.

Every line of each file that contains the pattern is displayed at the terminal.

The grep command is useful when you have lots of files and you want to find out which ones contain words or phrases.

# grep

Using the **-V** option, we can display the inverse of a pattern. Perhaps we want to select the lines in data.txt that *do not* contain the word "the":

If the **-W** option was not specified, then any word containing "the" would match, like "toge[the]r." The -w option specifies that the pattern must be a whole word.

```
# grep -vw 'the' data.txt
```

And finally, the **-i** option ignores the difference between upper and lowercase letters when searching for the pattern.

# Searching for files using find command

Change to the root directory, and enter find.

find will work for a long time if you enter it as you have

press Ctrl-C to stop it.

Now change back to your home directory and type find again.

You will see all your personal files.

# Searching for files using find command

There are a number of options find can take to look for specific files.

find -type d will show only directories and not the files they contain.

find -type f will show only files and not the directories that contain them, even though it will still descend into all directories.

# find (cont....)

find -name <filename> will find only files that have the name <filename>.

For instance, find -name '*.c‘ Will find all files that end in a .c extension

**without the quote characters will not work.**

find -name Mary Jones.letter will find the file with the name

Mary Jones.letter.

find -size [[+|-]]<size> will find only files that have a size larger (for +) or smaller (for -) than <size> kilobytes, or the same as <size> kilobytes if the sign is not specified.

# Try this

(search for shutdown)

find / -name shutdown

(remove file during search)

find / -name core –type f –ok rm {} \;

(copy file during search)

find / -name passwd –type f –ok cp {} /root \;

(find all files that have been accessed during past 24 hrs)

find . –name “*.gif” –atime –1 –exec ls –l {} \;

(This displays all empty files in the current directory)

find . –type f –empty

# Unix command-line interface programs
# &
# shell builtins

| | |
|---|---|
| **File system** | cat • cd • chmod • chown • chgrp • cksum • cmp • cp • dd • du • df • file • fsck • fuser • ln • ls • mkdir • mount • mv • pax • pwd • rm • rmdir • size • split • tee • touch • type • umask |
| **Processes** | at • bg • chroot • cron • fg • kill • killall • nice • pgrep • pkill • ps • pstree • time • top |
| **User environment** | clear • env • exit • finger • history • id • logname • mesg • passwd • su • sudo • uptime • talk • tput • uname • w • wall • who • whoami • write |
| **Text processing** | awk • banner • basename • comm • csplit • cut • diff • dirname • ed • ex • fmt • fold • head • iconv • join • less • more • nl • paste • sed • sort • spell • strings • tail • tr • uniq • vi • wc • xargs |
| **Shell builtins** | alias • echo • expr • printf • sleep • test • true and false • unset • wait • yes |
| **Networking** | dig • host • ifconfig • inetd • netcat • netstat • nslookup • ping • rdate • rlogin • route • ssh • traceroute |
| **Searching** | find • grep • locate • whatis • whereis |
| **Documentation** | apropos • help • man |
| **Miscellaneous** | bc • dc • cal • lp • od |


QUESTIONS?
COMMENTS?
CONCERNS?