Linux/Unix Shell Intro

Dr. Chokchai (Box) Leangsuksun

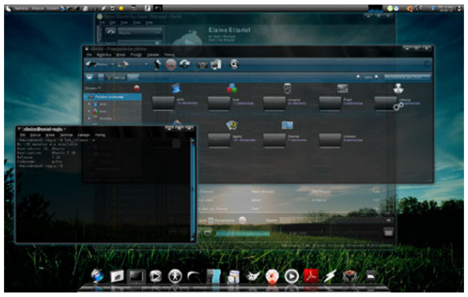
Louisiana Tech University

Original slides were created by Dr. deBry from uvu.edu

Outline

USER Interface

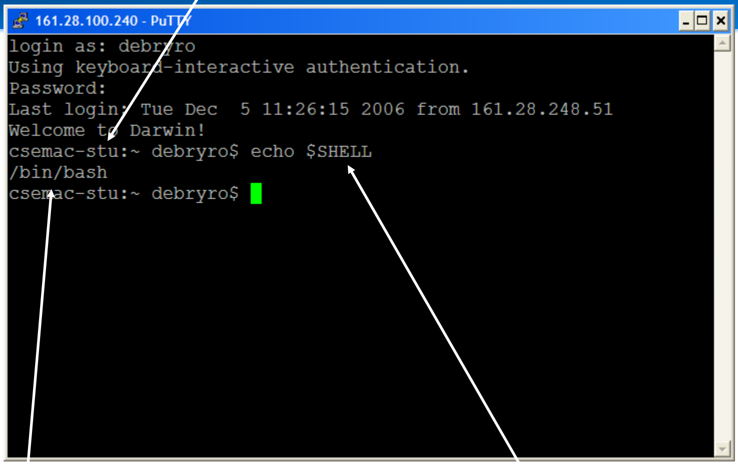
- Command Line Interface
  - Shell commands
  - C-shell, tsh-shell, bourne shell etc..
- Graphic User Interface
  - GNOME, KDE etc..



CLI or Shell

- Command Line Interface
- The shell is a command interpreter
- It provides the interface between a user and the operating system via command line
- Shell commands. Eg. ls, cd, pwd etc
- Various shells: C-shell, tsh-shell, bourne shell etc..

- When you log in to a Unix system, a shell starts running. You interact with the default shell this is the shell prompt

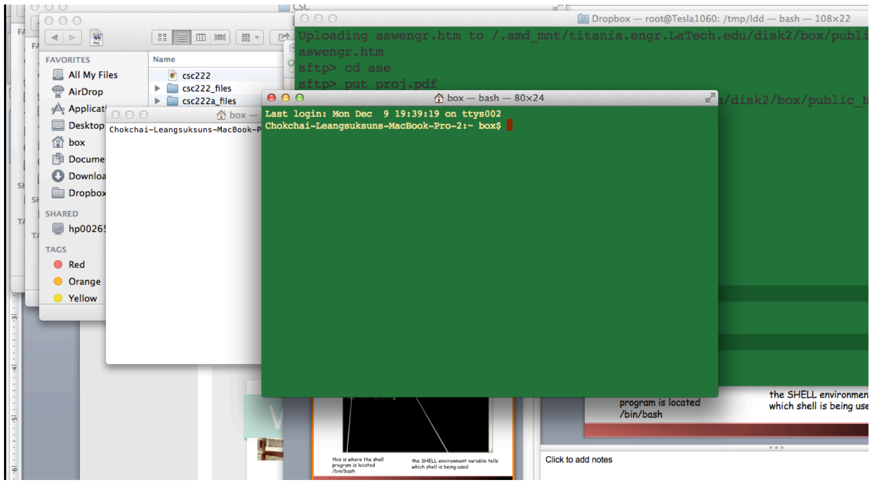


```
login as: debryro
Using keyboard-interactive authentication.
Password:
Last login: Tue Dec  5 11:26:15 2006 from 161.28.248.51
Welcome to Darwin!
csemac-stu:~ debryro$ echo $SHELL
/bin/bash
csemac-stu:~ debryro$
```

this is where the shell program is located /bin/bash

the SHELL environment variable tells which shell is being used

In MAC



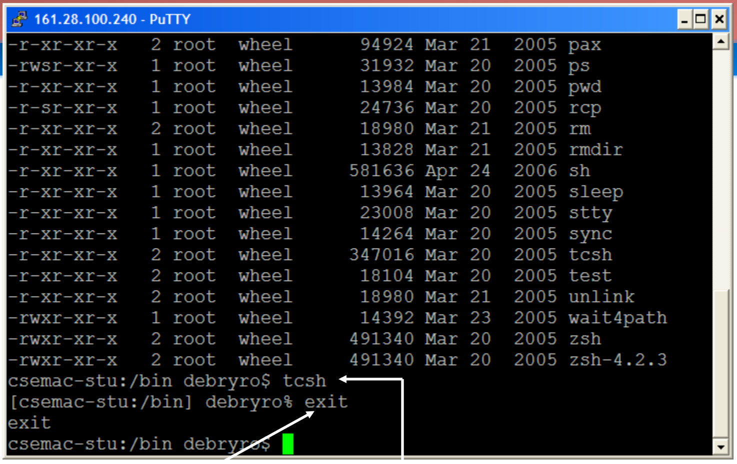
```
Uploading aswengr.htm to /.amd_mnt/titania.engr.LaTech.edu/disk2/box/publi
aswengr.htm
sftp> cd ase
sftp> put proj.pdf
```

```
Last login: Mon Dec  9 19:39:19 on ttys002
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$
```

Louisiana Tech University 6

Various shell programs

Shell name	Program (Command) name
rc	rc
Bourne Shell	sh
C Shell	csh
Bourne Again Shell	bash
Z shell	zsh
Korn Shell	ksh
TC	tcsh

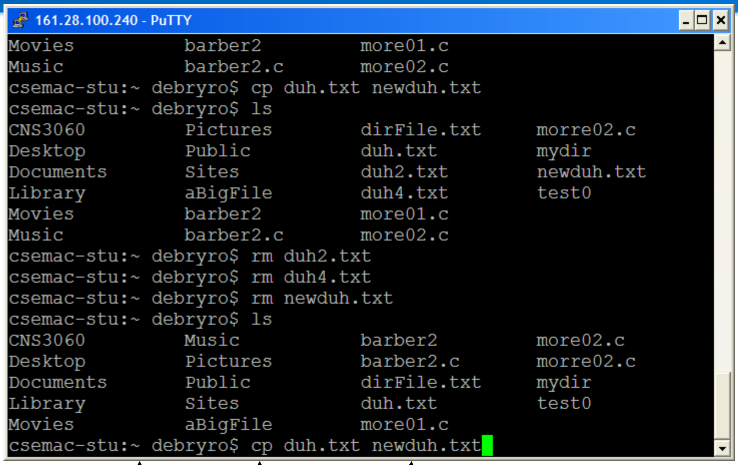


```
-r-xr-xr-x   2 root  wheel      94924 Mar 21  2005 pax
-rwsr-xr-x   1 root  wheel      31932 Mar 20  2005 ps
-r-xr-xr-x   1 root  wheel      13984 Mar 20  2005 pwd
-r-sr-xr-x   1 root  wheel      24736 Mar 20  2005 rcp
-r-xr-xr-x   2 root  wheel      18980 Mar 21  2005 rm
-r-xr-xr-x   1 root  wheel      13828 Mar 21  2005 rmdir
-r-xr-xr-x   1 root  wheel     581636 Apr 24  2006 sh
-r-xr-xr-x   1 root  wheel      13964 Mar 20  2005 sleep
-r-xr-xr-x   1 root  wheel      23008 Mar 20  2005 stty
-r-xr-xr-x   1 root  wheel      14264 Mar 20  2005 sync
-r-xr-xr-x   2 root  wheel     347016 Mar 20  2005 tcsh
-r-xr-xr-x   2 root  wheel      18104 Mar 20  2005 test
-r-xr-xr-x   2 root  wheel      18980 Mar 21  2005 unlink
-rwxr-xr-x   1 root  wheel      14392 Mar 23  2005 wait4path
-rwxr-xr-x   2 root  wheel     491340 Mar 20  2005 zsh
-rwxr-xr-x   2 root  wheel     491340 Mar 20  2005 zsh-4.2.3
csemac-stu:/bin debryro$ tcsh
[csemac-stu:/bin] debryro% exit
exit
csemac-stu:/bin debryro$ 
```

you can change shells by typing the shell command

return to the default shell
by typing "exit"

The shell command line



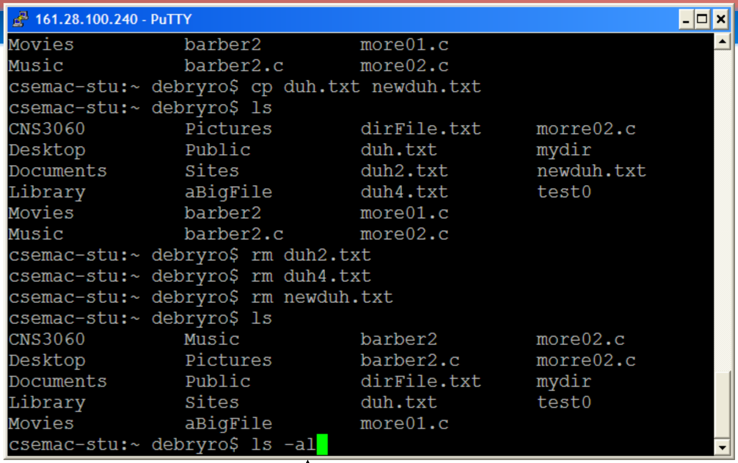
```
Movies              barber2             more01.c
Music               barber2.c           more02.c
csemac-stu:~ debryro$ cp duh.txt newduh.txt
csemac-stu:~ debryro$ ls
CNS3060             Pictures            dirFile.txt         morre02.c
Desktop             Public              duh.txt             mydir
Documents           Sites               duh2.txt            newduh.txt
Library             aBigFile            duh4.txt            test0
Movies              barber2             more01.c
Music               barber2.c           more02.c
csemac-stu:~ debryro$ rm duh2.txt
csemac-stu:~ debryro$ rm duh4.txt
csemac-stu:~ debryro$ rm newduh.txt
csemac-stu:~ debryro$ ls
CNS3060             Music               barber2             more02.c
Desktop             Pictures            barber2.c           morre02.c
Documents           Public              dirFile.txt         mydir
Library             Sites               duh.txt             test0
Movies              aBigFile            more01.c
csemac-stu:~ debryro$ cp duh.txt newduh.txt
```

prompt shows current directory.
~ is your home directory

command

list of arguments

The shell command line



```
Movies              barber2             more01.c
Music               barber2.c           more02.c
csemac-stu:~ debryro$ cp duh.txt newduh.txt
csemac-stu:~ debryro$ ls
CNS3060             Pictures            dirFile.txt         morre02.c
Desktop             Public              duh.txt             mydir
Documents           Sites               duh2.txt            newduh.txt
Library             aBigFile            duh4.txt            test0
Movies              barber2             more01.c
Music               barber2.c           more02.c
csemac-stu:~ debryro$ rm duh2.txt
csemac-stu:~ debryro$ rm duh4.txt
csemac-stu:~ debryro$ rm newduh.txt
csemac-stu:~ debryro$ ls
CNS3060             Music               barber2             more02.c
Desktop             Pictures            barber2.c           morre02.c
Documents           Public              dirFile.txt         mydir
Library             Sites               duh.txt             test0
Movies              aBigFile            more01.c
csemac-stu:~ debryro$ ls -al
```

command options (usually preceded with a hyphen)

finger displays information for logged in users

finger [options] user-list

-l	detailed listing
-s	brief listing

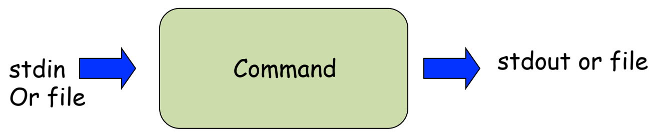
How command line is interpreted ?

What happens when you type a command?

1. The shell parses the command line
2. It looks for a program that matches the command
3. It starts a new process and runs that program
4. While the program executes, the shell sleeps
5. When the program is done, the shell wakes up
6. It displays its prompt
7. It waits for the user to type another command

- Note there are some exceptions for built-in commands
- E.g. pwd or cd etc..

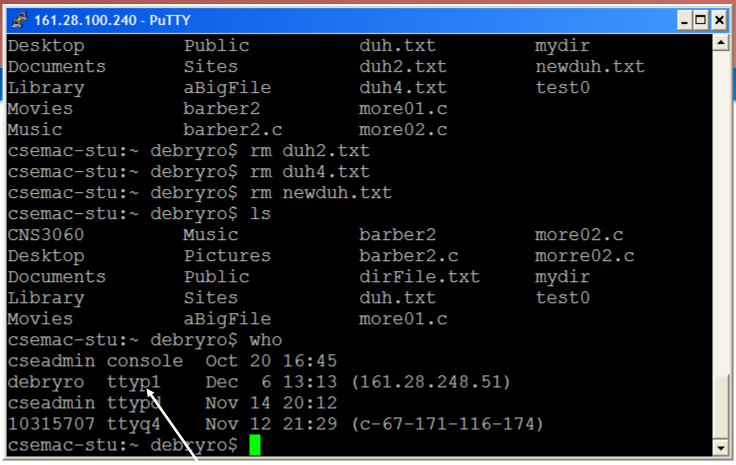
The shell command line



- Commands generally get their input from stdin and send their output to stdout
- stdin – keyboard
- stdout – screen
- stderr – screen
- Commands can take input from a file too
- With redirection, you can map stdin & stdout to files.

command interpreter

- case and syntax sensitive
- command: **`echo $SHELL`**
- **Command Line Processing** :
  1) evaluate special characters, such as: **~ $ & * ? \ ' " ` |**
  2) decide which program to execute
     - pathname, alias, shell command, search the **`$PATH`**
  3) execute appropriate program, passing to it the parameter list
  4) save the execution status in the **`$status`** variable (0 is considered success)



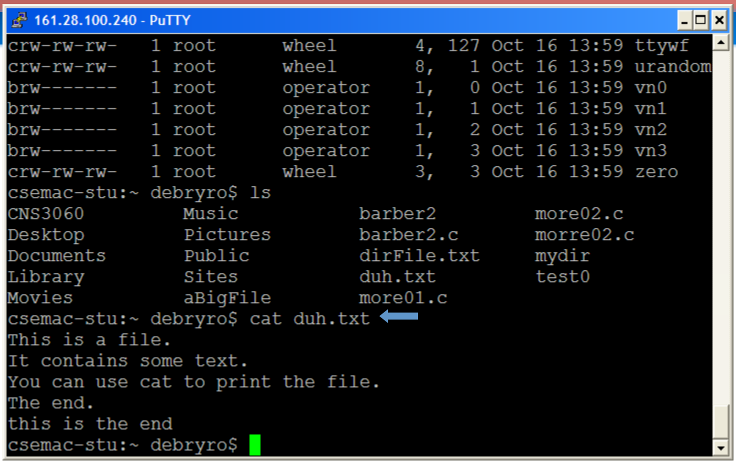
```
Desktop          Public           duh.txt          mydir
Documents        Sites            duh2.txt         newduh.txt
Library          aBigFile         duh4.txt         test0
Movies           barber2          more01.c
Music            barber2.c        more02.c
csemac-stu:~ debryro$ rm duh2.txt
csemac-stu:~ debryro$ rm duh4.txt
csemac-stu:~ debryro$ rm newduh.txt
csemac-stu:~ debryro$ ls
CNS3060          Music            barber2          more02.c
Desktop          Pictures         barber2.c        morre02.c
Documents        Public           dirFile.txt      mydir
Library          Sites            duh.txt          test0
Movies           aBigFile         more01.c
csemac-stu:~ debryro$ who
cseadmin console  Oct 20 16:45
debryro  ttyp1    Dec  6 13:13 (161.28.248.51)
cseadmin ttyp4    Nov 14 20:12
10315707 ttyq4    Nov 12 21:29 (c-67-171-116-174)
csemac-stu:~ debryro$
```

if you run the who command, the system tells you who is logged in and at what terminal.

but . . . this is really a file in the Unix file system that represents a real device, in this case a terminal

commands read from and write to this file!

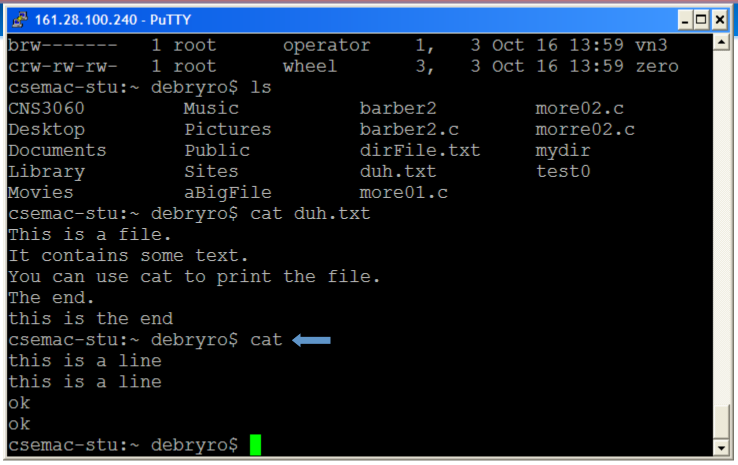
The shell command line



```
crw-rw-rw-   1 root      wheel       4, 127 Oct 16 13:59 ttywf
crw-rw-rw-   1 root      wheel       8,   1 Oct 16 13:59 urandom
brw-------   1 root      operator    1,   0 Oct 16 13:59 vn0
brw-------   1 root      operator    1,   1 Oct 16 13:59 vn1
brw-------   1 root      operator    1,   2 Oct 16 13:59 vn2
brw-------   1 root      operator    1,   3 Oct 16 13:59 vn3
crw-rw-rw-   1 root      wheel       3,   3 Oct 16 13:59 zero
csemac-stu:~ debryro$ ls
CNS3060          Music            barber2          more02.c
Desktop          Pictures         barber2.c        morre02.c
Documents        Public           dirFile.txt      mydir
Library          Sites            duh.txt          test0
Movies           aBigFile         more01.c
csemac-stu:~ debryro$ cat duh.txt
This is a file.
It contains some text.
You can use cat to print the file.
The end.
this is the end
csemac-stu:~ debryro$
```

the cat command is a good example. It takes its input from a file and outputs to stdout.

The shell command line



```
brw-------   1 root     operator    1,   3 Oct 16 13:59 vn3
crw-rw-rw-   1 root     wheel       3,   3 Oct 16 13:59 zero
csemac-stu:~ debryro$ ls
CNS3060         Music           barber2         more02.c
Desktop         Pictures        barber2.c       morre02.c
Documents       Public          dirFile.txt     mydir
Library         Sites           duh.txt         test0
Movies          aBigFile        more01.c
csemac-stu:~ debryro$ cat duh.txt
This is a file.
It contains some text.
You can use cat to print the file.
The end.
this is the end
csemac-stu:~ debryro$ cat
this is a line
this is a line
ok
ok
csemac-stu:~ debryro$
```

if you type the command with no parameters, it takes its input from stdin. It will do this until you type ctrl-D (end of file).

Redirection

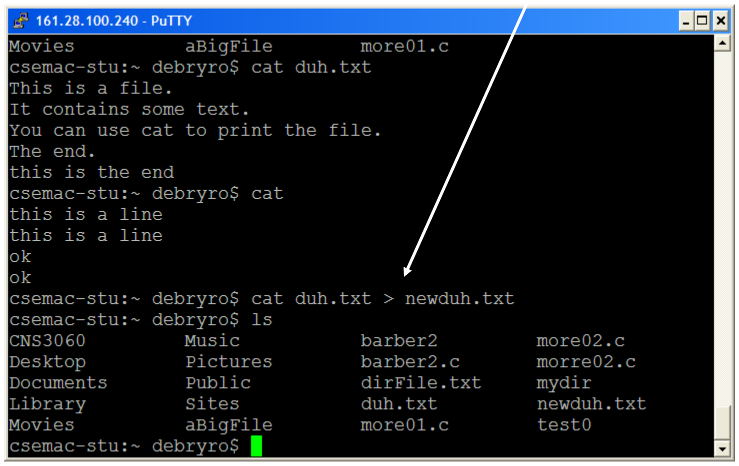
- Shell can get its input from some places
- other than stdin or send its output to some place other
- than stdout by using redirection.
- Redirection special symbols: < > or >>

Redirecting standard input/output

- Stdin from a file
- command [arguments] < filename

- Stdout to a file
- command [arguments] > filename

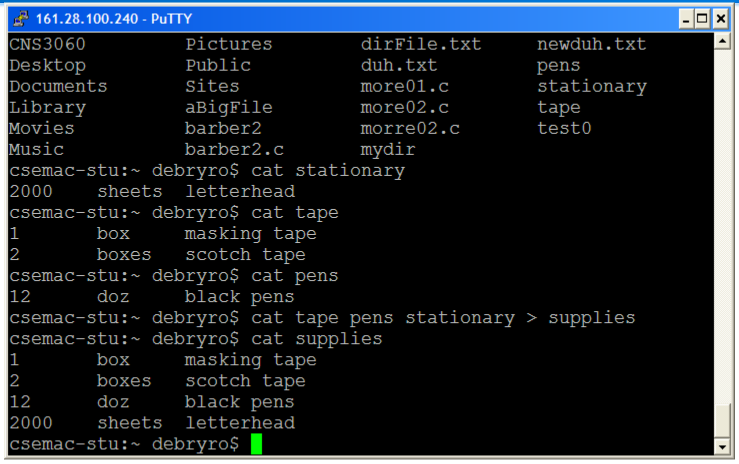
- Stdout appending to a file
- command [arguments] >> filename


redirect output to newduh.txt

161.28.100.240 - PuTTY

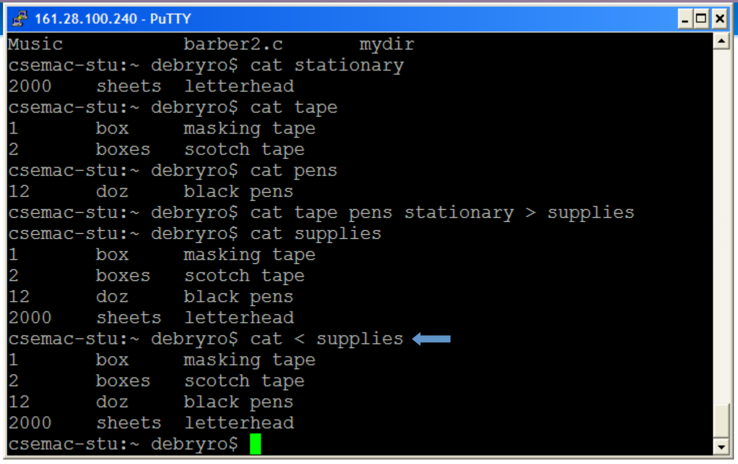
```
Movies          aBigFile        more01.c
csemac-stu:~ debryro$ cat duh.txt
This is a file.
It contains some text.
You can use cat to print the file.
The end.
this is the end
csemac-stu:~ debryro$ cat
this is a line
this is a line
ok
ok
csemac-stu:~ debryro$ cat duh.txt > newduh.txt
csemac-stu:~ debryro$ ls
CNS3060         Music           barber2         more02.c
Desktop         Pictures        barber2.c       morre02.c
Documents       Public          dirFile.txt     mydir
Library         Sites           duh.txt         newduh.txt
Movies          aBigFile        more01.c        test0
csemac-stu:~ debryro$
```

Concatenating Files with cat command



161.28.100.240 - PuTTY

```
CNS3060           Pictures          dirFile.txt       newduh.txt
Desktop           Public            duh.txt           pens
Documents         Sites             more01.c          stationary
Library           aBigFile          more02.c          tape
Movies            barber2           morre02.c         test0
Music             barber2.c         mydir
csemac-stu:~ debryro$ cat stationary
2000     sheets  letterhead
csemac-stu:~ debryro$ cat tape
1        box     masking tape
2        boxes   scotch tape
csemac-stu:~ debryro$ cat pens
12       doz     black pens
csemac-stu:~ debryro$ cat tape pens stationary > supplies
csemac-stu:~ debryro$ cat supplies
1        box     masking tape
2        boxes   scotch tape
12       doz     black pens
2000     sheets  letterhead
csemac-stu:~ debryro$
```



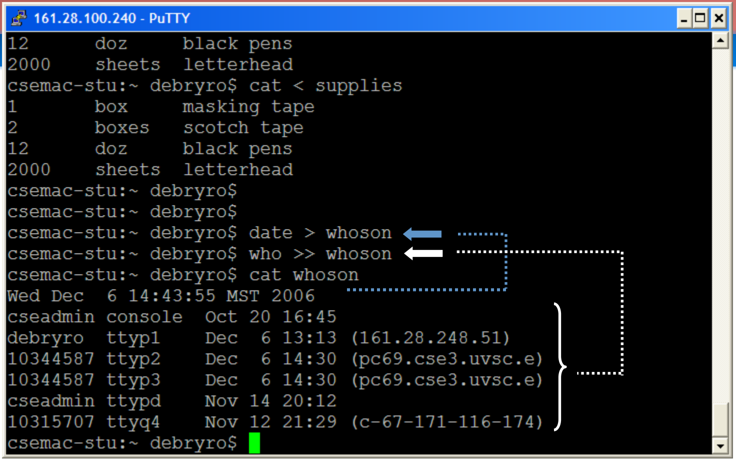
161.28.100.240 - PuTTY

```
Music             barber2.c         mydir
csemac-stu:~ debryro$ cat stationary
2000     sheets  letterhead
csemac-stu:~ debryro$ cat tape
1        box     masking tape
2        boxes   scotch tape
csemac-stu:~ debryro$ cat pens
12       doz     black pens
csemac-stu:~ debryro$ cat tape pens stationary > supplies
csemac-stu:~ debryro$ cat supplies
1        box     masking tape
2        boxes   scotch tape
12       doz     black pens
2000     sheets  letterhead
csemac-stu:~ debryro$ cat < supplies
1        box     masking tape
2        boxes   scotch tape
12       doz     black pens
2000     sheets  letterhead
csemac-stu:~ debryro$
```

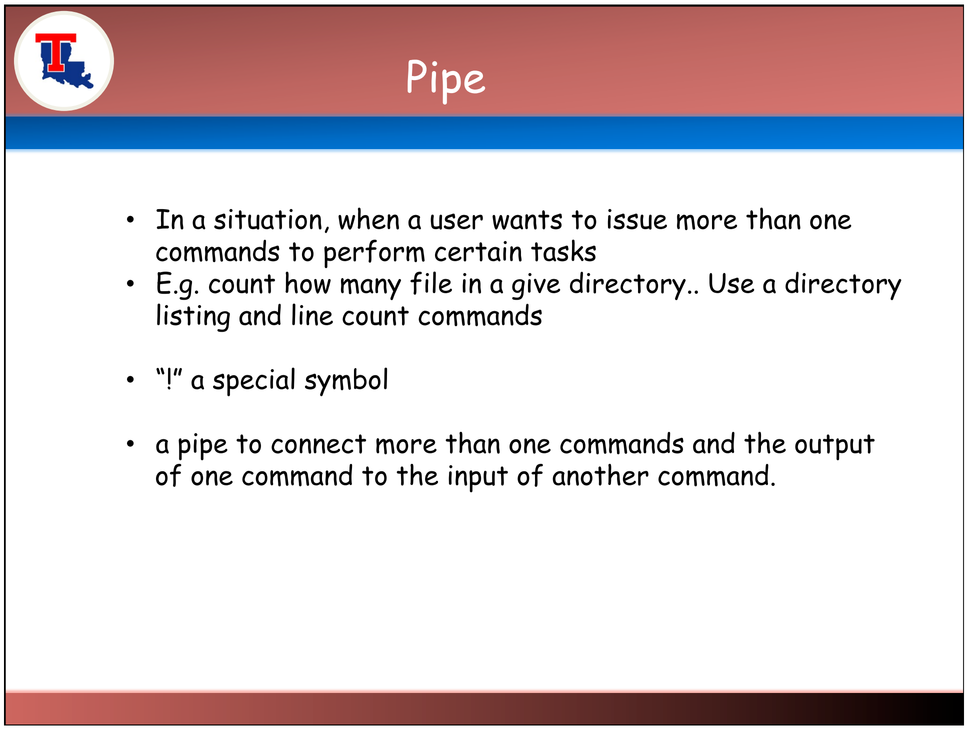
cat takes its input from the file supplies
and writes to standard output

Appending standard ouput to a file

command [arguments] >> filename

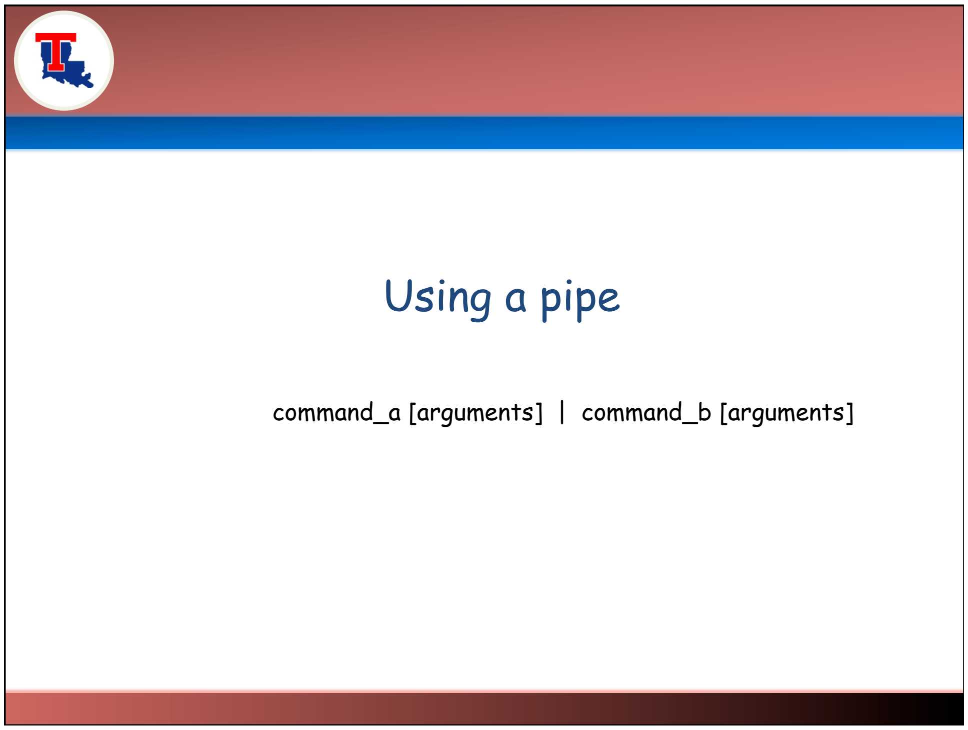


```
12       doz     black pens
2000     sheets  letterhead
csemac-stu:~ debryro$ cat < supplies
1        box     masking tape
2        boxes   scotch tape
12       doz     black pens
2000     sheets  letterhead
csemac-stu:~ debryro$
csemac-stu:~ debryro$
csemac-stu:~ debryro$ date > whoson
csemac-stu:~ debryro$ who >> whoson
csemac-stu:~ debryro$ cat whoson
Wed Dec  6 14:43:55 MST 2006
cseadmin console  Oct 20 16:45
debryro  ttyp1    Dec  6 13:13  (161.28.248.51)
10344587 ttyp2    Dec  6 14:30  (pc69.cse3.uvsc.e)
10344587 ttyp3    Dec  6 14:30  (pc69.cse3.uvsc.e)
cseadmin ttypd    Nov 14 20:12
10315707 ttyq4    Nov 12 21:29  (c-67-171-116-174)
csemac-stu:~ debryro$
```



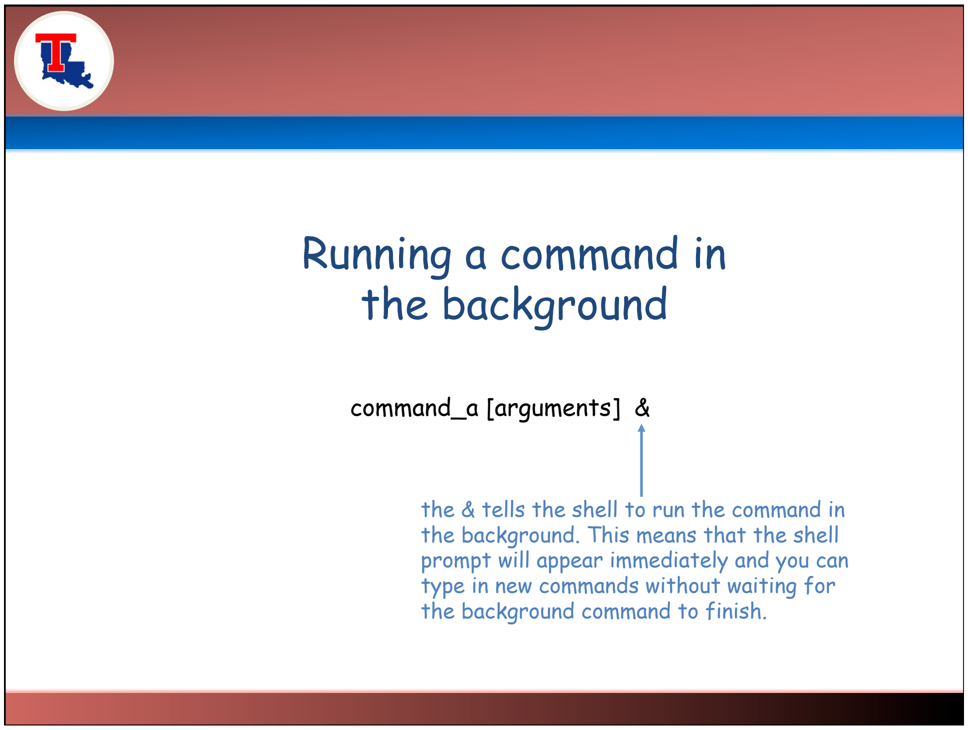
Pipe

- In a situation, when a user wants to issue more than one commands to perform certain tasks
- E.g. count how many file in a give directory.. Use a directory listing and line count commands
- “!” a special symbol
- a pipe to connect more than one commands and the output of one command to the input of another command.



Using a pipe

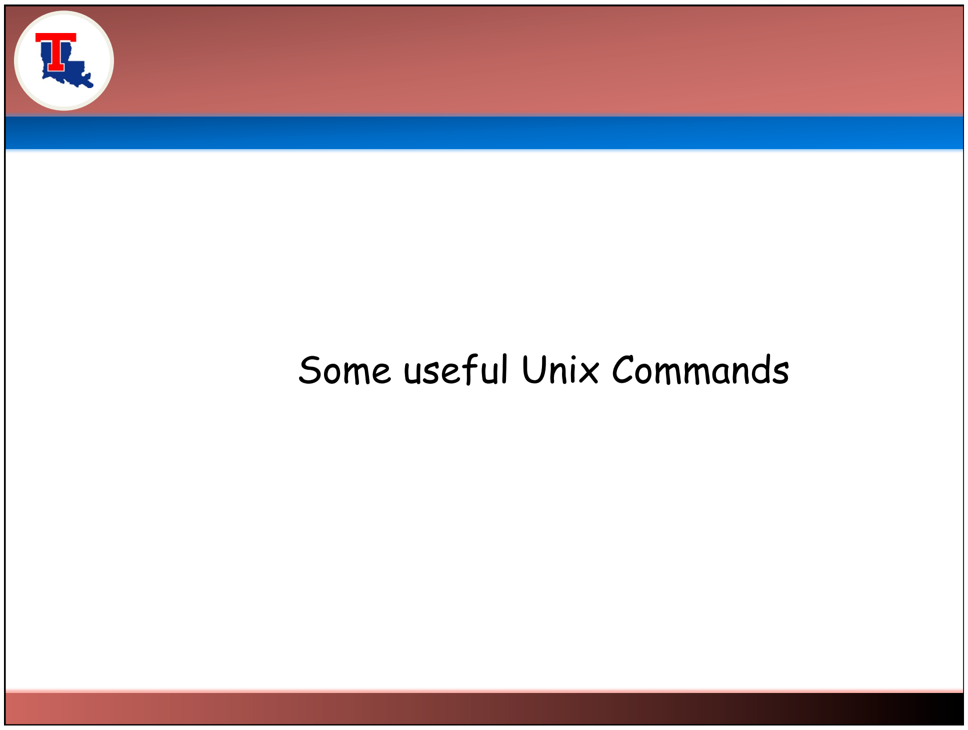
```
command_a [arguments] | command_b [arguments]
```



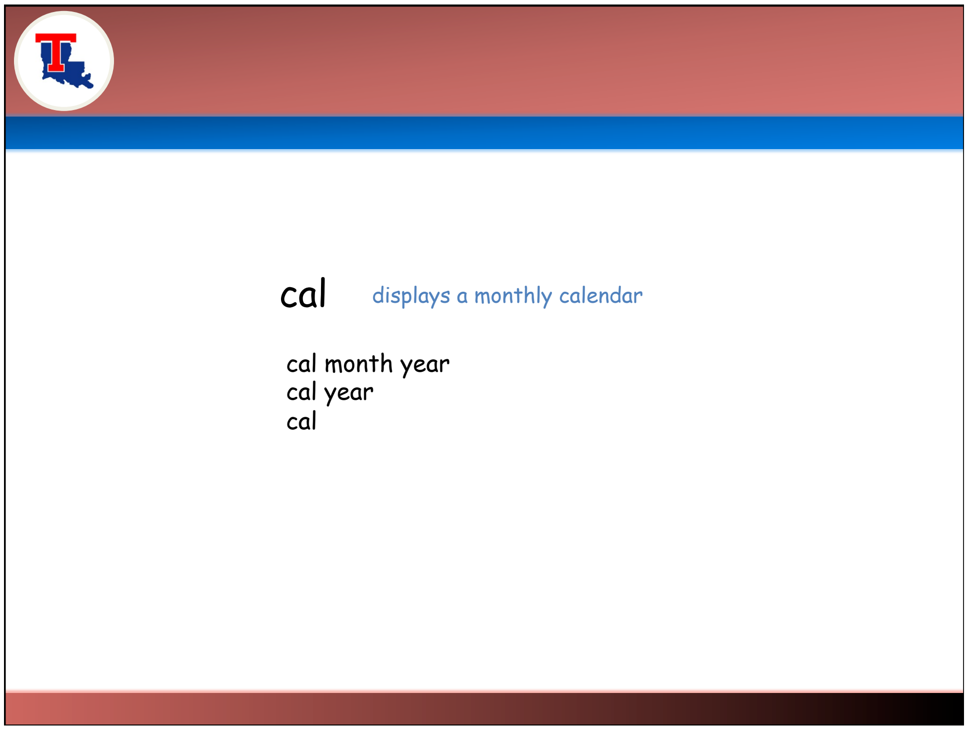
Running a command in the background

```
command_a [arguments]  &
```

the & tells the shell to run the command in the background. This means that the shell prompt will appear immediately and you can type in new commands without waiting for the background command to finish.



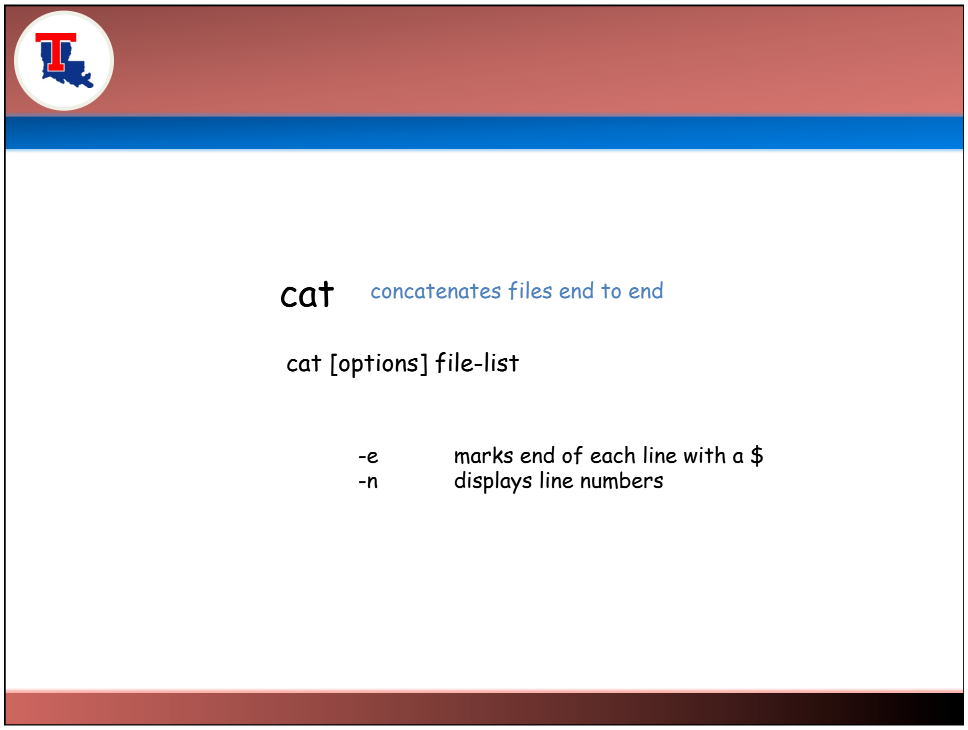
Some useful Unix Commands



cal

displays a monthly calendar

```
cal month year
cal year
cal
```

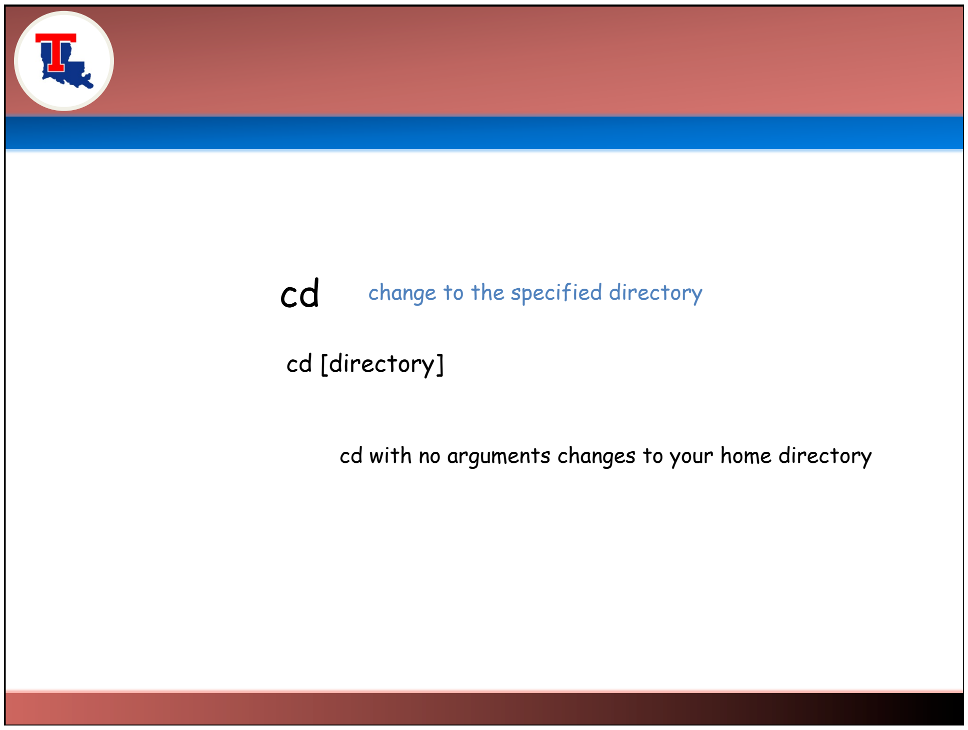


cat

concatenates files end to end

```
cat [options] file-list
```

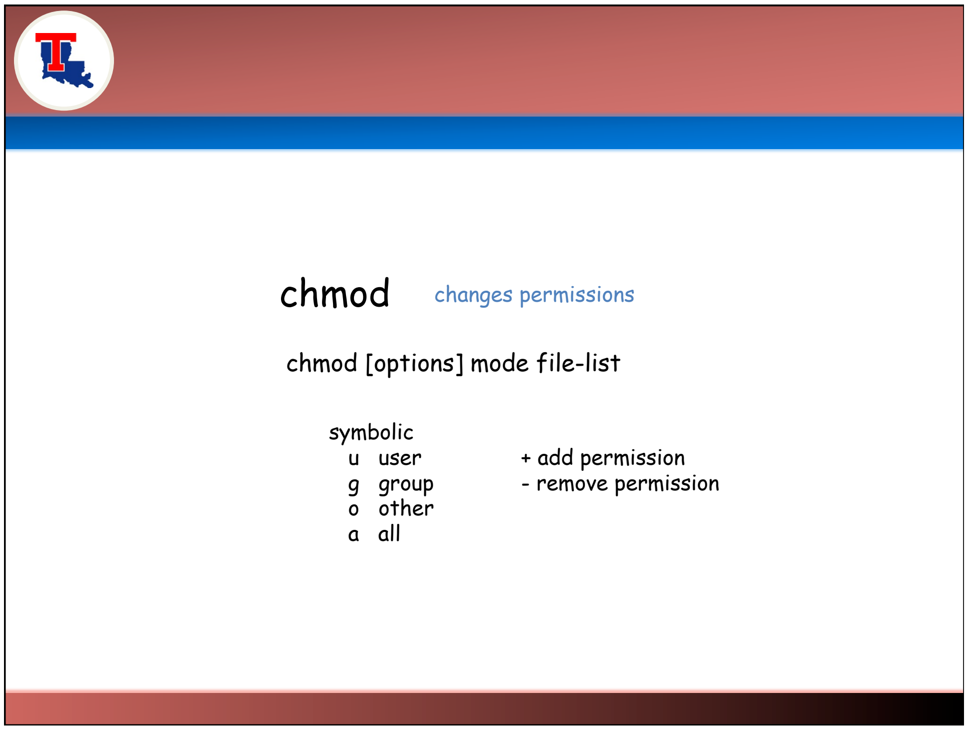
-e	marks end of each line with a $
-n	displays line numbers



cd change to the specified directory

cd [directory]

cd with no arguments changes to your home directory

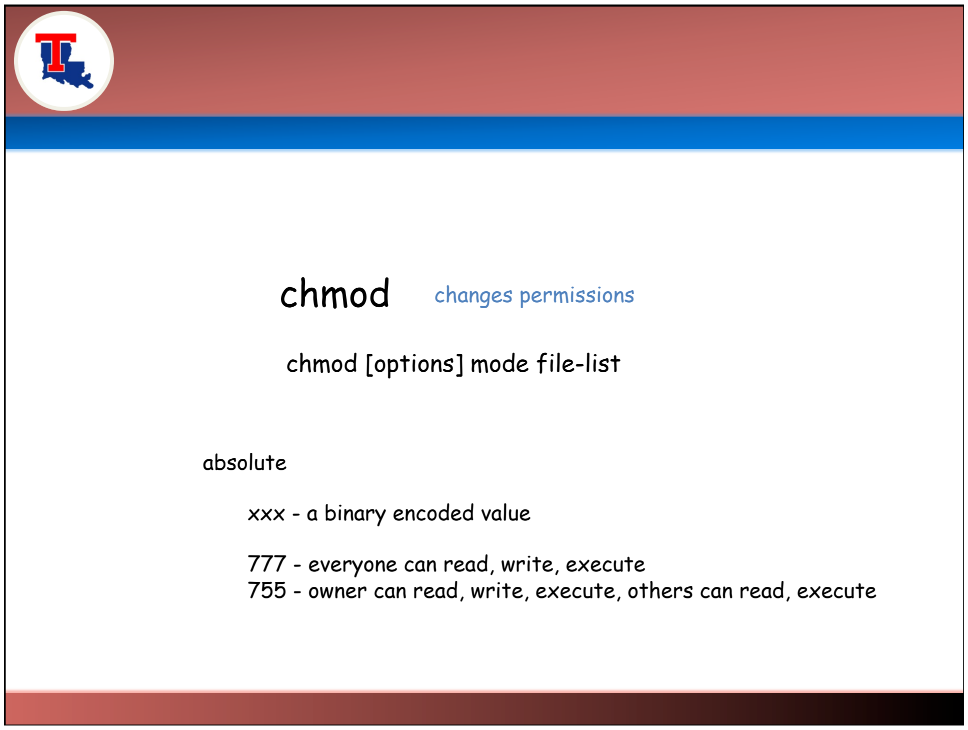


chmod changes permissions

chmod [options] mode file-list

symbolic

u	user	+ add permission
g	group	- remove permission
o	other	
a	all	



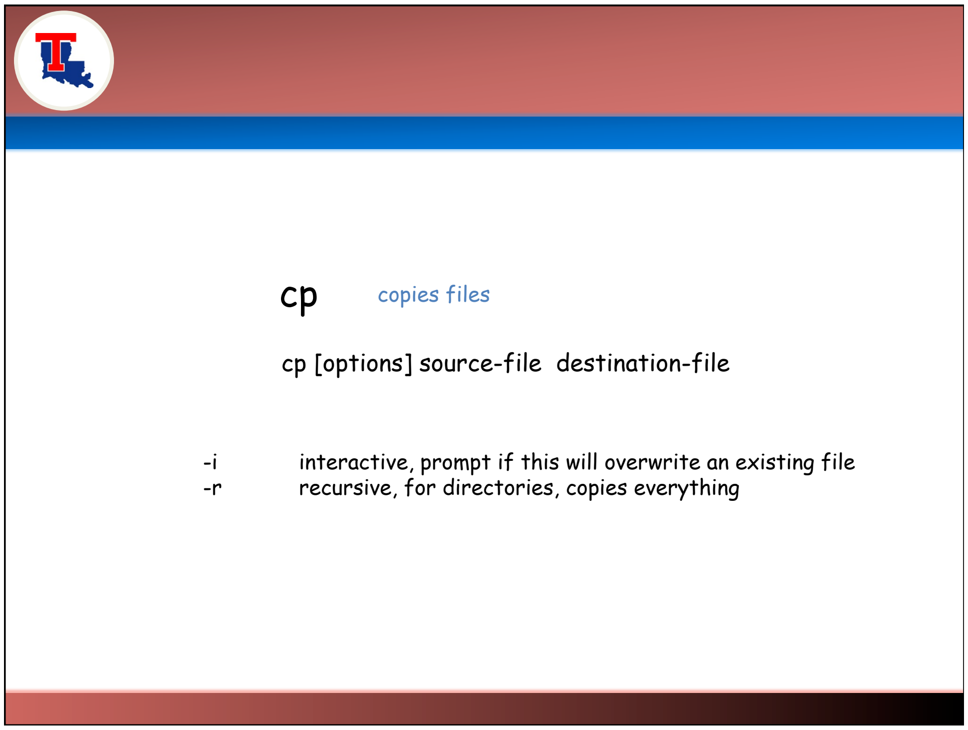
chmod changes permissions

chmod [options] mode file-list

absolute

xxx - a binary encoded value

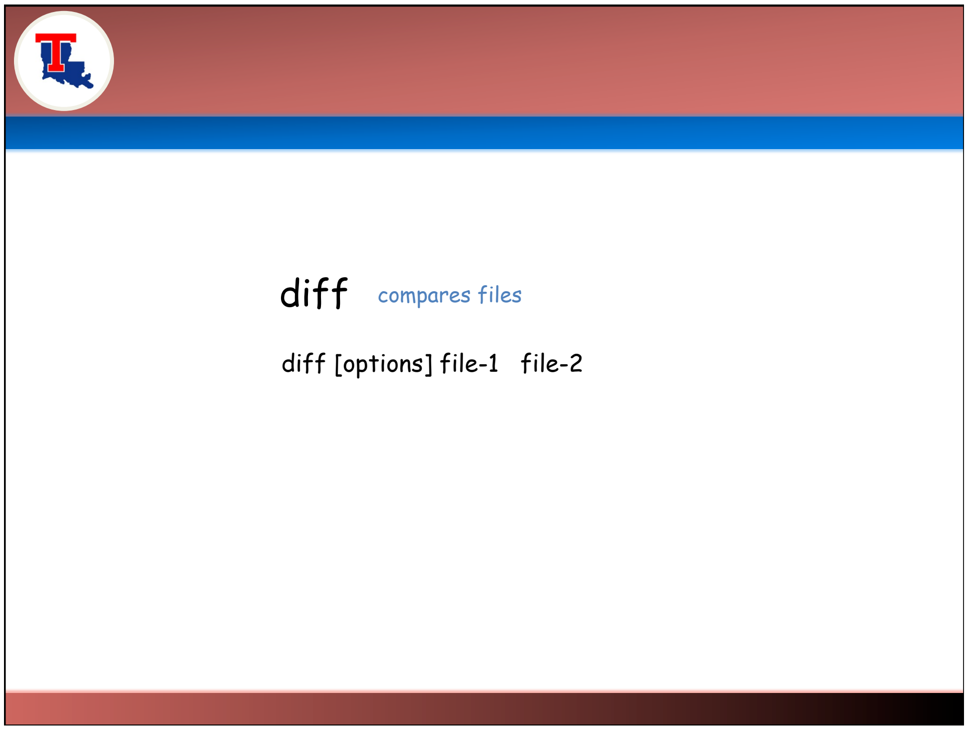
777 - everyone can read, write, execute
755 - owner can read, write, execute, others can read, execute



cp copies files

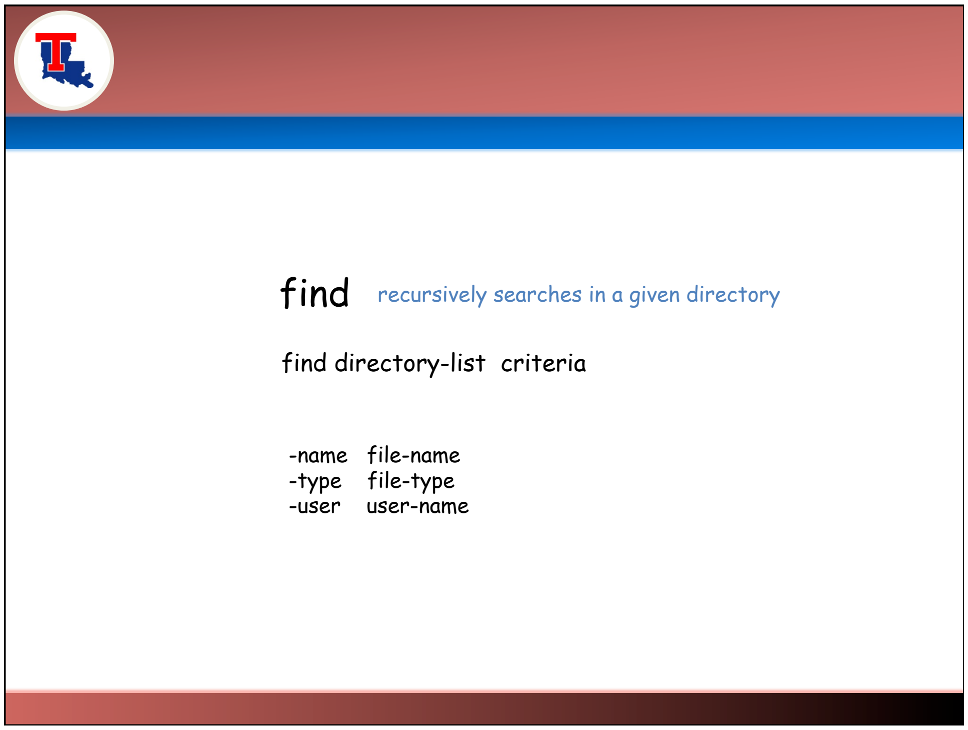
cp [options] source-file destination-file

-i	interactive, prompt if this will overwrite an existing file
-r	recursive, for directories, copies everything



diff compares files

```
diff [options] file-1   file-2
```



find recursively searches in a given directory

```
find directory-list  criteria

 -name   file-name
 -type   file-type
 -user   user-name
```

grep

grep searches files for a given pattern

grep [options] pattern [file-list]

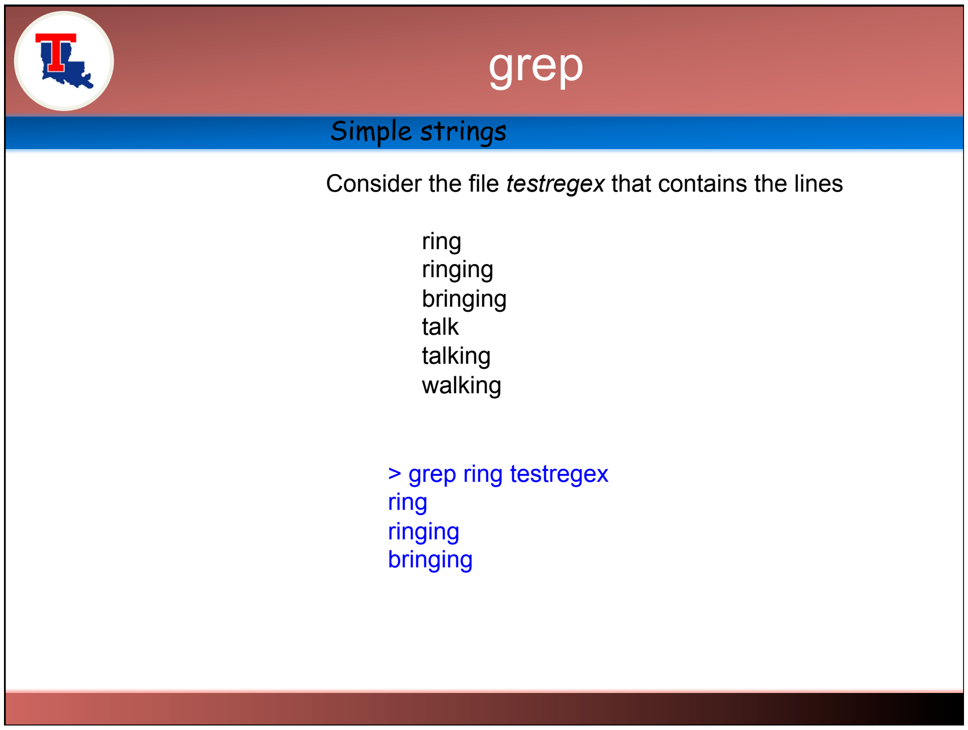
-c	display the number of lines that match
-i	ignore case
-l	display the filenames where a match is found
-n	displays each line with its line number in the file

grep

grep uses regular expressions in its pattern matching

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```



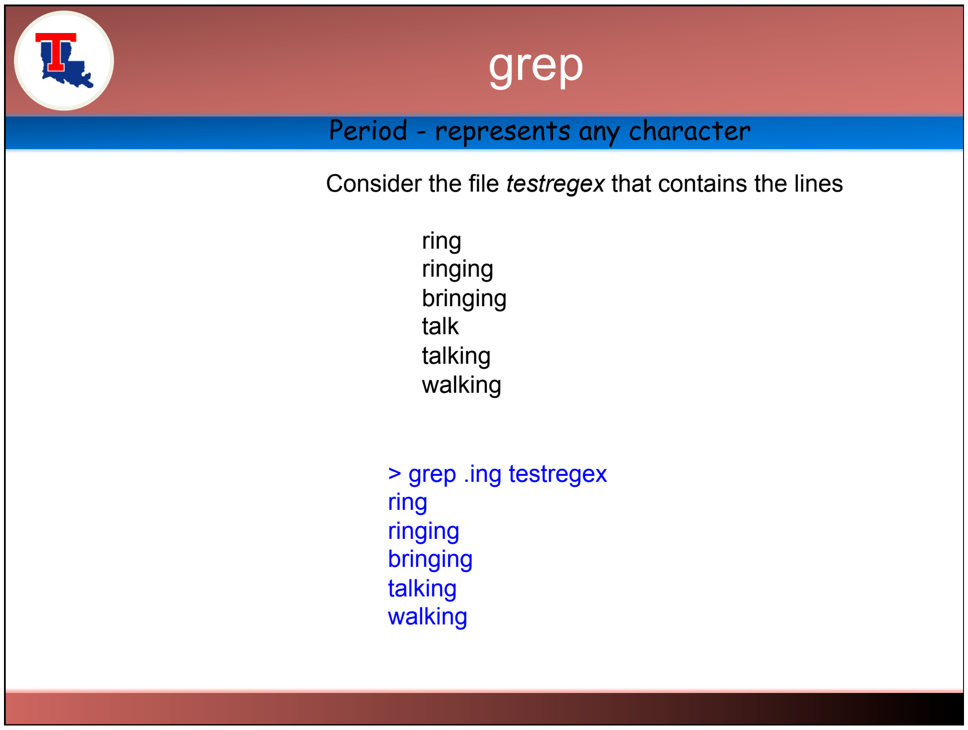
grep

Simple strings

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```

```
> grep ring testregex
ring
ringing
bringing
```



grep

Period - represents any character

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```

```
> grep .ing testregex
ring
ringing
bringing
talking
walking
```

grep

[ ] - represents a set of characters

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```

```
> grep [tw] testregex
talk
talking
walking
```

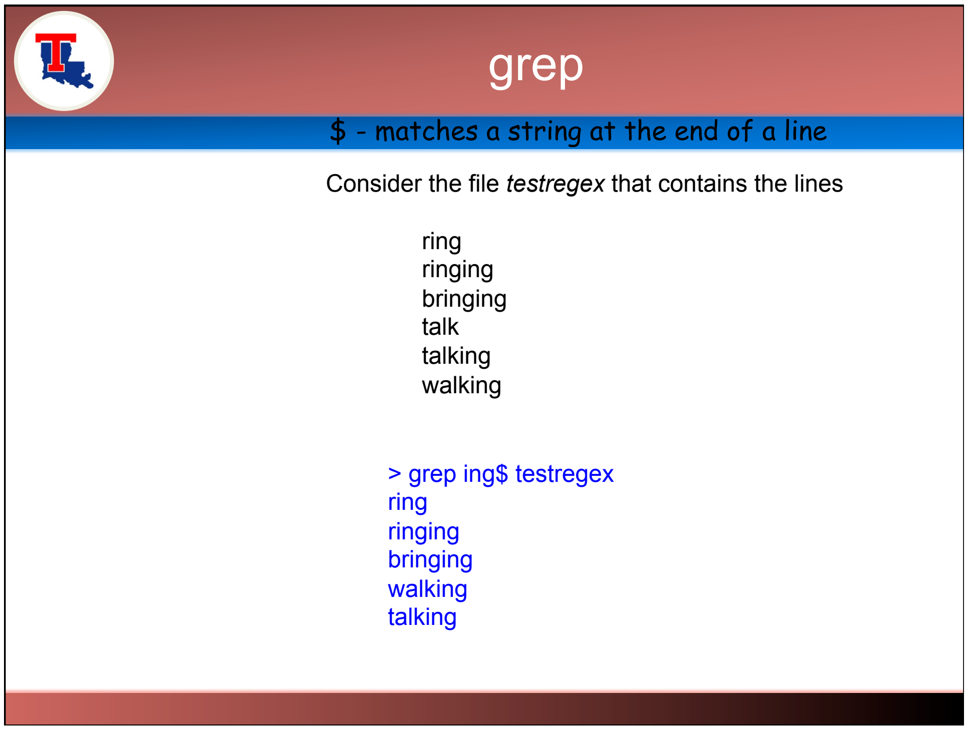
grep

^ - matches a string at the beginning of a line

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```

```
> grep ^ri testregex
ring
ringing
```



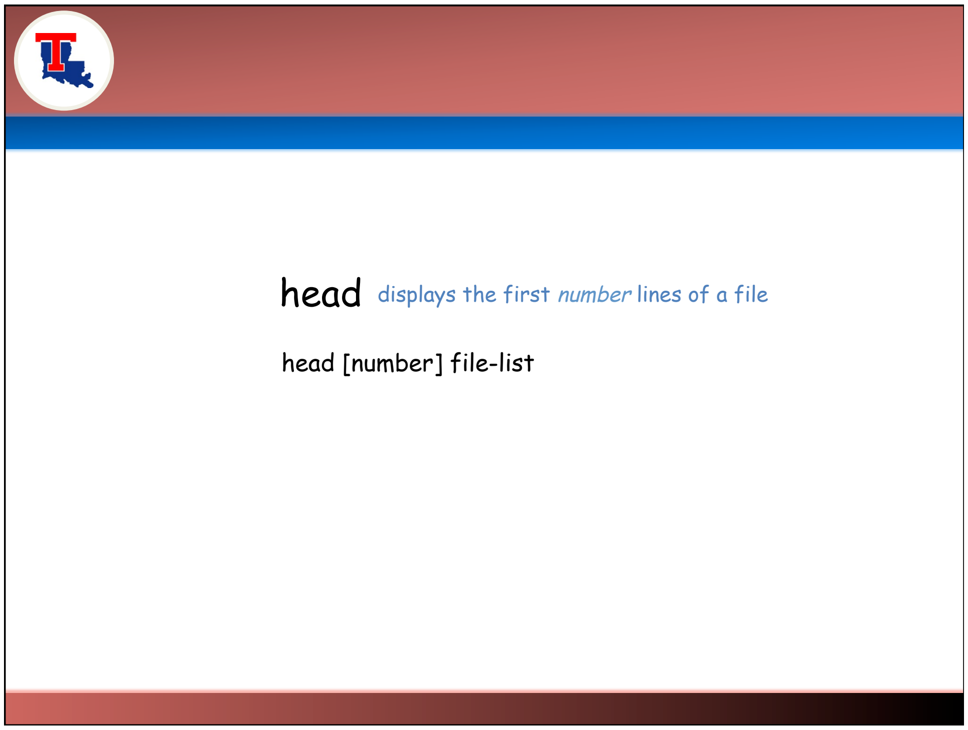
grep

$ - matches a string at the end of a line

Consider the file *testregex* that contains the lines

```
ring
ringing
bringing
talk
talking
walking
```

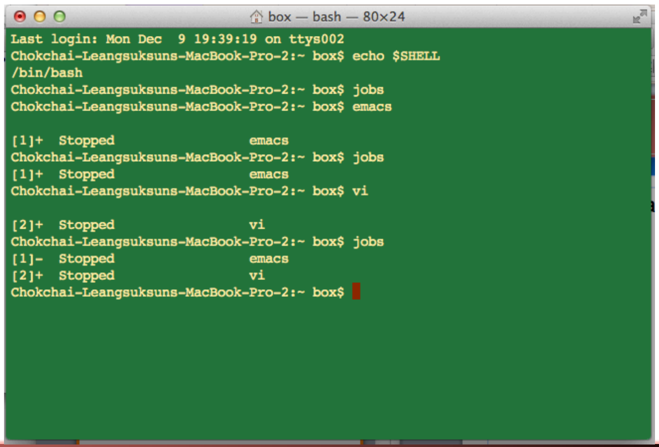
```
> grep ing$ testregex
ring
ringing
bringing
walking
talking
```



head displays the first *number* lines of a file

head [number] file-list

- job – show what are all processes running at the current shell



```
Last login: Mon Dec  9 19:39:19 on ttys002
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ echo $SHELL
/bin/bash
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ jobs
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ emacs

[1]+  Stopped                 emacs
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ jobs
[1]+  Stopped                 emacs
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ vi

[2]+  Stopped                 vi
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$ jobs
[1]-  Stopped                 emacs
[2]+  Stopped                 vi
Chokchai-Leangsuksuns-MacBook-Pro-2:~ box$
```

kill   kills a process

kill [signal-number] PID-list

ln

create a link to a file

ln [option] existing-file link-name

by default ln creates a hard link. Hard links must be
in the same directory as the file being linked to.

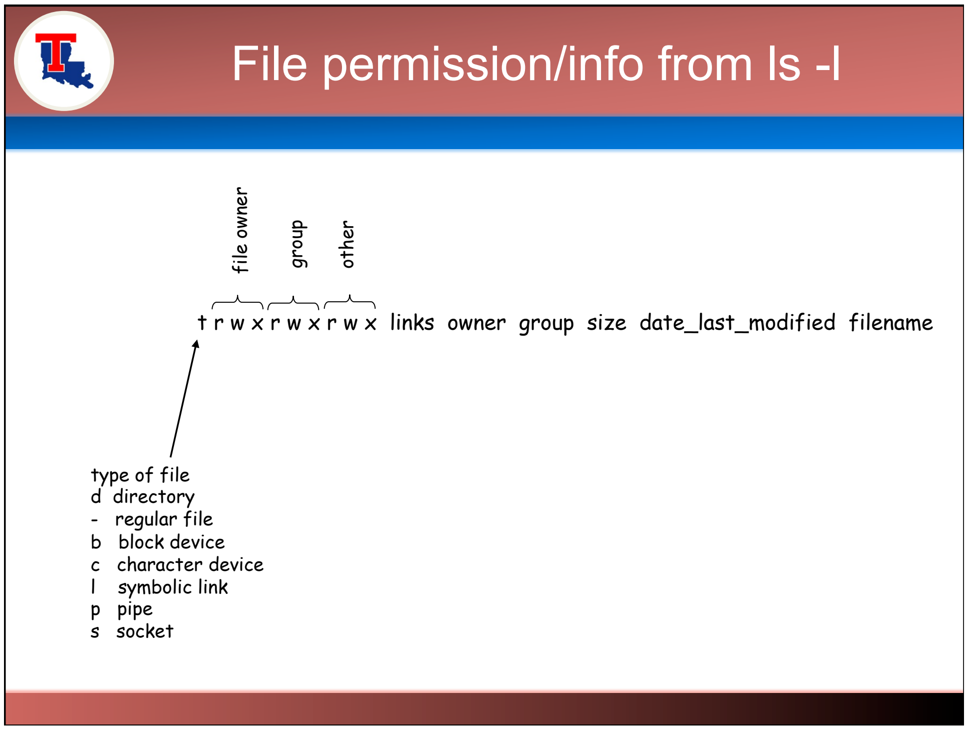
the -s option creates a symbolic link. Symbolic links can
be across file systems and directories.

ls

list information about one or more files

ls [options] file-list

-a	list all entries, including invisible files
-l	show all file information
-r	shows all subdirectories

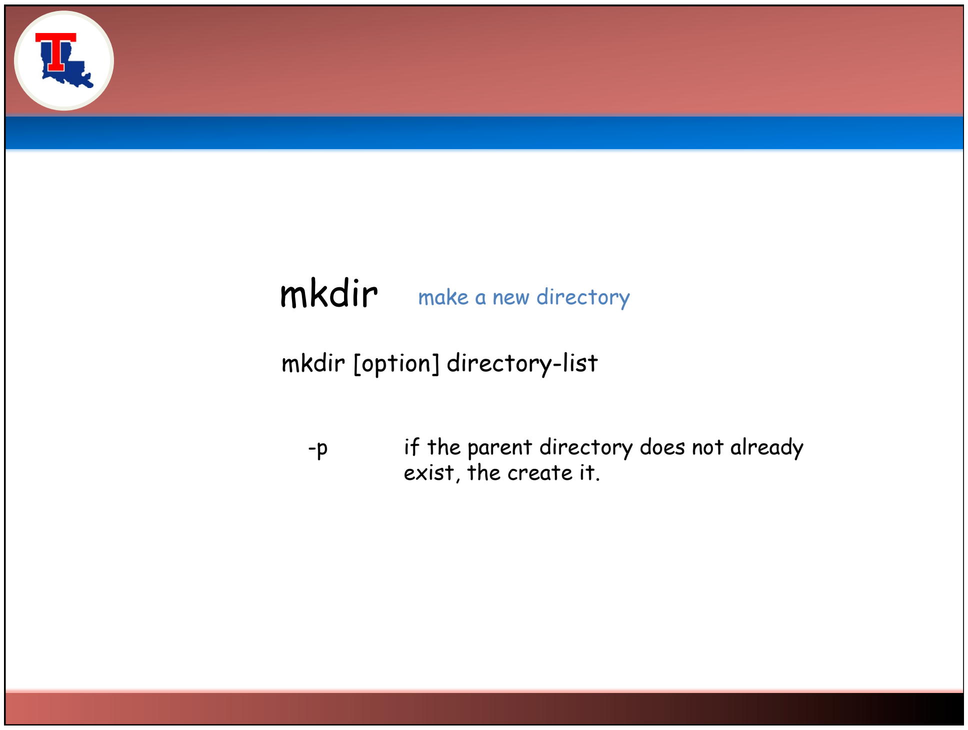


File permission/info from ls -l

$t\ \overbrace{r\ w\ x}^{\text{file owner}}\ \overbrace{r\ w\ x}^{\text{group}}\ \overbrace{r\ w\ x}^{\text{other}}$ links owner group size date_last_modified filename

type of file

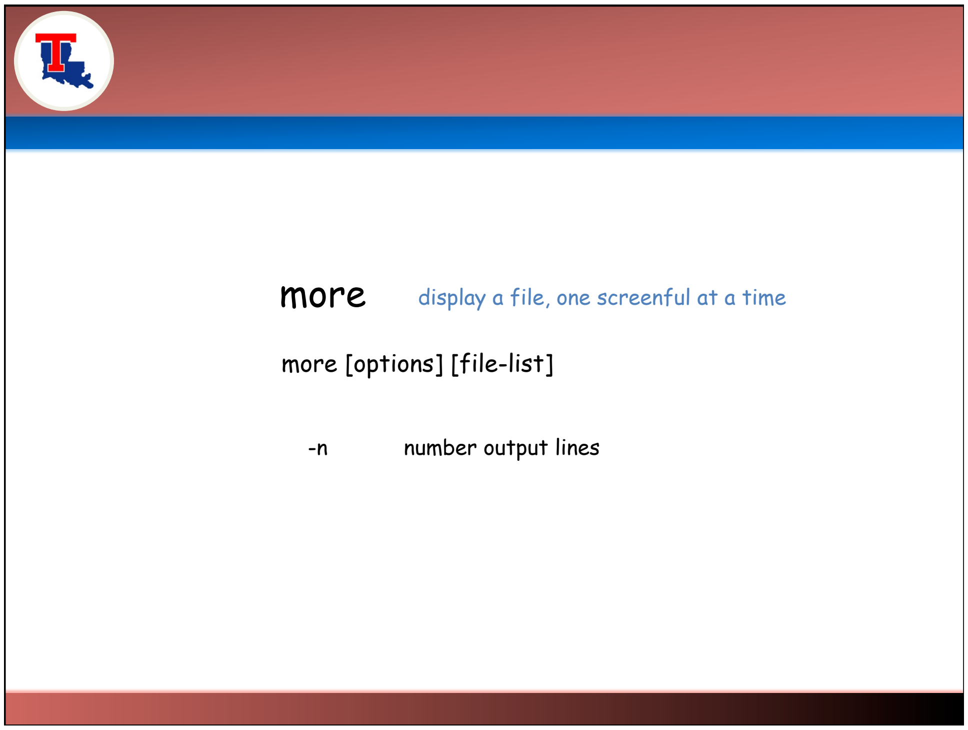
d	directory
-	regular file
b	block device
c	character device
l	symbolic link
p	pipe
s	socket



mkdir make a new directory

mkdir [option] directory-list

-p	if the parent directory does not already exist, the create it.

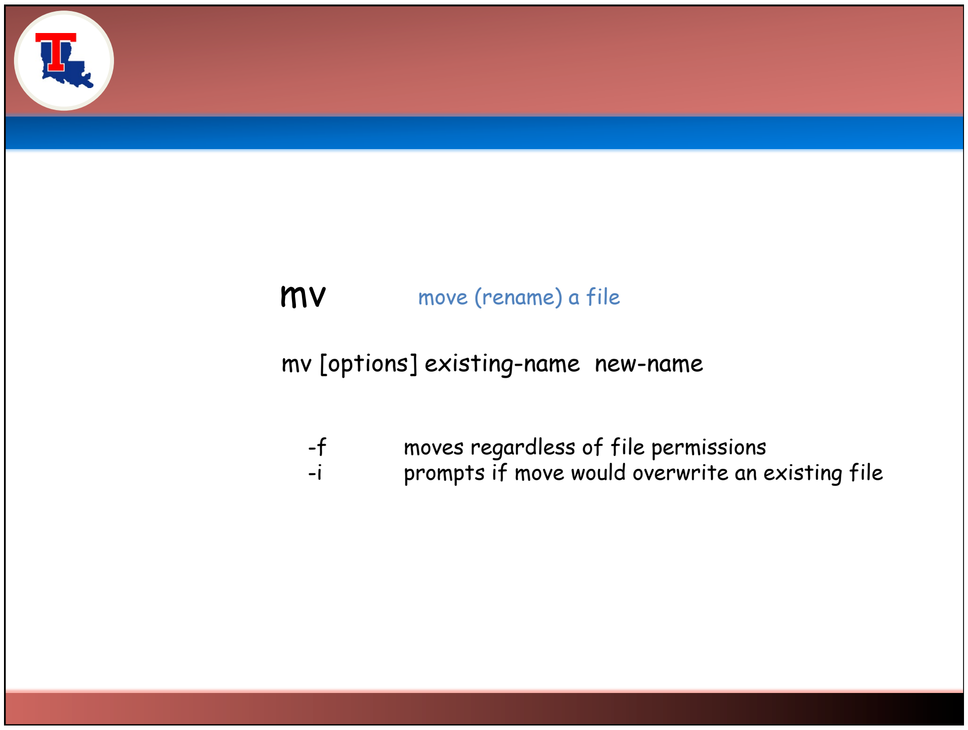


more

display a file, one screenful at a time

more [options] [file-list]

-n	number output lines

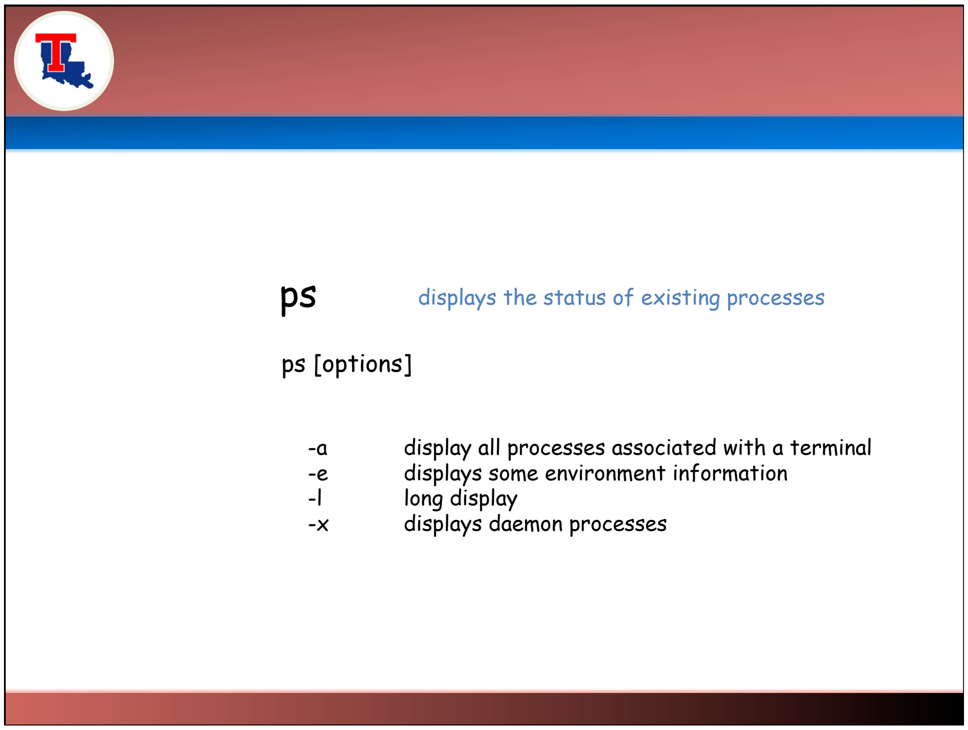


mv

move (rename) a file

mv [options] existing-name new-name

-f	moves regardless of file permissions
-i	prompts if move would overwrite an existing file

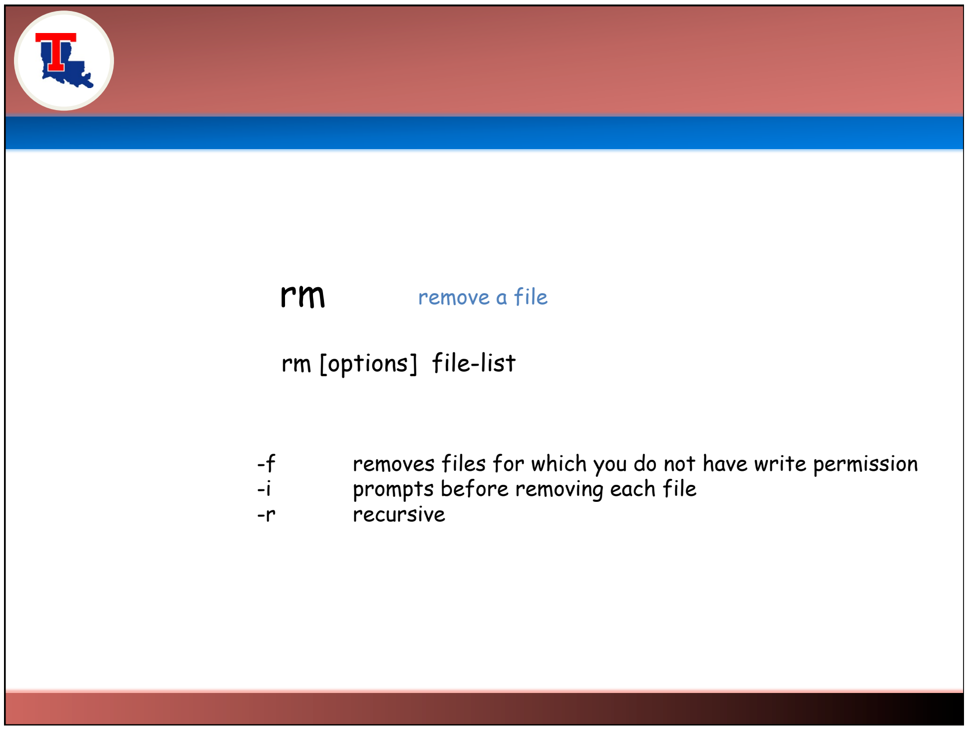


ps

displays the status of existing processes

```
ps [options]
```

-a	display all processes associated with a terminal
-e	displays some environment information
-l	long display
-x	displays daemon processes

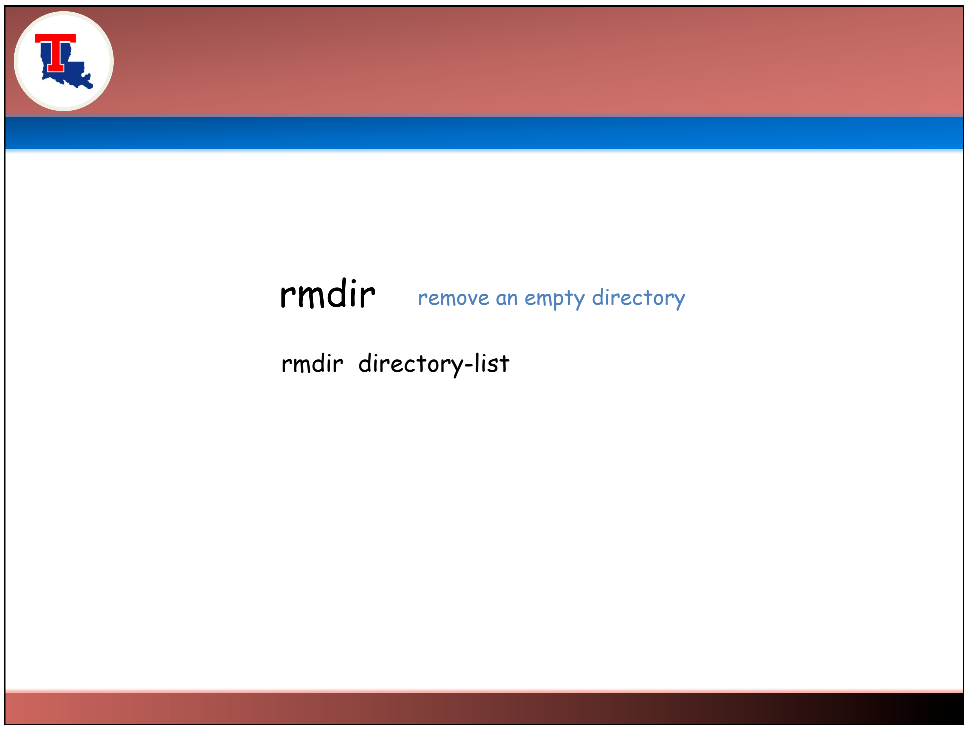


rm

remove a file

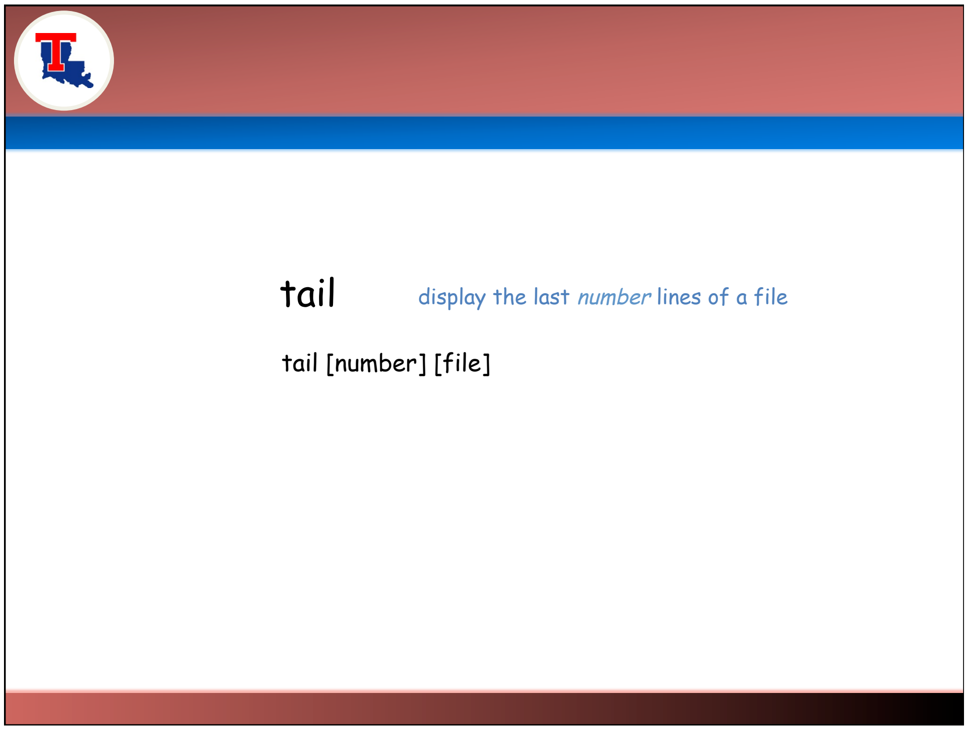
```
rm [options]  file-list
```

-f	removes files for which you do not have write permission
-i	prompts before removing each file
-r	recursive



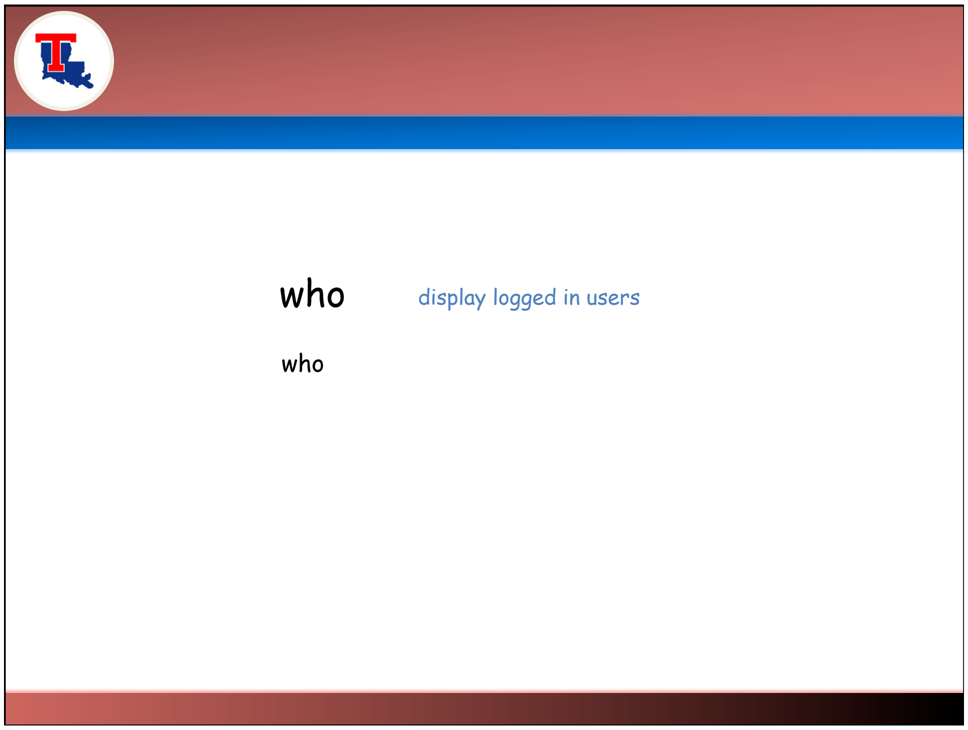
rmdir remove an empty directory

rmdir directory-list



tail display the last *number* lines of a file

tail [number] [file]



who

display logged in users

```
who
```