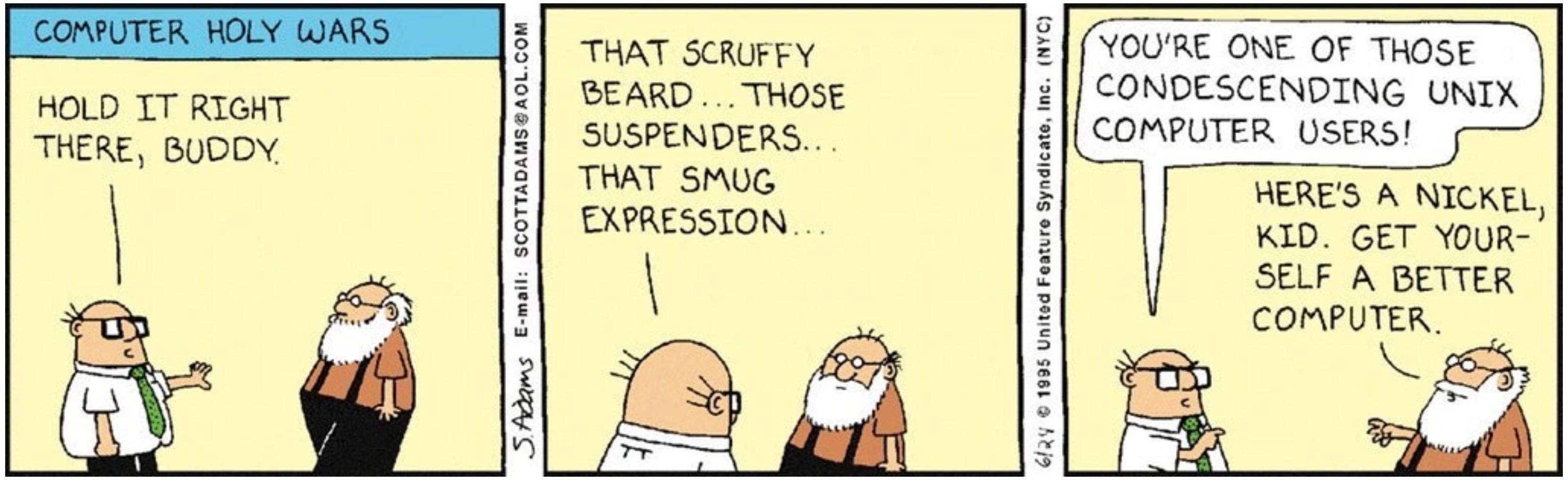
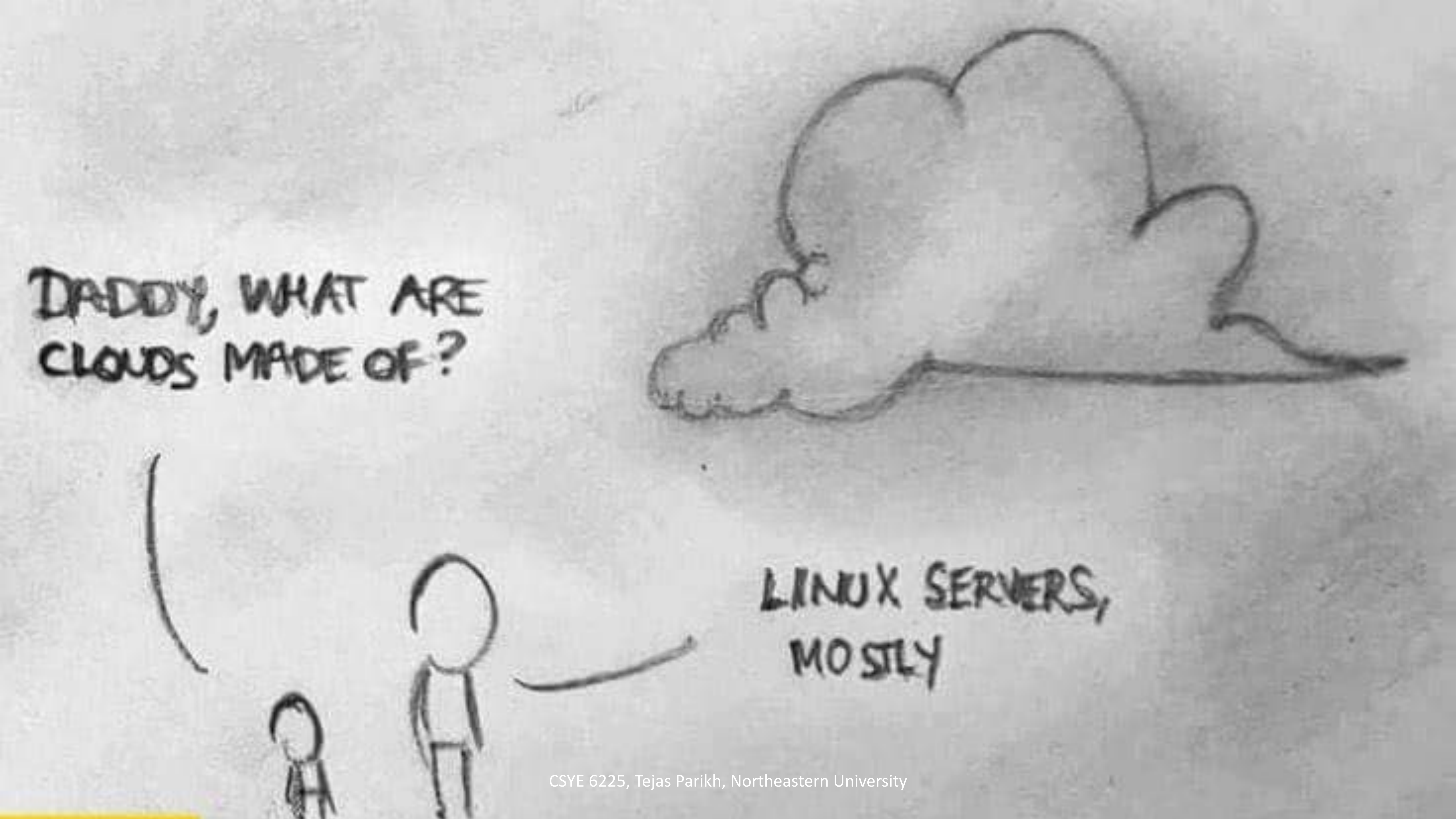
Introduction to Linux Command Line

Tejas Parikh (t.parikh@northeastern.edu)

CSYE 6225

Northeastern University


COMPUTER HOLY WARS
HOLD IT RIGHT THERE, BUDDY.
THAT SCRUFFY BEARD... THOSE SUSPENDERS... THAT SMUG EXPRESSION...
YOU'RE ONE OF THOSE CONDESCENDING UNIX COMPUTER USERS!
HERE'S A NICKEL, KID. GET YOURSELF A BETTER COMPUTER.
S. Adams E-mail: SCOTTADAMS@AOL.COM
6/24 © 1995 United Feature Syndicate, Inc. (NYC)


DADDY, WHAT ARE CLOUDS MADE OF?
LINUX SERVERS, MOSTLY

What is Linux?

- The term “Linux” is often used to refer to the entire operating system, but in reality, Linux is the operating system kernel, which is started by the boot loader, which is itself started by the BIOS/UEFI.
- The kernel assumes a role like that of a conductor in an orchestra—it ensures coordination between hardware and software.
- This role includes managing hardware, processes, users, permissions, and the file system.
- The kernel provides a common base to all other programs on the system and typically runs in *ring zero*, also known as **kernel space**.

Components of Operating Systems

- **The Bootloader:** The software that manages the boot process of your computer.
- **The kernel:** This is the one piece of the whole that is actually called “Linux”. The kernel is the core of the system and manages the CPU, memory, and peripheral devices. The kernel is the “lowest” level of the OS.
- **Daemons:** These are background services (printing, sound, scheduling, etc) that either start up during boot, or after you log into the desktop.
- **Desktop Environment:** This is the piece of the puzzle that the users interact with.
- **Applications:** Desktop environments do not offer the full array of apps.
- **The Shell:** A command process that allows you to control the computer via commands typed into a text interface.

The Command Line (Linux Shell)

- The **shell** acts as an interface between the user and the kernel.
- When a user logs in, the login program checks the username and password, and then starts another program called the shell.
- The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out.
- The commands are themselves programs: when they terminate, the shell gives the user another prompt.

Popular Linux Shells

- sh - Bourne Shell
- ksh - Korn Shell
- csh / tcsh - C Shell
- bash - Bourne-Again Shell
- zsh - Z shell

Switching Between Linux Shells

```
➜  ~ 
```

Bash Shell Special Characters

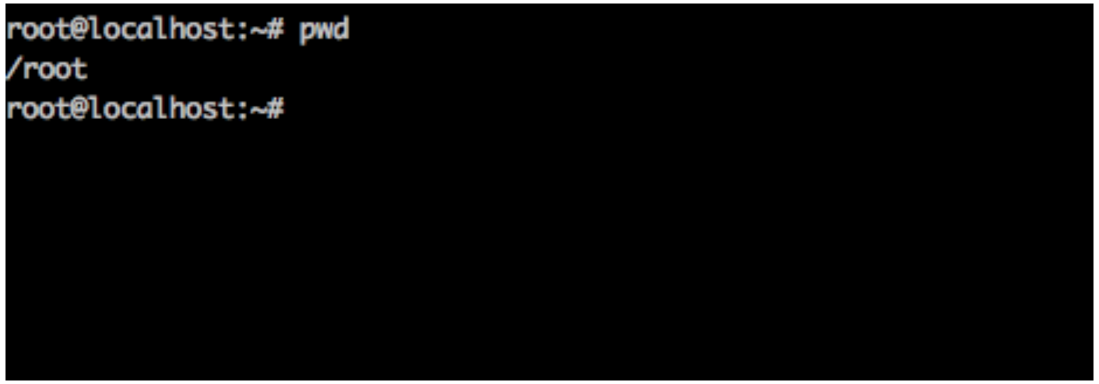
Special Character	Meaning
~	current user's home directory, same as $HOME (e.g. /home/user)
$	used to access a variable (e.g. $HOME)
&	used to put a command in the background
*	wildcard, matching zero or more character

Navigation

pwd, cd, ls

pwd – Print Working Directory

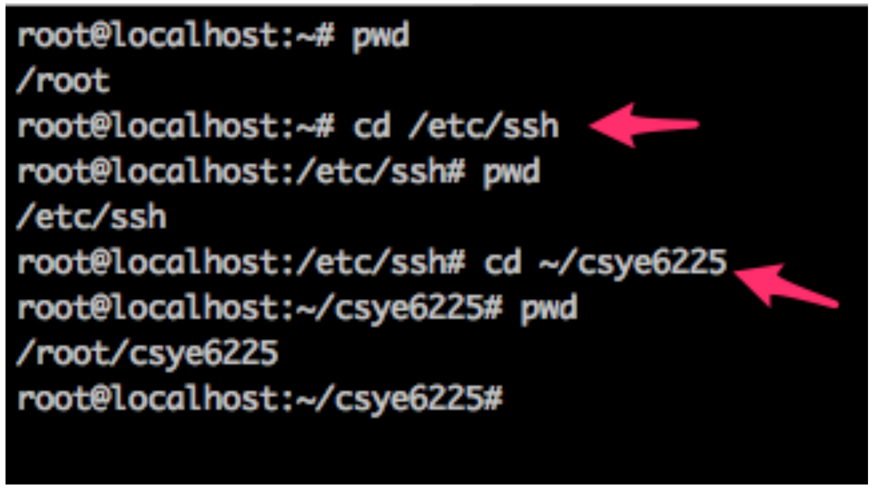
- At any given time, we are inside a single directory, and we can see the files contained in the directory and the pathway to the directory above us (called the parent directory) and any subdirectories below us.
- The directory we are standing in is called the current working directory.
- To display the current working directory, we use the **pwd** (print working directory) command.



```
root@localhost:~# pwd
/root
root@localhost:~#
```

Changing the Current Working Directory

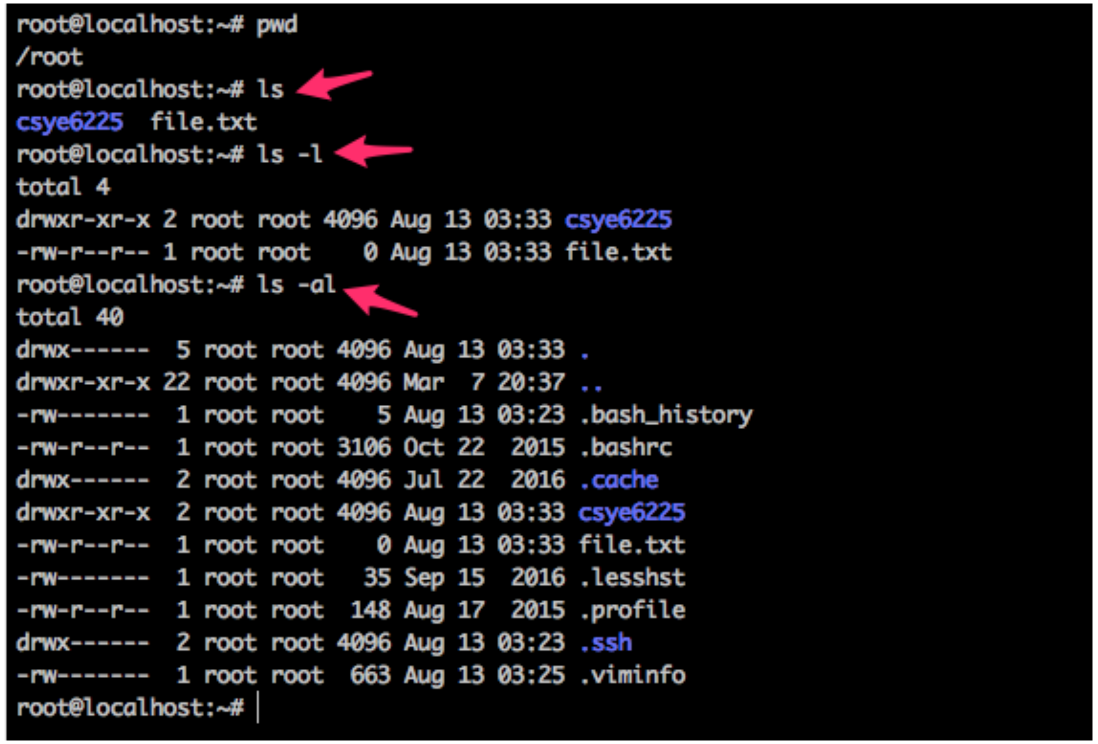
- To change your working directory, we use the **cd** command.
- To do this, type **cd** followed by the pathname of the desired working directory.
- Pathnames can be specified in one of two different ways; as ***absolute pathnames*** or as ***relative pathnames***.



```
root@localhost:~# pwd
/root
root@localhost:~# cd /etc/ssh
root@localhost:/etc/ssh# pwd
/etc/ssh
root@localhost:/etc/ssh# cd ~/csye6225
root@localhost:~/csye6225# pwd
/root/csye6225
root@localhost:~/csye6225#
```

ls – Listing the Contents Of A Directory

To list the files and directories in the current working directory, we use the **ls** command.



```
root@localhost:~# pwd
/root
root@localhost:~# ls
csye6225  file.txt
root@localhost:~# ls -l
total 4
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
-rw-r--r-- 1 root root    0 Aug 13 03:33 file.txt
root@localhost:~# ls -al
total 40
drwx------  5 root root 4096 Aug 13 03:33 .
drwxr-xr-x 22 root root 4096 Mar  7 20:37 ..
-rw-------  1 root root    5 Aug 13 03:23 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  2 root root 4096 Jul 22  2016 .cache
drwxr-xr-x  2 root root 4096 Aug 13 03:33 csye6225
-rw-r--r--  1 root root    0 Aug 13 03:33 file.txt
-rw-------  1 root root   35 Sep 15  2016 .lesshst
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
drwx------  2 root root 4096 Aug 13 03:23 .ssh
-rw-------  1 root root  663 Aug 13 03:25 .viminfo
root@localhost:~# 
```

Important Facts About Filenames

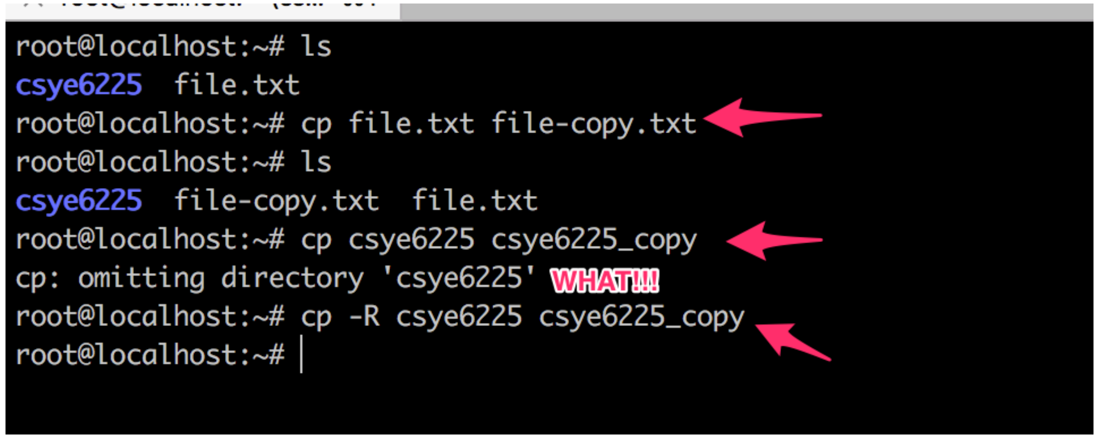
- Filenames that begin with period character are hidden.
- Filenames and commands in Linux are case sensitive.
- Linux has no concept of a “file extension”.

Manipulating Files & Directories

cp, mv, mkdir, rm, ln

cp – Copy files and directories

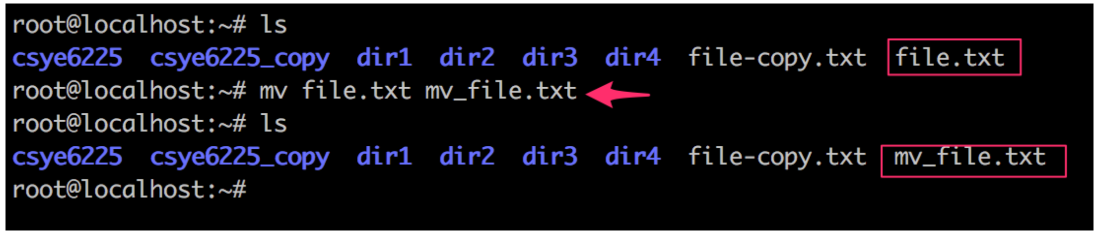
The **cp** command copies files or directories.



```
root@localhost:~# ls
csye6225  file.txt
root@localhost:~# cp file.txt file-copy.txt
root@localhost:~# ls
csye6225  file-copy.txt  file.txt
root@localhost:~# cp csye6225 csye6225_copy
cp: omitting directory 'csye6225' WHAT!!!
root@localhost:~# cp -R csye6225 csye6225_copy
root@localhost:~# 
```

mv – Move/rename files and directories

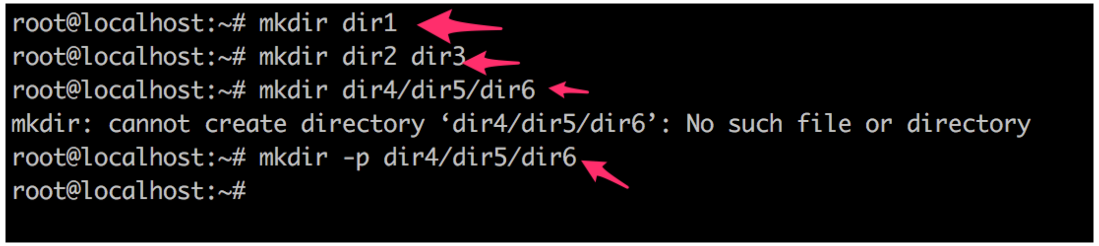
- The **mv** command performs both file *moving* and file *renaming*, depending on how it is used.
- In either case, the original filename no longer exists after the operation.
- **mv** is used in much the same way as **cp**.



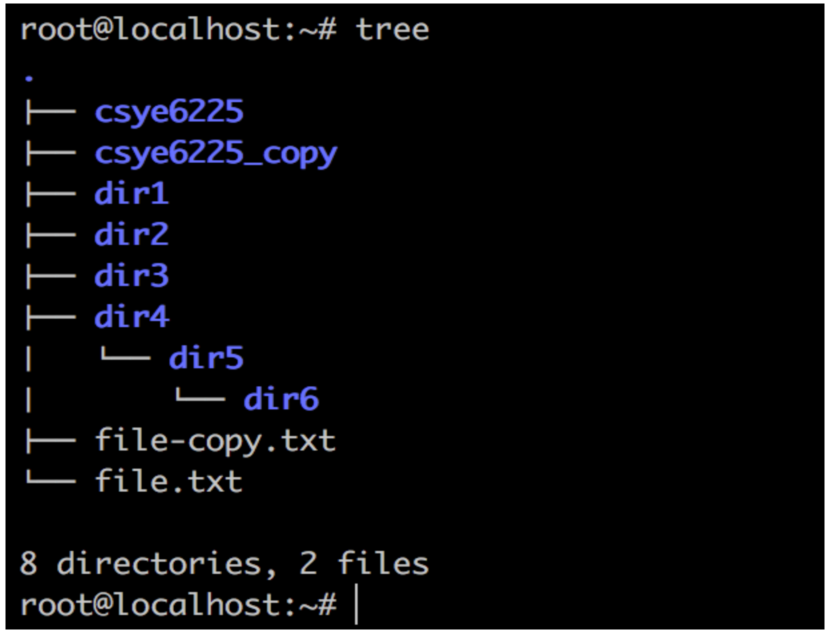
```
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  dir4  file-copy.txt  file.txt
root@localhost:~# mv file.txt mv_file.txt
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  dir4  file-copy.txt  mv_file.txt
root@localhost:~#
```

mkdir – Create directories

- The **mkdir** command is used to create directories.



```
root@localhost:~# mkdir dir1
root@localhost:~# mkdir dir2 dir3
root@localhost:~# mkdir dir4/dir5/dir6
mkdir: cannot create directory ‘dir4/dir5/dir6’: No such file or directory
root@localhost:~# mkdir -p dir4/dir5/dir6
root@localhost:~#
```

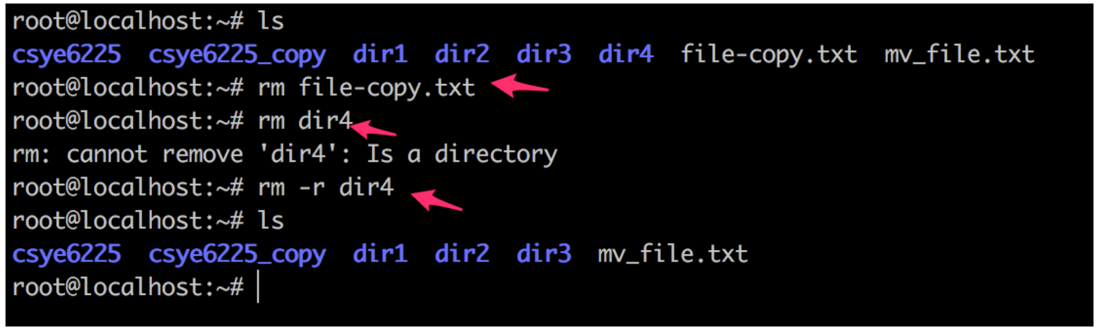


```
root@localhost:~# tree
.
├── csye6225
├── csye6225_copy
├── dir1
├── dir2
├── dir3
├── dir4
|   └── dir5
|       └── dir6
├── file-copy.txt
└── file.txt

8 directories, 2 files
root@localhost:~#
```

rm – remove files and directories

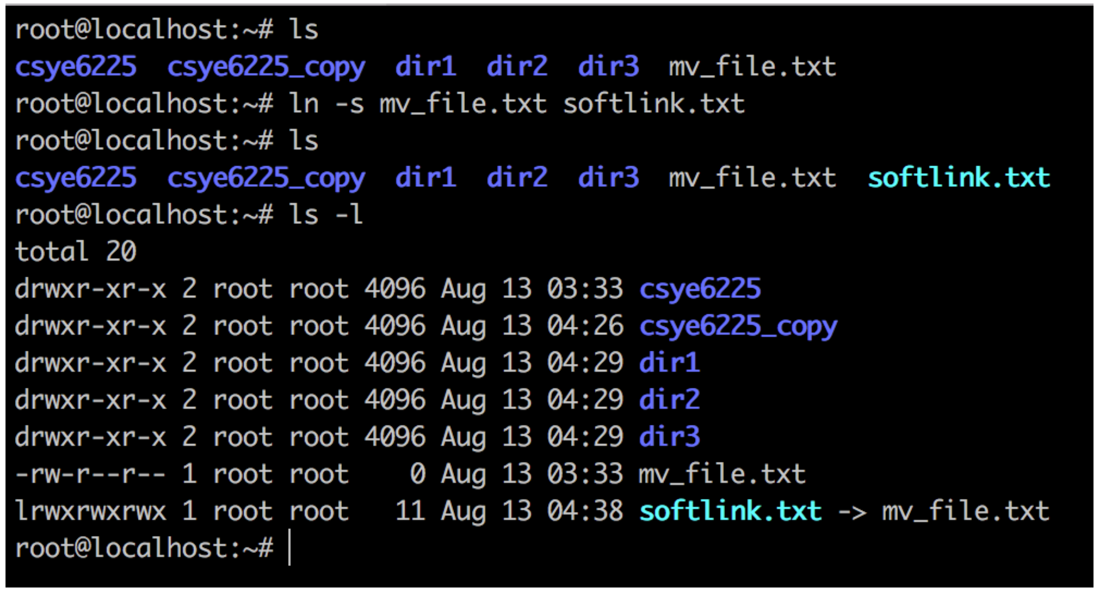
- The **rm** command is used to remove (delete) files and directories



```
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  dir4  file-copy.txt  mv_file.txt
root@localhost:~# rm file-copy.txt
root@localhost:~# rm dir4
rm: cannot remove 'dir4': Is a directory
root@localhost:~# rm -r dir4
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  mv_file.txt
root@localhost:~# 
```

ln – Create hard or soft symbolic links

- The **ln** command is used to create either hard or symbolic links.
- **Symbolic links** work by creating a special type of file that contains a text pointer to the referenced file or directory.
  - When you delete a symbolic link, only the link is deleted, not the file itself.
- A **hard link** is indistinguishable from the file itself.
  - Unlike a symbolic link, when you list a directory containing a hard link you will see no special indication of the link.



```
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  mv_file.txt
root@localhost:~# ln -s mv_file.txt softlink.txt
root@localhost:~# ls
csye6225  csye6225_copy  dir1  dir2  dir3  mv_file.txt  softlink.txt
root@localhost:~# ls -l
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# 
```

I/O Redirection

Commands: cat, grep, head, tail

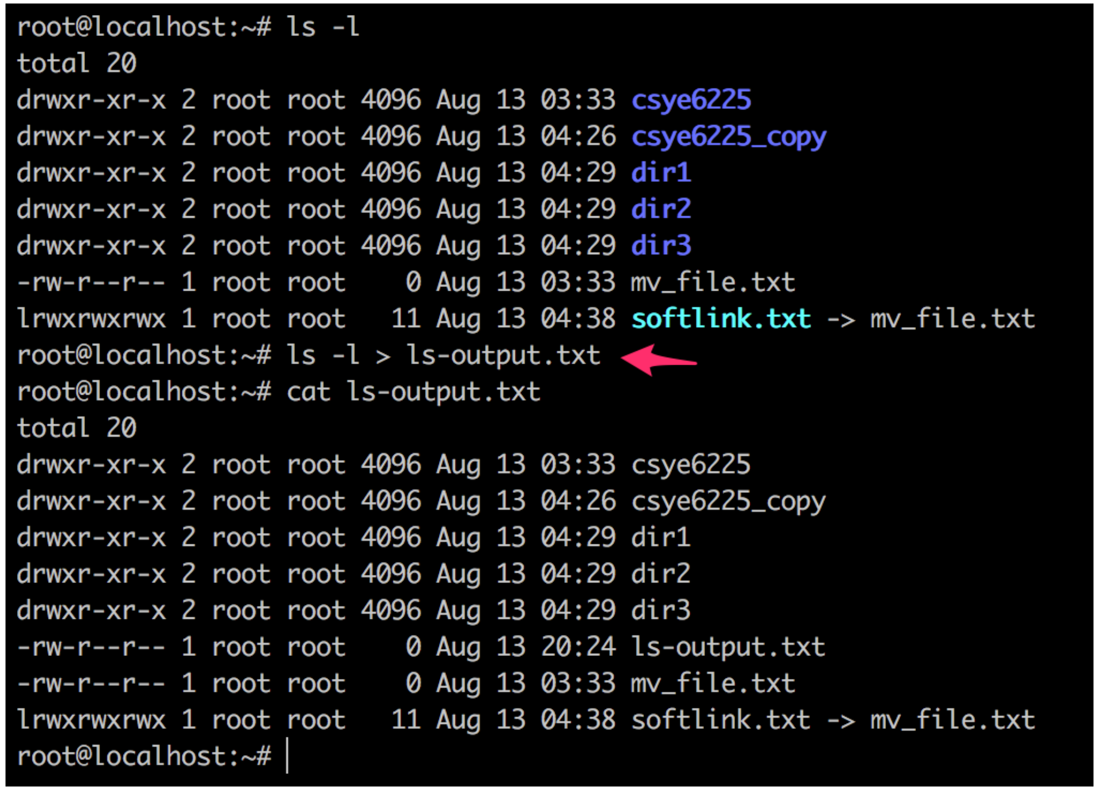
& Pipes

Output Streams

- Standard Output
  - **System.out**.println(“this message goes to standard output”);
- Standard Error
  - **System.err**.println(“this message goes to standard error”);

Redirecting | Standard Output

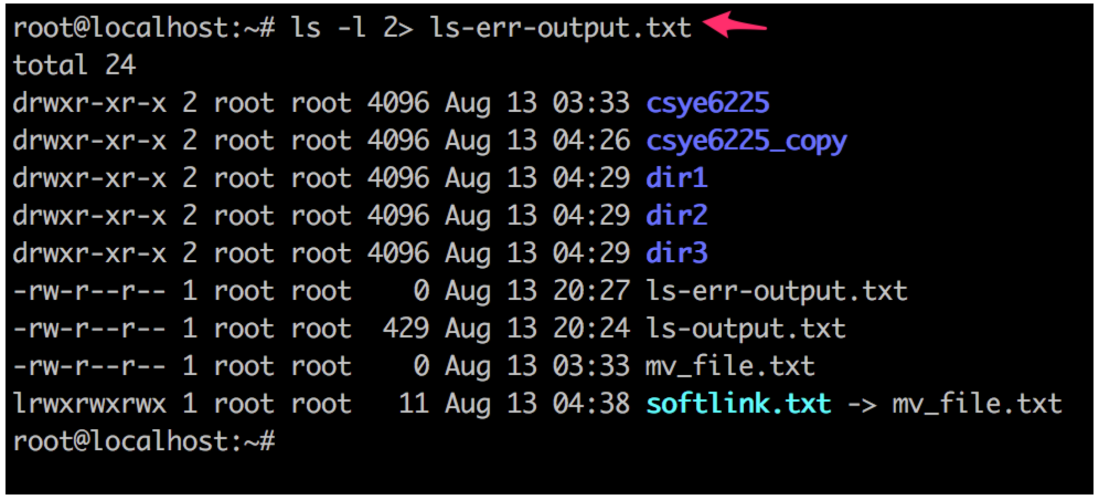
- I/O redirection allows us to redefine where standard output goes.
- To redirect standard output to another file instead of the screen, we use the **“>”** redirection operator followed by the name of the file.



```
root@localhost:~# ls -l
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# ls -l > ls-output.txt
root@localhost:~# cat ls-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

Redirecting | Standard Error

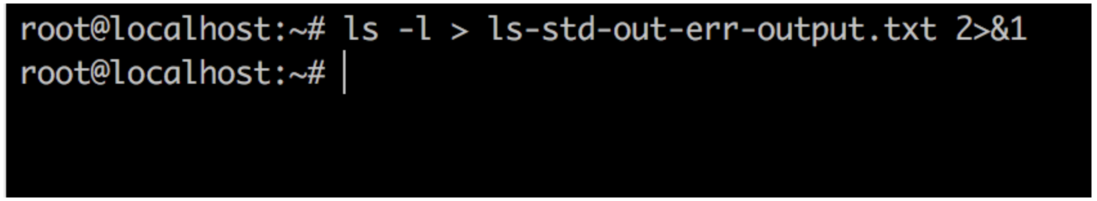
- Redirecting standard error lacks the ease of a dedicated redirection operator.
- To redirect standard error, we must refer to its *file descriptor*.



```
root@localhost:~# ls -l 2> ls-err-output.txt
total 24
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:27 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

Redirecting Standard Output And Standard Error To One File

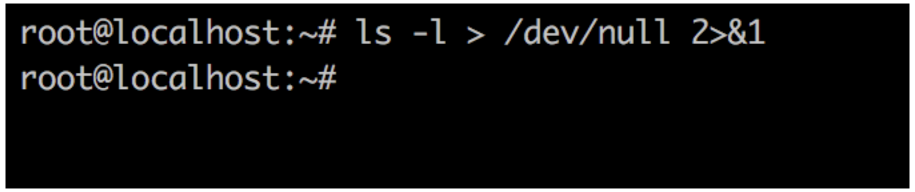
- There are cases in which we may wish to capture all the output of a command to a single file.



```
root@localhost:~# ls -l > ls-std-out-err-output.txt 2>&1
root@localhost:~#
```

/dev/null

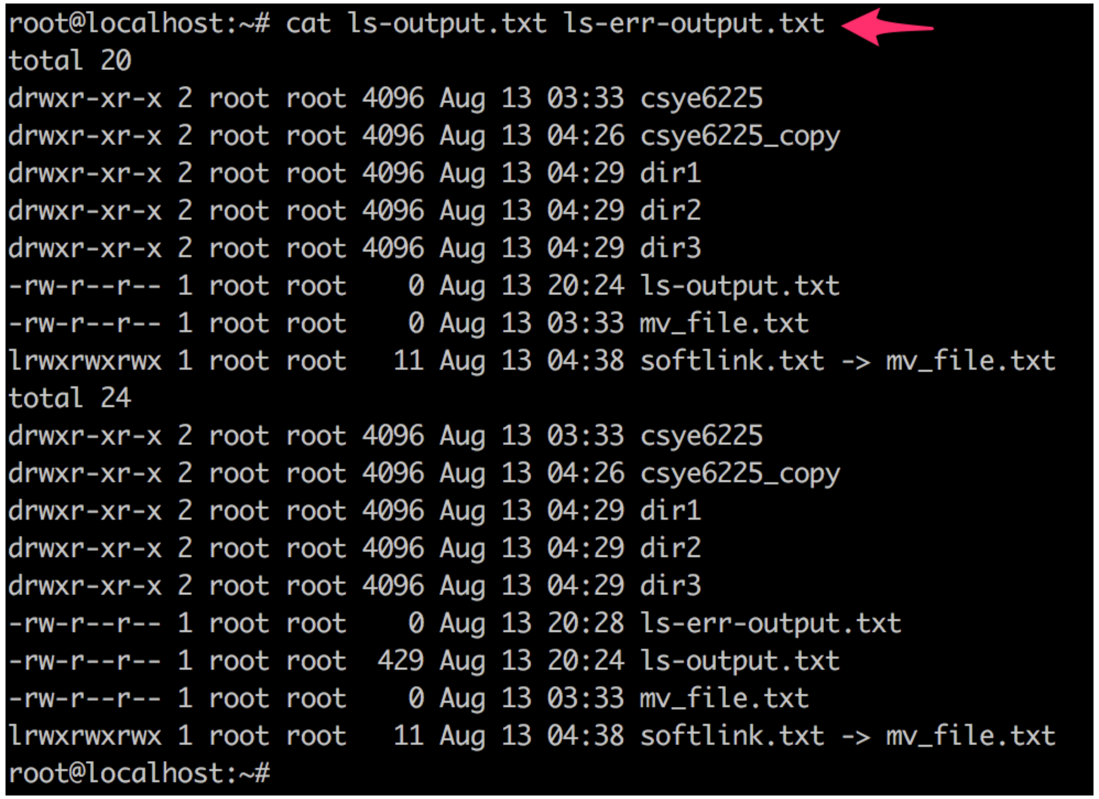
- Used for disposing of unwanted output.
- It is sometimes referred to as "black hole".



```
root@localhost:~# ls -l > /dev/null 2>&1
root@localhost:~#
```

cat - Concatenate files

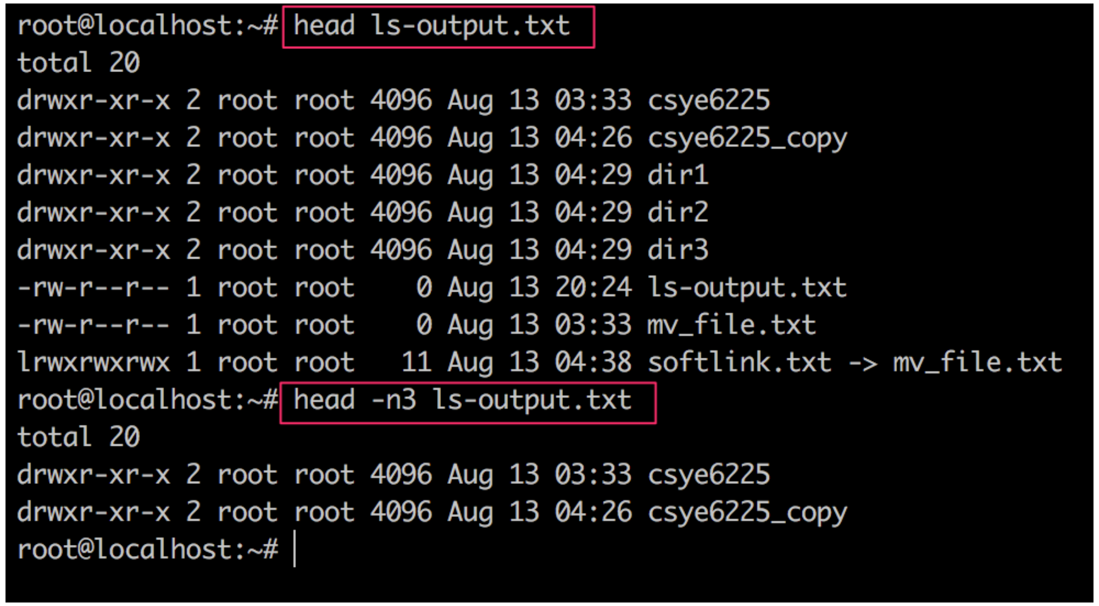
- The **cat** command reads one or more files and copies them to standard output.



```
root@localhost:~# cat ls-output.txt ls-err-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
total 24
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

head – Output the first part of a file

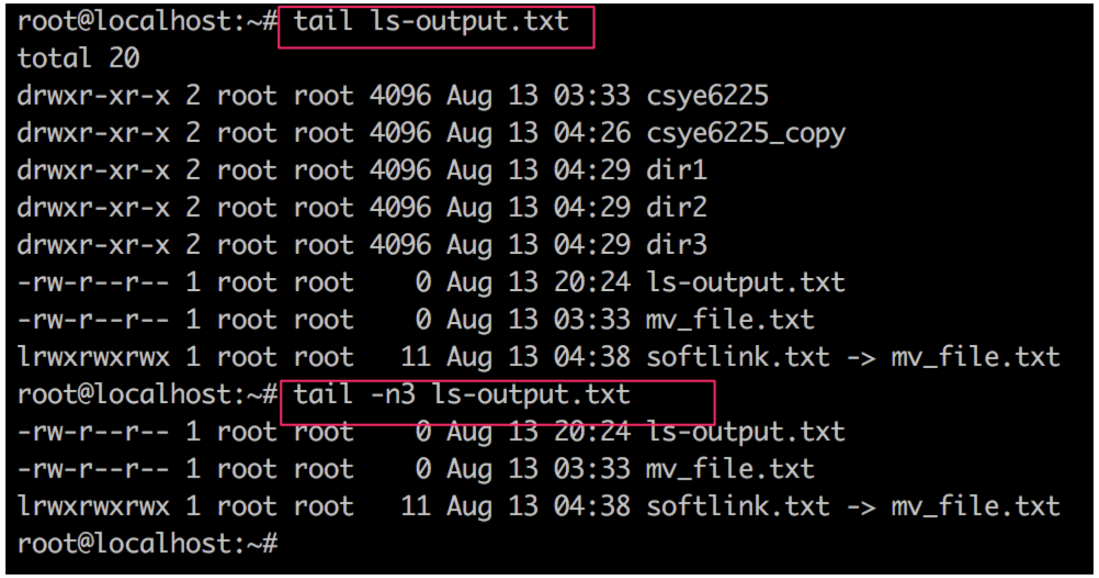
- Sometimes you don't want all the output from a command. You may only want the first few lines.
- The head command prints the first ten lines of a file .



```
root@localhost:~# head ls-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# head -n3 ls-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
root@localhost:~#
```

tail – Output the last part of a file

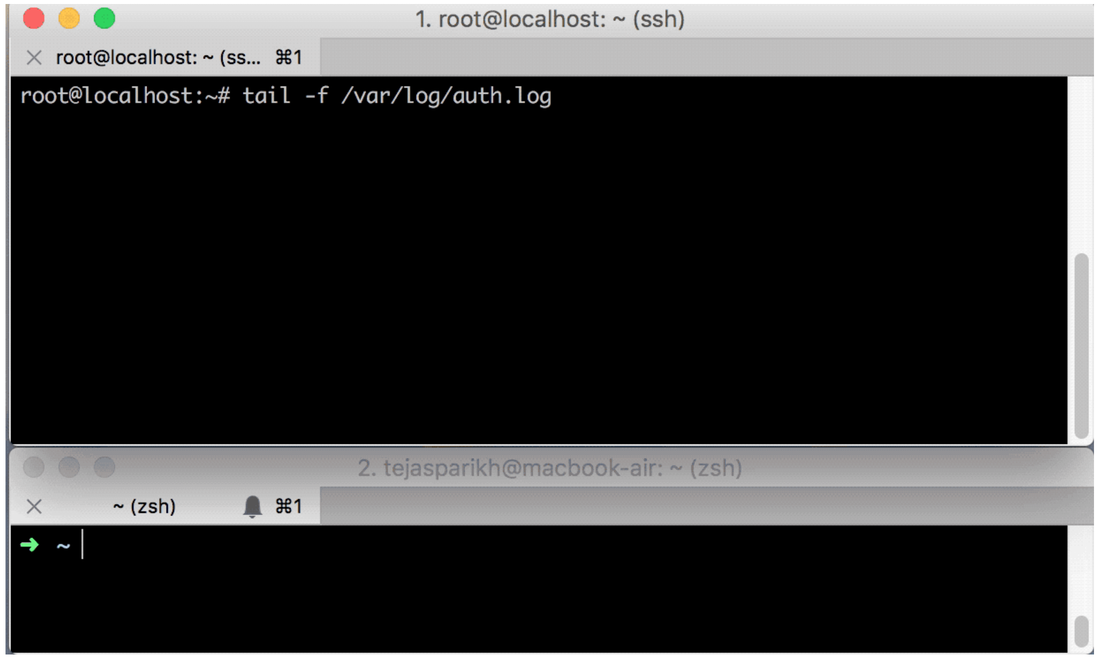
- Sometimes you don't want all the output from a command. You may only want the last few lines.
- The **tail** command prints the last ten lines of a file.



```
root@localhost:~# tail ls-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# tail -n3 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

tail – Follow files in real-time

- **tail** has an option which allows you to view files in real-time

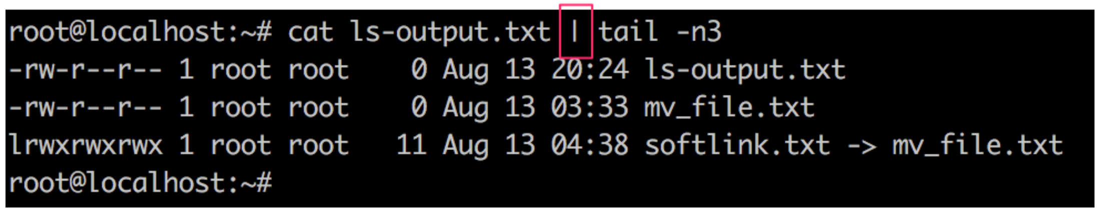

1. root@localhost: ~ (ssh)
root@localhost: ~ (ss... ⌘1
2. tejasparikh@macbook-air: ~ (zsh)
~ (zsh) ⌘1

```
root@localhost:~# tail -f /var/log/auth.log
```

```
➜ ~ |
```

Pipelines

- The ability of commands to read data from standard input and send to standard output is utilized by a shell feature called **pipelines**.
- Using the pipe operator "**|**" (vertical bar), the standard output of one command can be **piped** into the standard input of another.



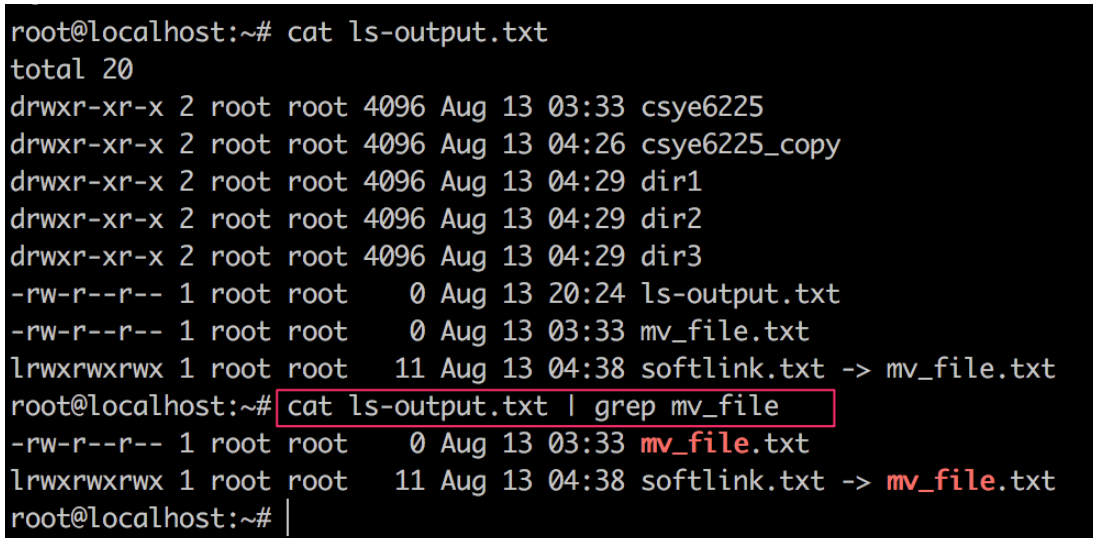
```
root@localhost:~# cat ls-output.txt | tail -n3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

The Difference Between **>** and |

- It may be hard to understand the redirection performed by the pipeline operator **“|”** versus the redirection operator **“>”**.
- The redirection operator (“>”) connects a command with a file while the pipeline operator (“|”) connects the output of one command with the input of a second command.

grep – Print lines matching a pattern

- **grep** is a powerful program used to find text patterns.
- When grep encounters a “pattern”, it prints out the lines containing it.



```
root@localhost:~# cat ls-output.txt
total 20
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root    0 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# cat ls-output.txt | grep mv_file
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

Permissions

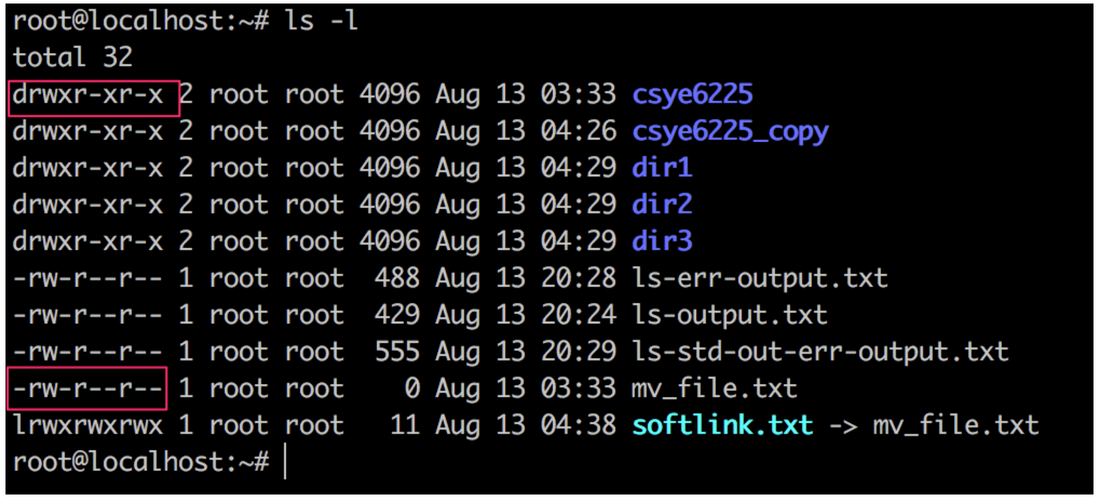
Commands: chmod, su, sudo, chown, passwd

Permissions

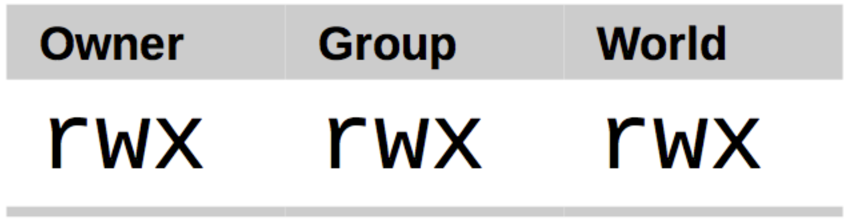
Under the traditional UNIX and Linux filesystem model, every file has a set of nine permission bits that control who can read, write, and execute the contents of the file.

The Permission Bits

- Three sets of permissions define access for the owner of the file, the group owners of the file, and everyone else (in that order).
- Each set has three bits: a **read** bit, a **write** bit, and an **execute** bit (in that order)



```
root@localhost:~# ls -l
total 32
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root  488 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root  555 Aug 13 20:29 ls-std-out-err-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# |
```

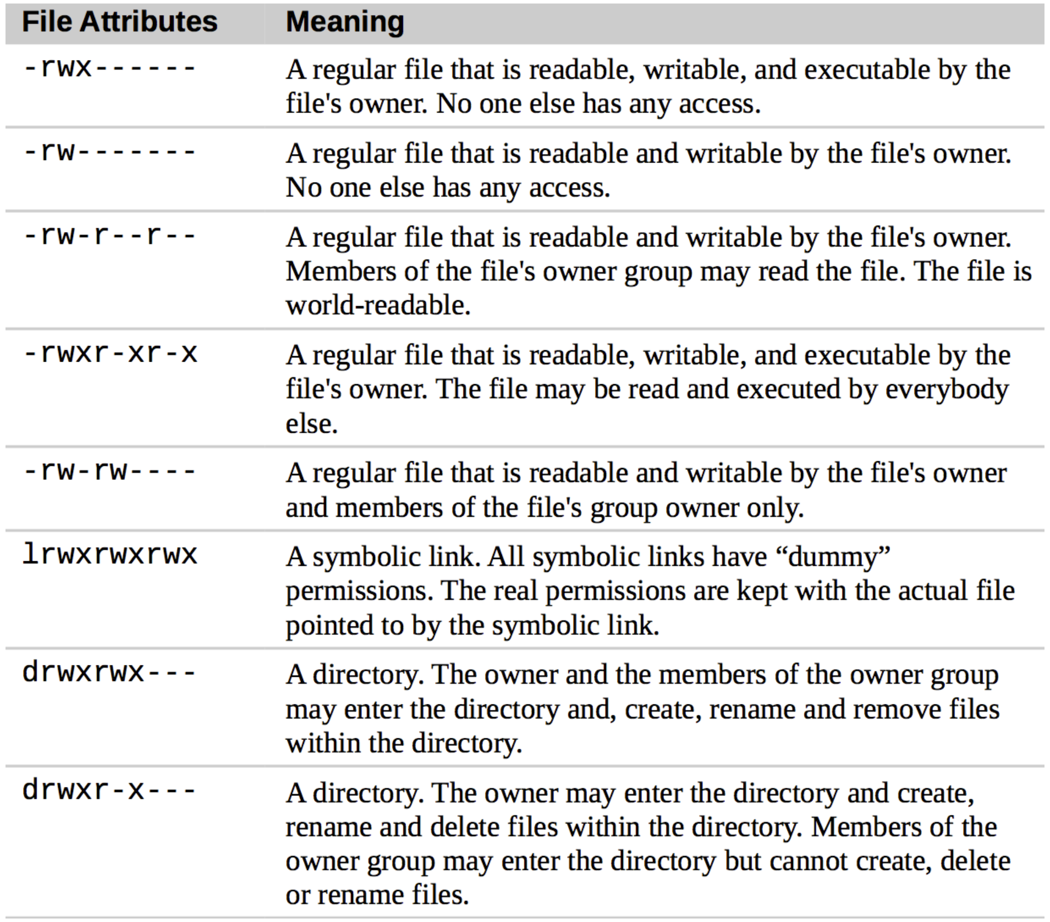


Owner	Group	World
rwx	rwx	rwx

Permission Attributes

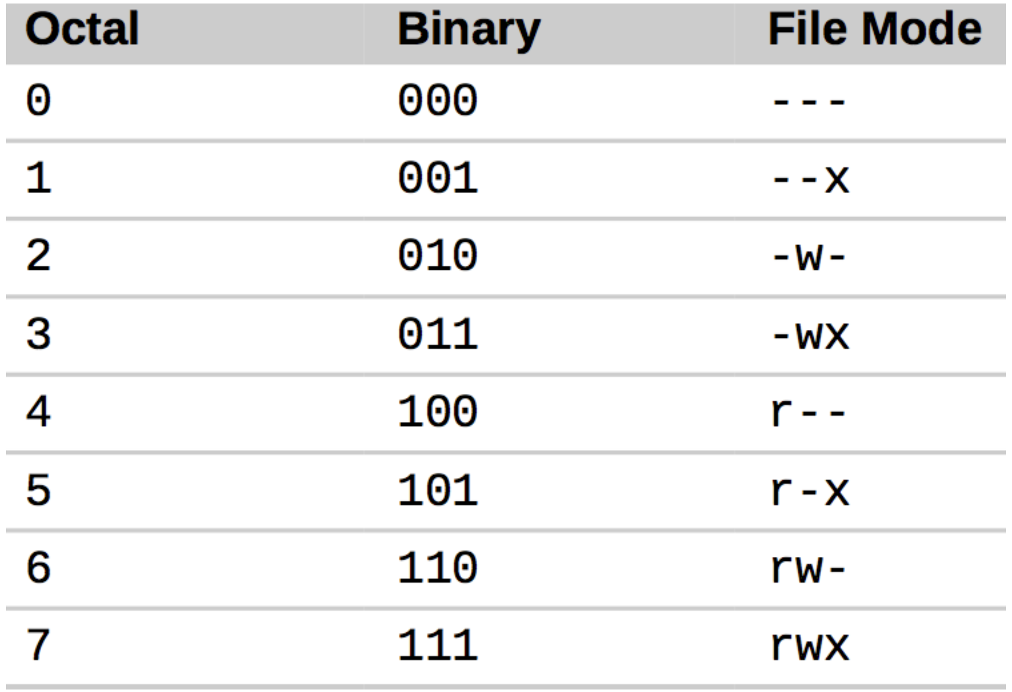
Attribute	Files	Directories
`r`	Allows a file to be opened and read.	Allows a directory's contents to be listed if the execute attribute is also set.
`w`	Allows a file to be written to or truncated, however this attribute does not allow files to be renamed or deleted. The ability to delete or rename files is determined by directory attributes.	Allows files within a directory to be created, deleted, and renamed if the execute attribute is also set.
`x`	Allows a file to be treated as a program and executed. Program files written in scripting languages must also be set as readable to be executed.	Allows a directory to be entered, e.g., cd *directory*.

Permission Attribute Examples



File Attributes	Meaning
`-rwx------`	A regular file that is readable, writable, and executable by the file's owner. No one else has any access.
`-rw-------`	A regular file that is readable and writable by the file's owner. No one else has any access.
`-rw-r--r--`	A regular file that is readable and writable by the file's owner. Members of the file's owner group may read the file. The file is world-readable.
`-rwxr-xr-x`	A regular file that is readable, writable, and executable by the file's owner. The file may be read and executed by everybody else.
`-rw-rw----`	A regular file that is readable and writable by the file's owner and members of the file's group owner only.
`lrwxrwxrwx`	A symbolic link. All symbolic links have “dummy” permissions. The real permissions are kept with the actual file pointed to by the symbolic link.
`drwxrwx---`	A directory. The owner and the members of the owner group may enter the directory and, create, rename and remove files within the directory.
`drwxr-x---`	A directory. The owner may enter the directory and create, rename and delete files within the directory. Members of the owner group may enter the directory but cannot create, delete or rename files.

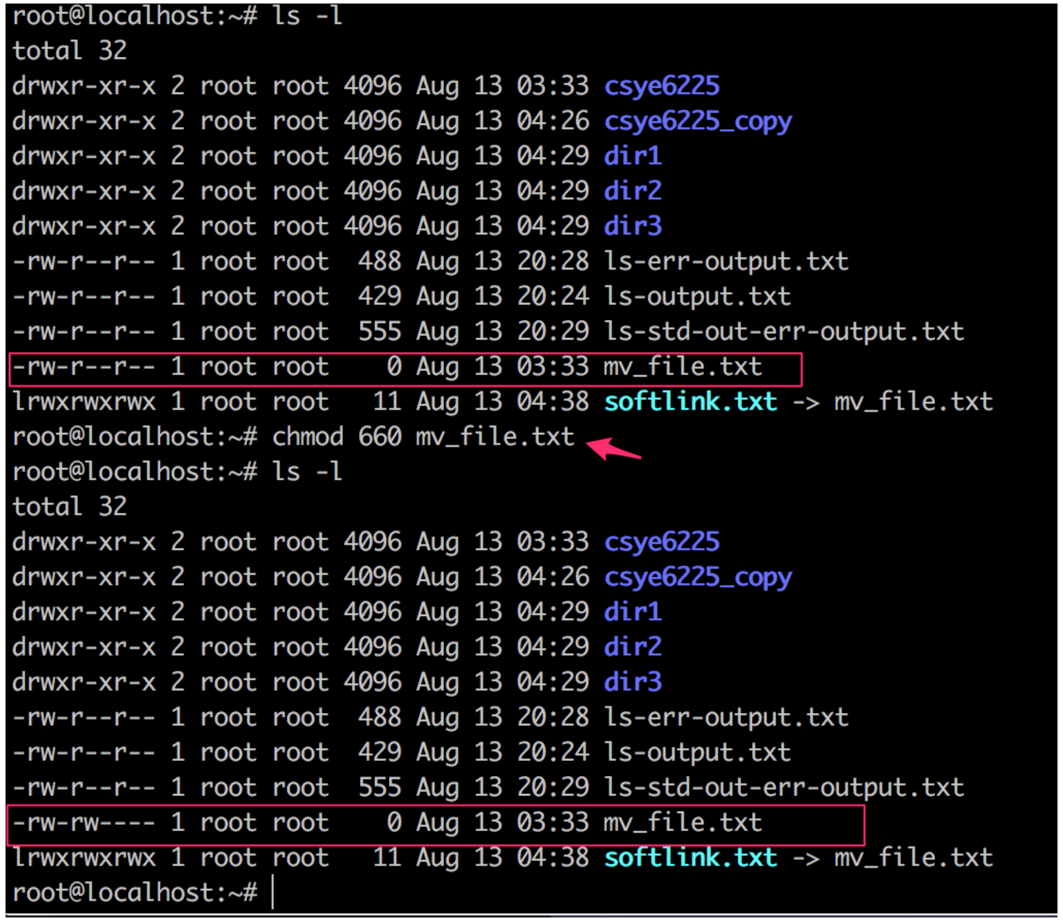
File Modes In Binary And Octal



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

chmod – Change File Mode

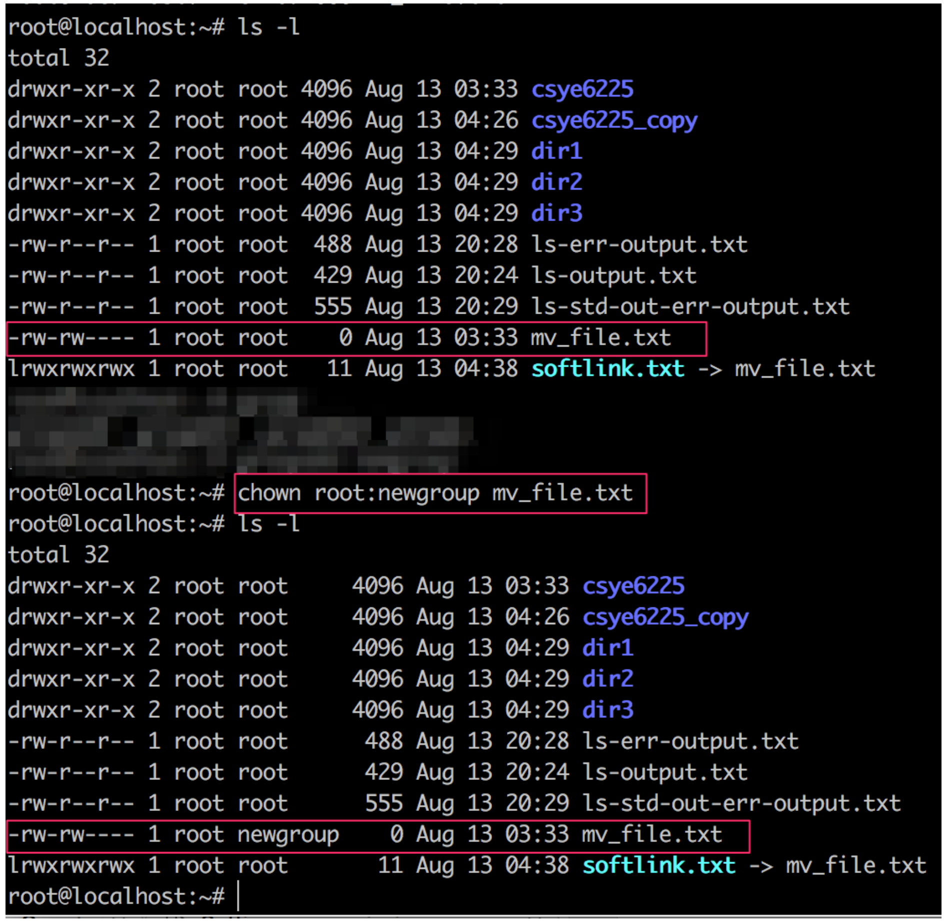
- The **chmod** command changes the permissions on a file.
- Only the owner of the file and the superuser can change its permissions.



```
root@localhost:~# ls -l
total 32
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root  488 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root  555 Aug 13 20:29 ls-std-out-err-output.txt
-rw-r--r-- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~# chmod 660 mv_file.txt
root@localhost:~# ls -l
total 32
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root  488 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root  555 Aug 13 20:29 ls-std-out-err-output.txt
-rw-rw---- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

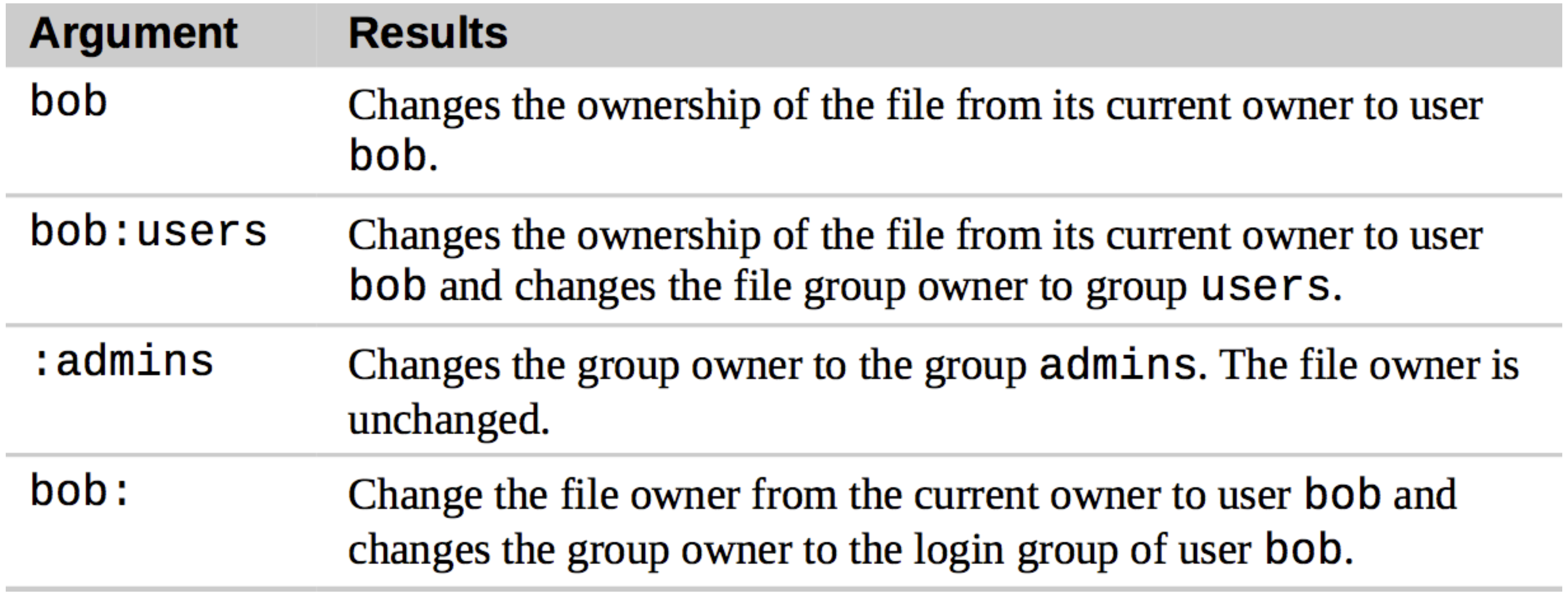
chown – Change File Owner And Group

- The **chown** command is used to change the owner and group owner of a file or directory.
- Superuser privileges are required to use this command.
- **chown** can change the file owner and/or the file group owner depending on the first argument of the command.



```
root@localhost:~# ls -l
total 32
drwxr-xr-x 2 root root 4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root 4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root 4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root  488 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root  429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root  555 Aug 13 20:29 ls-std-out-err-output.txt
-rw-rw---- 1 root root    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root   11 Aug 13 04:38 softlink.txt -> mv_file.txt
[redacted]
root@localhost:~# chown root:newgroup mv_file.txt
root@localhost:~# ls -l
total 32
drwxr-xr-x 2 root root     4096 Aug 13 03:33 csye6225
drwxr-xr-x 2 root root     4096 Aug 13 04:26 csye6225_copy
drwxr-xr-x 2 root root     4096 Aug 13 04:29 dir1
drwxr-xr-x 2 root root     4096 Aug 13 04:29 dir2
drwxr-xr-x 2 root root     4096 Aug 13 04:29 dir3
-rw-r--r-- 1 root root      488 Aug 13 20:28 ls-err-output.txt
-rw-r--r-- 1 root root      429 Aug 13 20:24 ls-output.txt
-rw-r--r-- 1 root root      555 Aug 13 20:29 ls-std-out-err-output.txt
-rw-rw---- 1 root newgroup    0 Aug 13 03:33 mv_file.txt
lrwxrwxrwx 1 root root       11 Aug 13 04:38 softlink.txt -> mv_file.txt
root@localhost:~#
```

chown Argument Examples



Argument	Results
`bob`	Changes the ownership of the file from its current owner to user `bob`.
`bob:users`	Changes the ownership of the file from its current owner to user `bob` and changes the file group owner to group `users`.
`:admins`	Changes the group owner to the group `admins`. The file owner is unchanged.
`bob:`	Change the file owner from the current owner to user `bob` and changes the group owner to the login group of user `bob`.

su – Run A Shell With Substitute User And Group IDs

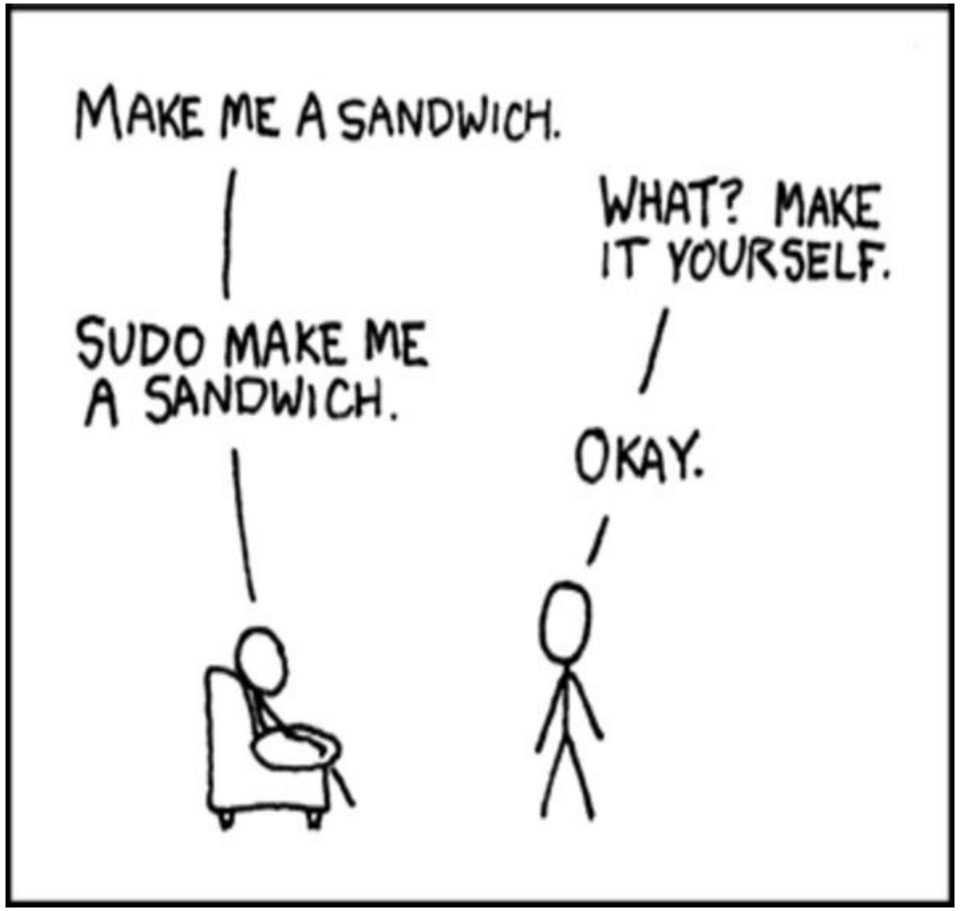
The **su** command is used to start a shell as another user. The command syntax looks like this:

```
su [-[l]] [user]
```

If the “-l” option is included, the resulting shell session is a login shell for the specified user. This means that the user's environment is loaded, and the working directory is changed to the user's home directory.

sudo – Execute A Command As Another User

- The **sudo** command is like su in many ways but has some important additional capabilities.
- The administrator can configure sudo to allow an ordinary user to execute commands as a different user (usually the superuser) in a very controlled way.
- A user may be restricted to one or more specific commands and no others.
- Another important difference is that the use of sudo does not require access to the superuser's password. To authenticate using sudo, the user uses his/her own password.

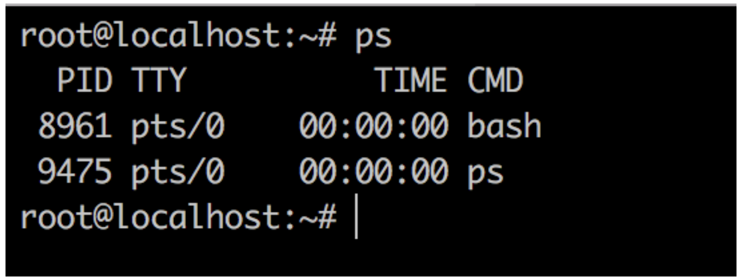


Processes

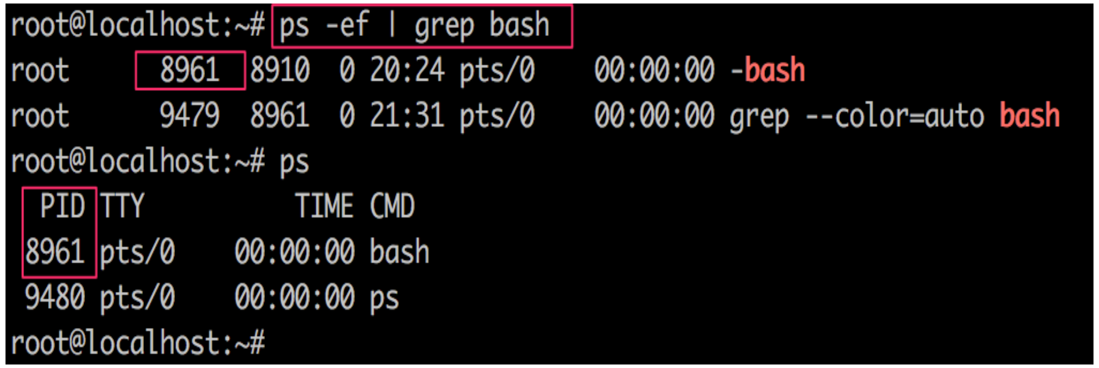
Commands: ps, top, fg, kill, shutdown, nohup

ps – Viewing Processes

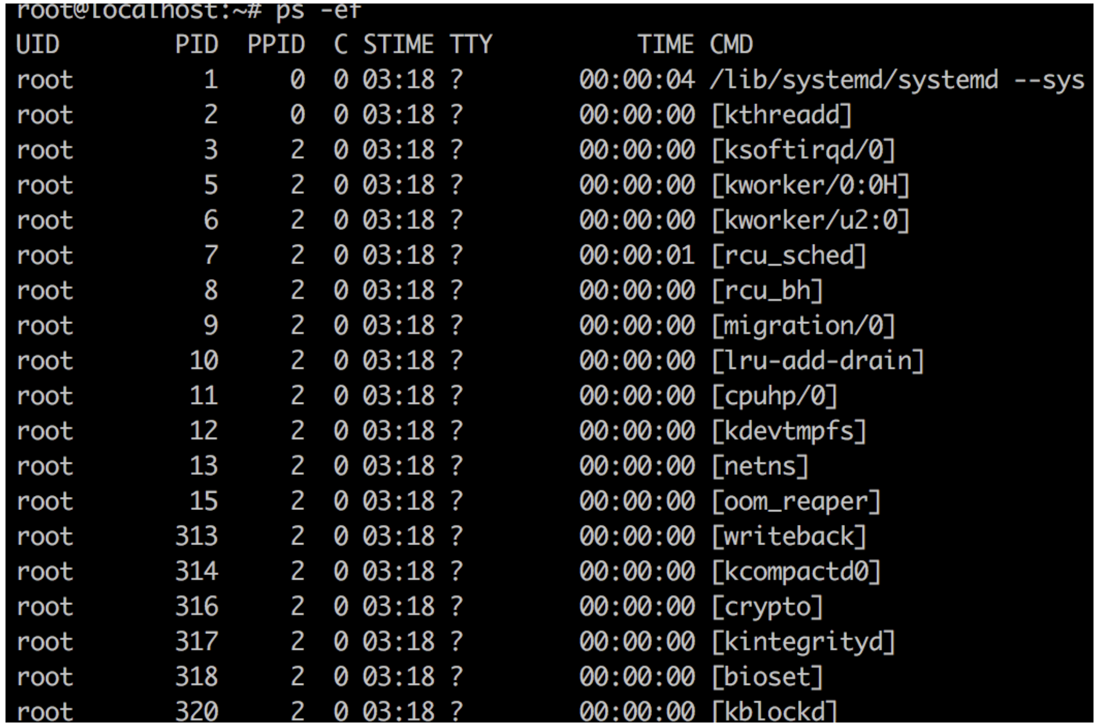
- **ps** is the most used command to view processes.



```
root@localhost:~# ps
  PID TTY          TIME CMD
 8961 pts/0    00:00:00 bash
 9475 pts/0    00:00:00 ps
root@localhost:~# 
```



```
root@localhost:~# ps -ef | grep bash
root      8961  8910  0 20:24 pts/0    00:00:00 -bash
root      9479  8961  0 21:31 pts/0    00:00:00 grep --color=auto bash
root@localhost:~# ps
  PID TTY          TIME CMD
 8961 pts/0    00:00:00 bash
 9480 pts/0    00:00:00 ps
root@localhost:~#
```



```
root@localhost:~# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 03:18 ?        00:00:04 /lib/systemd/systemd --sys
root         2     0  0 03:18 ?        00:00:00 [kthreadd]
root         3     2  0 03:18 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 03:18 ?        00:00:00 [kworker/0:0H]
root         6     2  0 03:18 ?        00:00:00 [kworker/u2:0]
root         7     2  0 03:18 ?        00:00:01 [rcu_sched]
root         8     2  0 03:18 ?        00:00:00 [rcu_bh]
root         9     2  0 03:18 ?        00:00:00 [migration/0]
root        10     2  0 03:18 ?        00:00:00 [lru-add-drain]
root        11     2  0 03:18 ?        00:00:00 [cpuhp/0]
root        12     2  0 03:18 ?        00:00:00 [kdevtmpfs]
root        13     2  0 03:18 ?        00:00:00 [netns]
root        15     2  0 03:18 ?        00:00:00 [oom_reaper]
root       313     2  0 03:18 ?        00:00:00 [writeback]
root       314     2  0 03:18 ?        00:00:00 [kcompactd0]
root       316     2  0 03:18 ?        00:00:00 [crypto]
root       317     2  0 03:18 ?        00:00:00 [kintegrityd]
root       318     2  0 03:18 ?        00:00:00 [bioset]
root       320     2  0 03:18 ?        00:00:00 [kblockd]
```

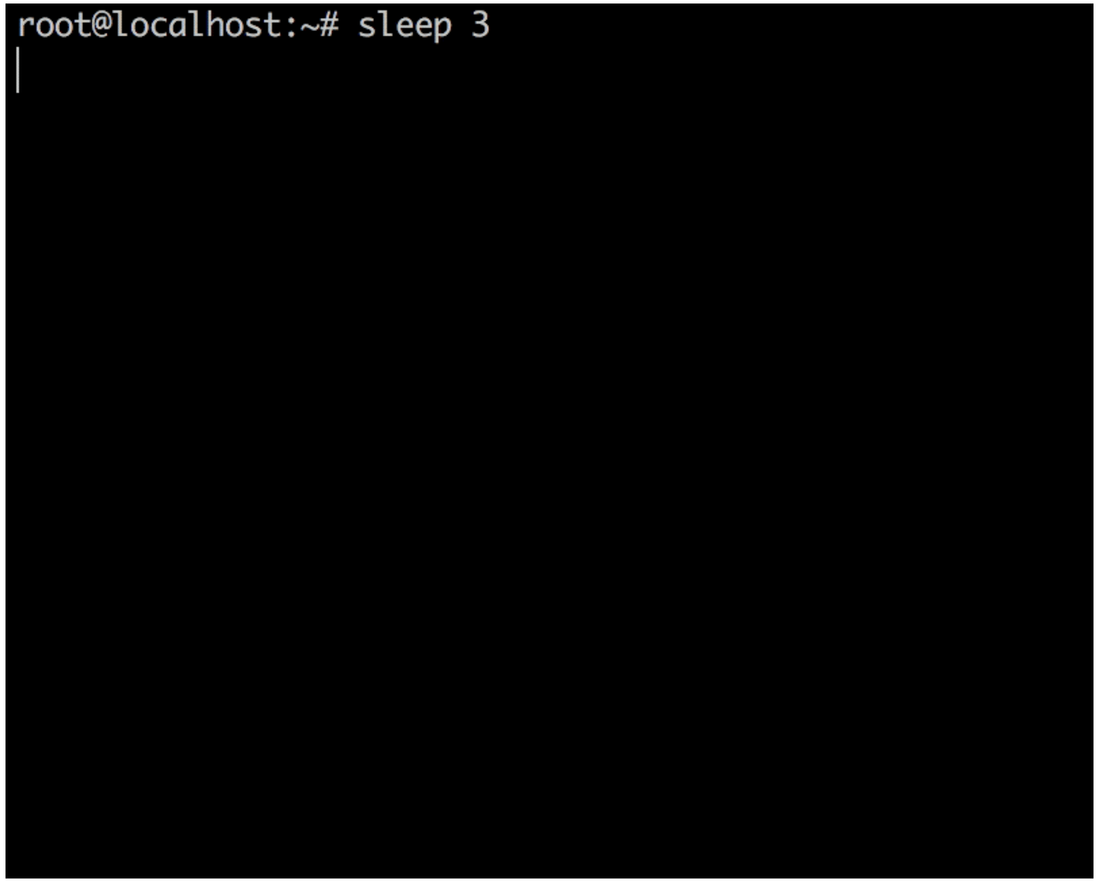
top – Viewing Processes Dynamically

- While the **ps** command can reveal a lot about what the machine is doing, it provides only a snapshot of the machine's state at the moment the **ps** command is executed.
- To see a more dynamic view of the machine's activity, we use the **top** command

```
root@localhost:~#
```

Putting A Process In The Background

- To launch a program so that it is immediately placed in the background, we follow the command with an **“&”** character


```
root@localhost:~# sleep 3
```

jobs – List background or suspended processes

When a process is running, backgrounded or suspended, it will be entered onto a list along with a job number. To examine this list, use the command **jobs**.

fg – Returning A Process To The Foreground

To return a process to the foreground, use the **fg** command .

Example Usage: fg <JOB_ID>

kill – Terminate Process

The **kill** command is used to "kill" processes.

This allows us to terminate programs that need killing.

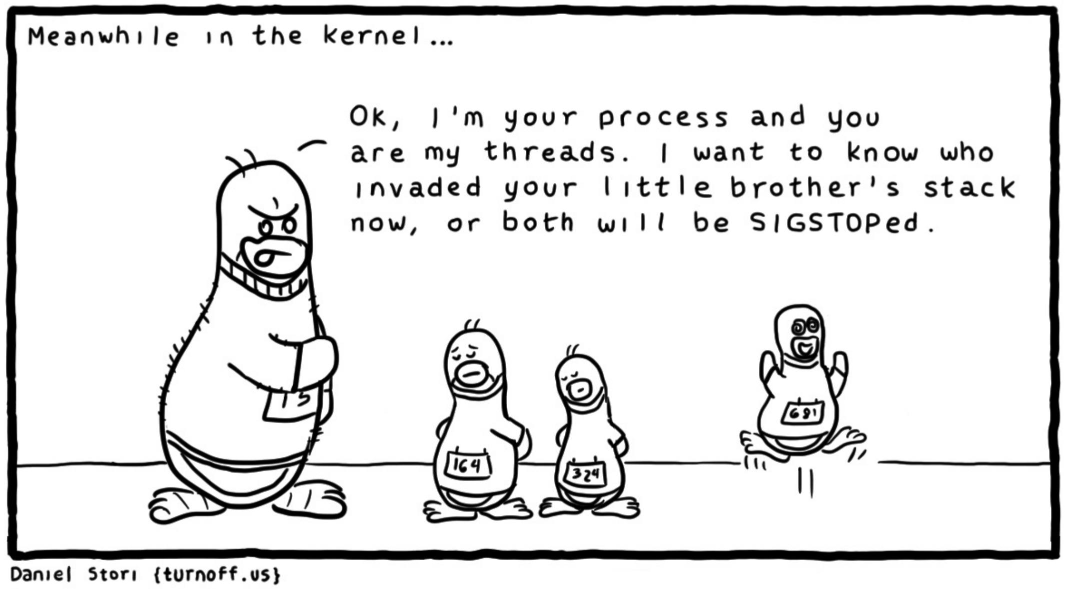
Example Usage: kill <pid>

If a process refuses to be killed, uses the **-9** option.

Example Usage: kill -9 <pid>

Notes:

- kill -9 -1 will kill all process that you have permission to kill.
- It will also end your current login session.
- Never execute this command as root as it will KILL ALL processes CRASHING your server.


Meanwhile in the kernel...
Ok, I'm your process and you are my threads. I want to know who invaded your little brother's stack now, or both will be SIGSTOPed.
164
324
681
Daniel Stori {turnoff.us}

Shutdown – Poweroff or Reboot

shutdown command will terminate processes in orderly function before powering off the system.

sudo shutdown –r now – reboot the system

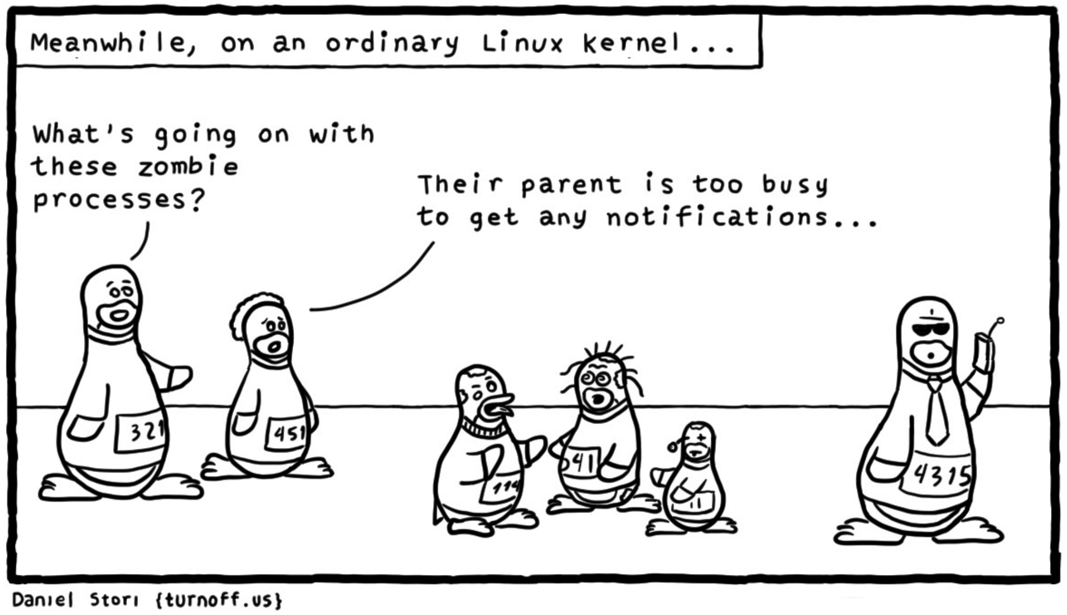
nohup – run a command without hangups

- **nohup** command is used to continue run programs in background after logging off.
- Ignores all **hangup** signals.
- Writes all terminal output to nohup.out file. nohup.out file is created in the directory the command is executed.

Example Usage: nohup *somecommand* &

Zombie Process

- On Unix and Unix-like computer operating systems, a zombie process or defunct process is a process that has completed execution (via the exit system call) but still has an entry in the process table: it is a process in the "Terminated state".
- This occurs for child processes, where the entry is still needed to allow the parent process to read its child's exit status: once the exit status is read via the wait system call, the zombie's entry is removed from the process table, and it is said to be "reaped".
- Zombie process can be identified with **ps aux | grep 'Z'** command.


Meanwhile, on an ordinary Linux kernel...
What's going on with these zombie processes?
Their parent is too busy to get any notifications...
321
451
114
541
4315
Daniel Stori {turnoff.us}

Disk Management

Command: du, df

du – estimate file space usage

Usage: du [OPTION]... [FILE]...

Example Usage:

- du –sm (display output in Megabytes)
- du –sg (display output in Gigabytes)

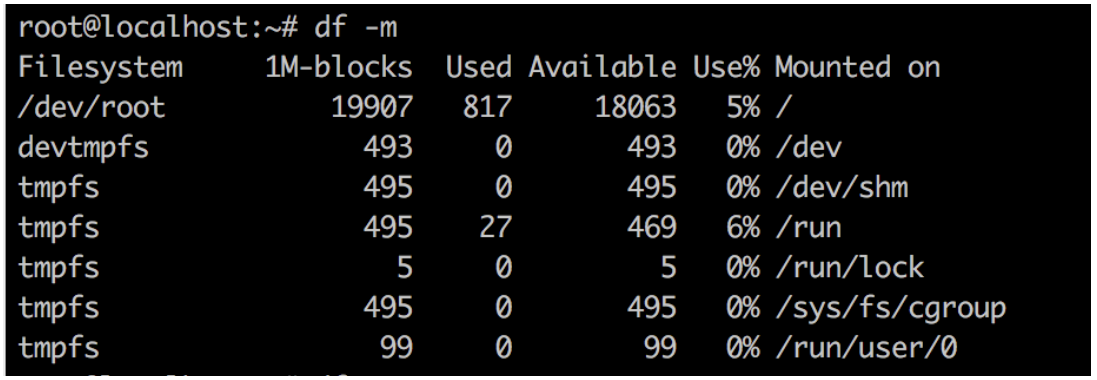
df – report file system disk space usage

Usage: df [OPTION]... [FILE]...

Example Usage:

- df –m

  (display output in Megabytes)



```
root@localhost:~# df -m
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/root          19907   817     18063   5% /
devtmpfs             493     0       493   0% /dev
tmpfs                495     0       495   0% /dev/shm
tmpfs                495    27       469   6% /run
tmpfs                  5     0         5   0% /run/lock
tmpfs                495     0       495   0% /sys/fs/cgroup
tmpfs                 99     0        99   0% /run/user/0
```

Misc

Commands: tar, zip, unzip, ssh, scp

Archives

- **tar** – to archive a file

  Usage: tar [OPTION] DEST SOURCE

  Example Usage: tar -cvf /home/archive.tar /home/original

  tar -xvf /home/archive.tar
- **zip** – package and compress (archive) files

  Usage: zip [OPTION] DEST SOURCE

  Example Usage: zip original.zip original
- **unzip** – list, test and extract compressed files in a ZIP archive

  Usage: unzip filename

  Example Usage: unzip original.zip

Network

- **ssh – SSH client (remote login program)**

  SSH is a program for logging into a remote machine and for executing commands on a remote machine.

  Usage: ssh [options] [user]@hostname

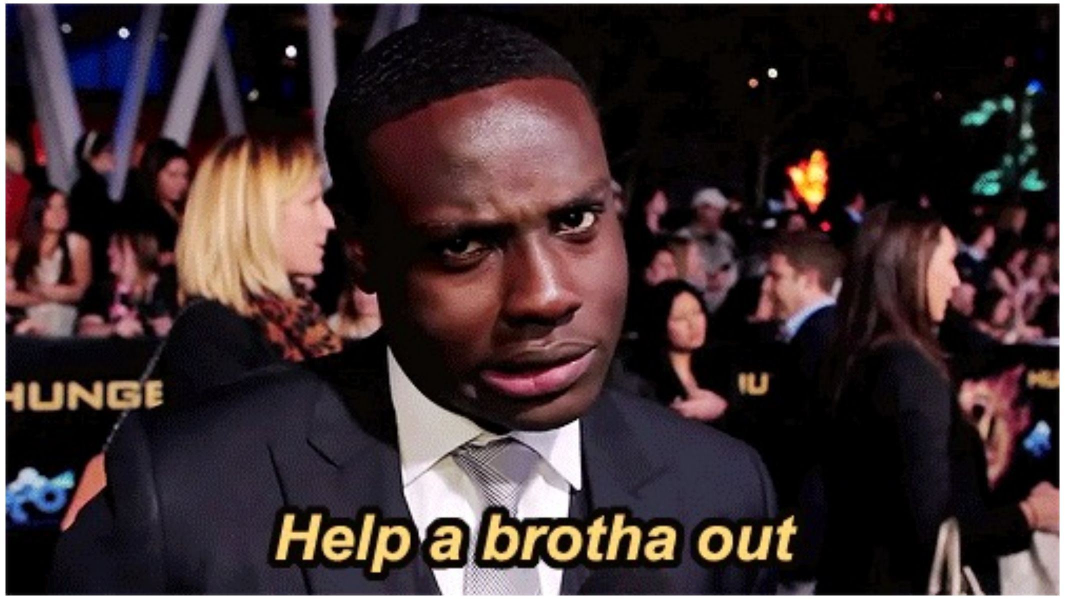
  Example Usage: ssh guest@10.105.11.20

- **scp – secure copy (remote file copy program)**

  scp copies files between hosts on a network

  Usage: scp [options] [[user]@host1:file1] [[user]@host2:file2]

  Example Usage: scp file1.txt guest@10.105.11.20:~/Desktop/



Getting Help

Commands: man

man – Display manual page

The manual pages tell you which options a particular command can take, and how each option modifies the behavior of the command.

```
root@localhost:~#
```

Additional Resources

See Lecture Page