EECS2031 Software Tools

F 2020-21

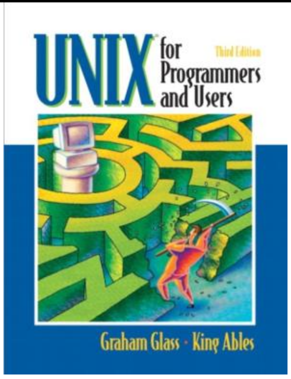
Nov 19, 2021 Lecture 20

1

Contents

- Overview of UNIX
  - Structures
  - File systems
    - Pathname: absolute vs relative
    - Security `-rwx-rw--x`
  - Process:
    - Exit code ≥ 0
    - IPC: Pipes who | sort

Previous lecture



….
- Utilities/commands
  - Basic: pwd, ls, rmdir, mkdir, cat, more, mv, cp, rm, wc, chmod
  - Advanced: grep/egrep, sort, find ….
- Shell and shell scripting language

3

3

Basic utilities/commands

Introduces the following command-line utilities, listed in alphabetical order:

cancel	head	mv
cat	less	newgrp
chgrp	lp	page
chmod	lpr	passwd
chown	lprm	pwd
clear	lpq	rm
cp	lpstat	rmdir
date	ls	stty
file	mail	tail
groups	man	tset
	mkdir	wc
	more	

22

Basic utilities/commands

Introduces the following utilities, listed in groups:

General	Directory	File	File print
man	pwd	cat	lp
clear	ls	more less	lpr
echo	cd	head tail	lprm
date	mkdir	cp	lpq
cal	rmdir	mv	lpstat
		rm	
		wc	
		file	
		chmod	
		chgrp	
		newgrp	
		chown	

23

Running a utility/command

- To run a utility, simply enter its name at the prompt and press the Enter key.

  - Up/down arrow for history
  - Tab key for auto complete

```
$ date          # run the utility.
Sun Jul 14 20:10:42 EDT 2019

$ cal 7 2017
```

24

24

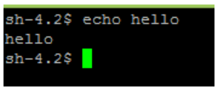
clear

- clears your screen

cls in DOS

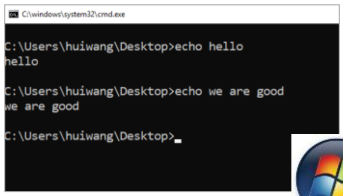
echo

- print statement in UNIX/LINUX
- **`echo $?`**
- **`echo hello`**



```
sh-4.2$ echo hello
hello
sh-4.2$ █
```

Same in DOS



```
C:\Users\huiwang\Desktop>echo hello
hello

C:\Users\huiwang\Desktop>echo we are good
we are good

C:\Users\huiwang\Desktop>_
```

25

25

man: online help

- All UNIX systems have a utility called man (short for manual page) that puts this information at your fingertips.

- The manual pages are on-line copies of the original UNIX documentation, which is usually divided into eight sections. They contain information about utilities, system calls, file formats, and shells.

  man [section] word
  man -k keyword

- The first usage of man displays the manual entry associated with word. If no section number is specified, the first entry that it finds is displayed.

- The second usage of man displays a list of all the manual entries that contain keyword.

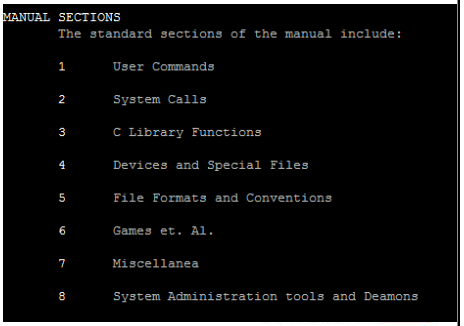
Organization of the manual pages

- The typical division of topics in manual pages (sections) is as follows:

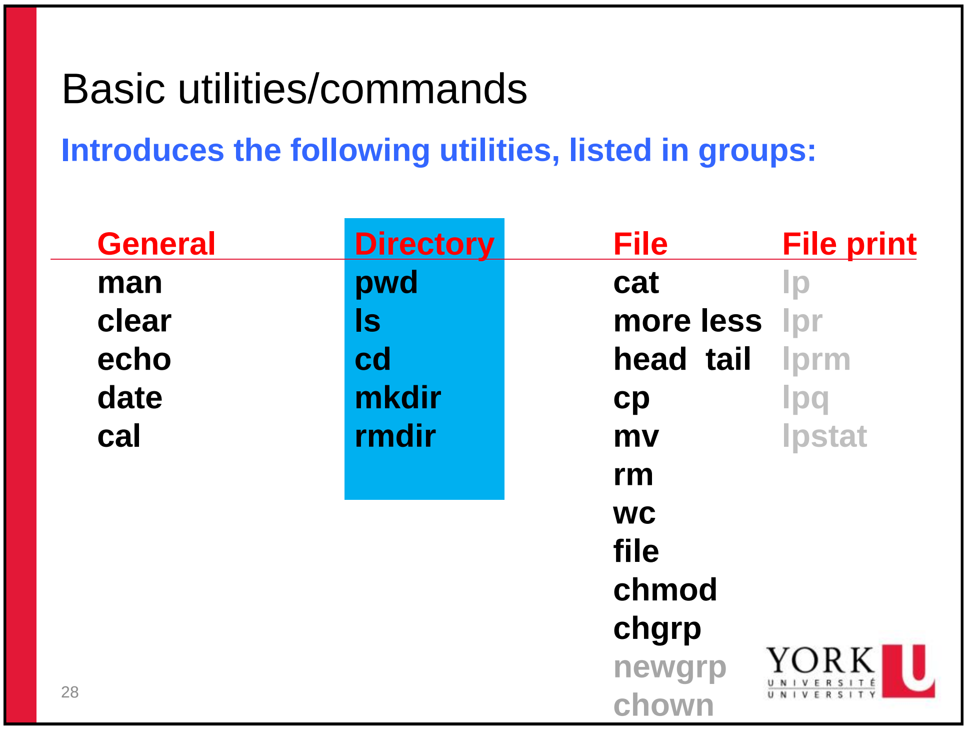
1. Commands and Application Programs.
2. System Calls
3. C Library Functions
4. Special Files
5. File Formats
6. Games
7. Miscellaneous
8. System Administration Utilities

```
$ man 1 date
$ man 3 strlen
```

```
% man man
```



```
MANUAL SECTIONS
       The standard sections of the manual include:

       1       User Commands

       2       System Calls

       3       C Library Functions

       4       Devices and Special Files

       5       File Formats and Conventions

       6       Games et. Al.

       7       Miscellanea

       8       System Administration tools and Deamons
```

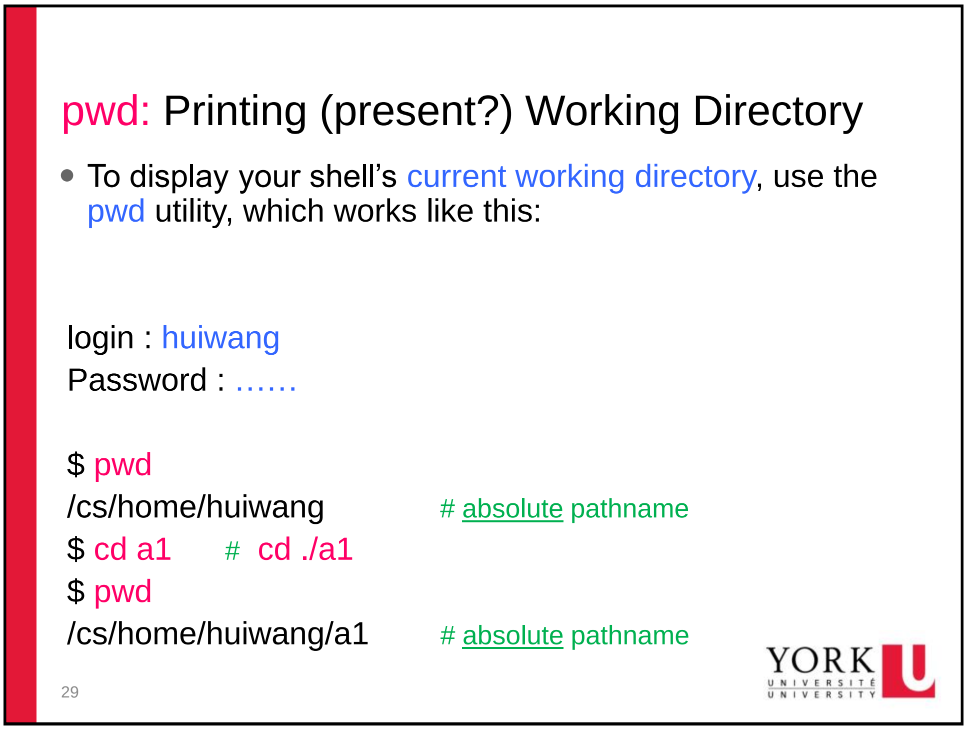

Basic utilities/commands

Introduces the following utilities, listed in groups:

General	Directory	File	File print
man	pwd	cat	lp
clear	ls	more less	lpr
echo	cd	head tail	lprm
date	mkdir	cp	lpq
cal	rmdir	mv	lpstat
		rm	
		wc	
		file	
		chmod	
		chgrp	
		newgrp	
		chown	

28

28


pwd: Printing (present?) Working Directory

- To display your shell's current working directory, use the pwd utility, which works like this:

```
login : huiwang
Password : ……

$ pwd
/cs/home/huiwang            # absolute pathname
$ cd a1      #  cd ./a1
$ pwd
/cs/home/huiwang/a1         # absolute pathname
```

29

29

Listing Contents of a Directory: ls

dir in DOS

ls -adglsFR { fileName }* {directoryName}*

- ls lists all of the files and sub-directories in the current working directory in alphabetical order, excluding files whose names start with a period.
- The -a option causes . .. .file (hidden) to be included in the listing.
- The -l option generates a long listing, including permission flags, the file's owner, and the last modification time.
- The -d option causes the details of the directories themselves to be listed, rather than their contents.
- The -S option sorts the list on the size of entries (largest first).
- The -t option sorts the list on the modification time of entries (newest first)
- The -r reverse the order of sorting

30

Directory Listing, an example

```
$ ls a5            # ls ./a5   list all files in a5, a subdir of current directory.
 a.out               arrayAddressPPP.c   inputE2.txt
 arithmetic2017.c    inputA.txt

$ ls -l a5         # long listing of contents of a5
-rwx------ 1 huiwang faculty 7315 Mar 16 23:27 a.out
-rw------- 1 huiwang faculty  2210 Feb 20 15:40 arithmetic2017.c
-rw------- 1 huiwang faculty  1079 Feb 20 14:03 arrayAddressPPP.c
-rw------- 1 huiwang faculty     62  Feb 23 17:29 inputA.txt
-rw------- 1 huiwang faculty     50  Feb 23 19:56 inputE2.txt

$ ls -ld a5        # long listing of directory a5 itself
drwxr--r-- 2 huiwang faculty 4096 Mar 16 23:27 a5
```

```
$ ls a5 -lSr   ?    # sort by size. same as ls -l -S -r   or  ls -lS -r
$ ls lab5 -lt  -r   # sort by time. Who submitted lab5 in first/last minute?
```

31

File Listing

- Here's an example of the use of ls on files:

```
$ ls                    # ls .   ls ./      list all files in current directory.
a.out   heart.txt
$ ls heart.txt
heart.txt
$ ls  -l  heart.txt     # ls -l ./heart.txt     long listing of "heart.txt"
-rw-r--r--   1   huiwang   faculty  106   Jan  30  19:46   heart.txt
$
```

name of the file
time that the file was last modified
size of the file, in bytes
group of the owner of the file
username of the owner of the file
hard-link count of the file
type and permission mode of the file

Changing Directories: cd

Same in DOS

cd [directoryName]

absolute or relative path

- The following might be inconvenient; especially if we deal with large hierarchy:
```
$ cat   lyrics/heart.txt                # cat  ./lyrics/heart.txt
```

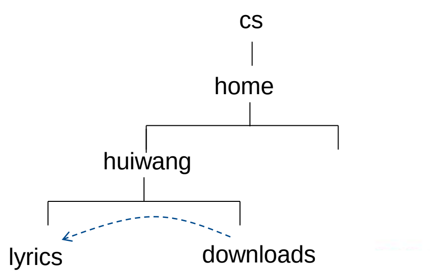
- Instead, change directory:
```
$ cd  lyrics                            # cd  ./lyrics
$ cat  heart.txt                        # cat ./heart.txt
```

- The cd shell command changes a shell's current working directory to be directoryName.

- If the directoryName argument is omitted, the shell is moved to its owner's home directory.
```
$ cd       #  $ cd ~
```

Traversing Directories example

```
$ cd /cs/home/huiwang            # absolute   or just cd
$ pwd                            # display where I am
/cs/home/huiwang
$ cd lyrics                      # ./lyrics move into the "lyrics" directory
$ pwd
/cs/home/huiwang/lyrics
$ cd ..                          # move up one level, back to huiwang
$ pwd                            # display new position
/cs/home/huiwang
$ cd downloads        # ./downloads
$ pwd
/cs/home/huiwang/downloads
$ cd ../lyrics
$ pwd
/cs/home/huiwang/lyrics
$ cd  ../../
$ _
```



35

35

Creating Directory: mkdir

Same in DOS

mkdir   -p newDirectoryName

- The mkdir utility creates a directory. The -p option creates any parent directories in the newDirectoryName pathname that do not already exist.
- If newDirectoryName already exists, an error message is displayed and the existing file is not altered

```
$ mkdir lyrics              # creates a directory called "lyrics".

$ ls -l                     # check the directory listing to confirm
- rw-r-- r--     1    huiwang faculty    106    Jan  30  23:28     heart.ver1
drwxr-xr-x       2    huiwang faculty    512    Jan  31  19:49     lyrics/
```

```
$ mkdir  -p study/programming/C/2021S/2031/lab2/
```

Error without p, unless study/../../2031 exists

36

36

Deleting a Directory: rmdir

Same in DOS

rmdir { directoryName }+

- The rmdir utility removes all of the directories in the list of directory names provided in the command.
  - A directory must be empty before it can be removed.

```
$ rmdir  lab5
  rmdir: lab5 : Directory not empty.
$ _
```

- To (recursively) remove a directory and all of its contents, use the rm utility with the -r option instead

```
$ rm -r  lab5
```

Later today

37

YORK U UNIVERSITÉ UNIVERSITY

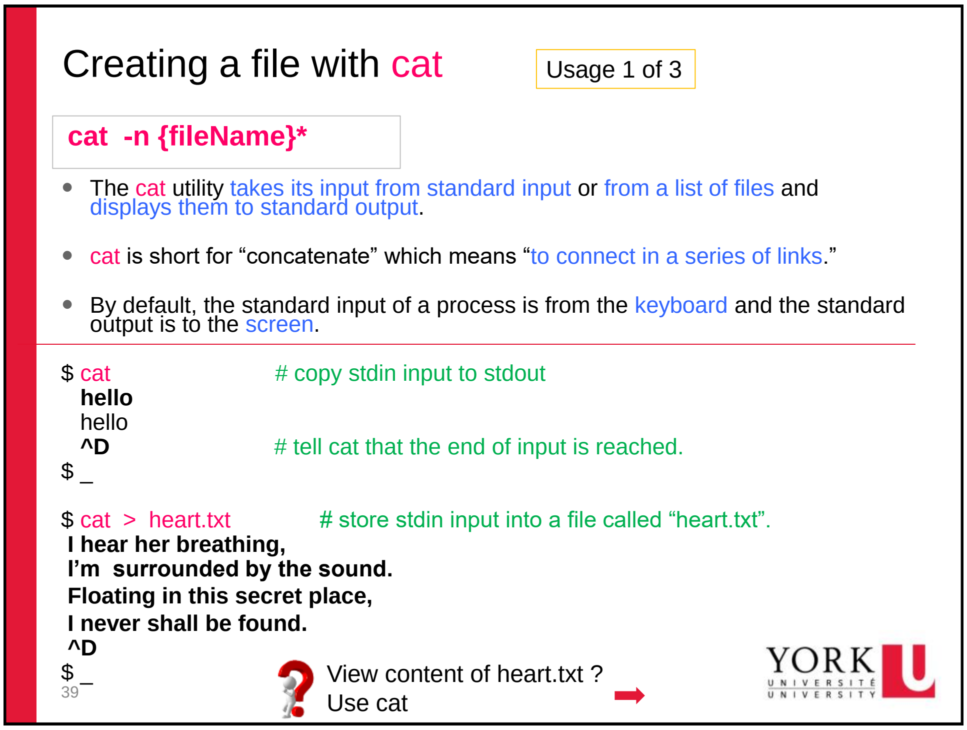
Basic utilities

Introduces the following utilities, listed in alphabetical order:

General	Directory	File	Print File
man	pwd	cat	lp
clear	ls	more less	lpr
echo	cd	head tail	lprm
date	mkdir	cp	lpq
cal	rmdir	mv	lpstat
		rm	
		wc	
		file	
		chmod	
		chgrp	
		newgrp	
		chown	

38

YORK U UNIVERSITÉ UNIVERSITY



Creating a file with cat

Usage 1 of 3

cat -n {fileName}*

- The cat utility takes its input from standard input or from a list of files and displays them to standard output.
- cat is short for “concatenate” which means “to connect in a series of links.”
- By default, the standard input of a process is from the keyboard and the standard output is to the screen.

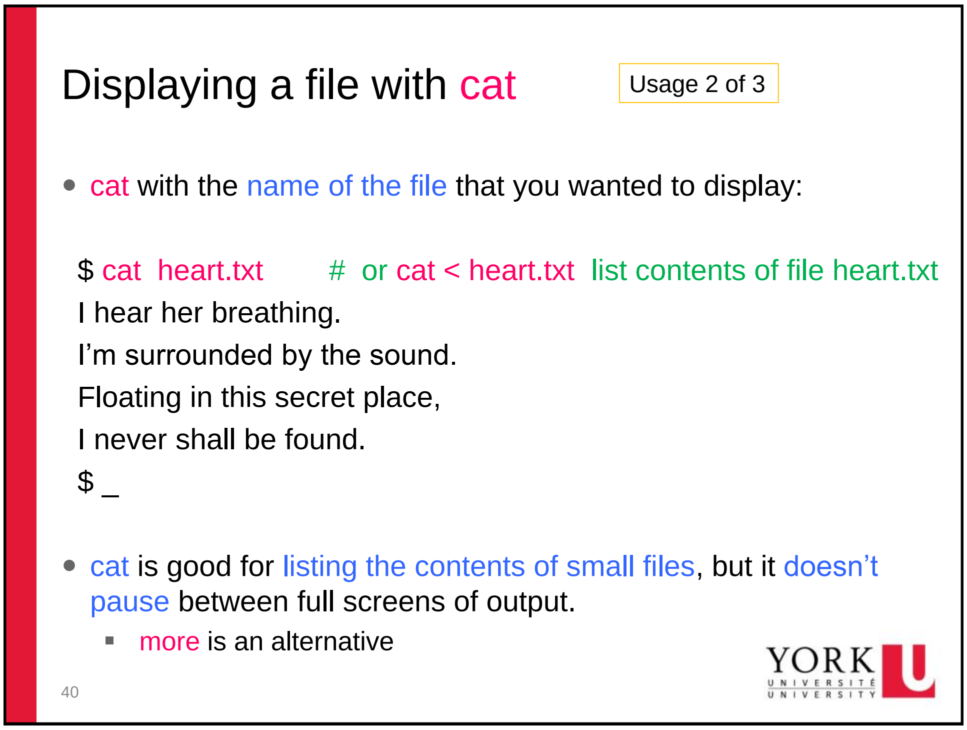
```
$ cat                    # copy stdin input to stdout
  hello
  hello
  ^D                     # tell cat that the end of input is reached.
$ _

$ cat  >  heart.txt          # store stdin input into a file called “heart.txt”.
 I hear her breathing,
 I’m  surrounded by the sound.
 Floating in this secret place,
 I never shall be found.
 ^D
$ _
```

View content of heart.txt ?
Use cat

YORK UNIVERSITÉ UNIVERSITY

39

39



Displaying a file with cat

Usage 2 of 3

- cat with the name of the file that you wanted to display:

```
$ cat  heart.txt        #  or cat < heart.txt  list contents of file heart.txt
I hear her breathing.
I’m surrounded by the sound.
Floating in this secret place,
I never shall be found.
$ _
```

- cat is good for listing the contents of small files, but it doesn’t pause between full screens of output.
  - more is an alternative

YORK UNIVERSITÉ UNIVERSITY

40

40

Concatenate files with cat

Usage 3 of 3

- cat with the name of the files that you wanted to concatenate (display together):

```
$ cat  heart.txt   heart2.txt          #  list the contents of both the files.
I  hear her breathing.
I’m  surrounded by the sound.
Floating in this secret place,             heart.txt
I never shall be found.
This is my heart
beating..                                  heart2.txt
$ _
```

- usually use redirection to create a new file concatenating contents of both the input files

```
$ cat  heart.txt heart2.txt  > heartNew.txt
```

YORK UNIVERSITÉ UNIVERSITY

41

41

Displaying a file: more and less

more in DOS

more -f [+lineNumber] { fileName }*

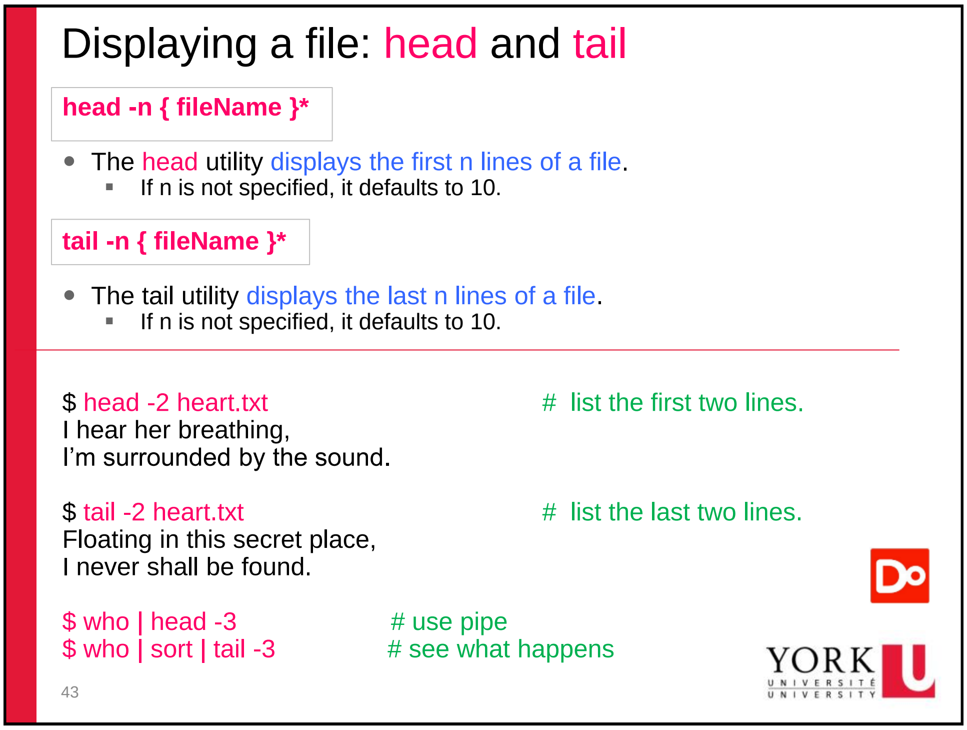
- The more utility allows you to scroll a list of files, one page at a time.
- The less utility is similar
- After each page is displayed, displays the message “--more– x%” to indicate that it’s waiting for a command.
  - To display the next page, press the space bar.
  - To display the next line, press the Enter key.
  - To display the previous page, press ^b or just b.   Not working if piped.
  - To quit from more, press the “q” key.

```
$ ls -l /usr/bin > myLongFile          # myLongFile is a long file
$ more myLongFile

$ cat myLongFile | more
$ ls -l /usr/bin  |  more         # use pipe
$ ls -l /usr/bin  |  less         # use pipe, backward still working
```

YORK UNIVERSITÉ UNIVERSITY

42

42



Displaying a file: head and tail

head -n { fileName }*

- The head utility displays the first n lines of a file.
  - If n is not specified, it defaults to 10.

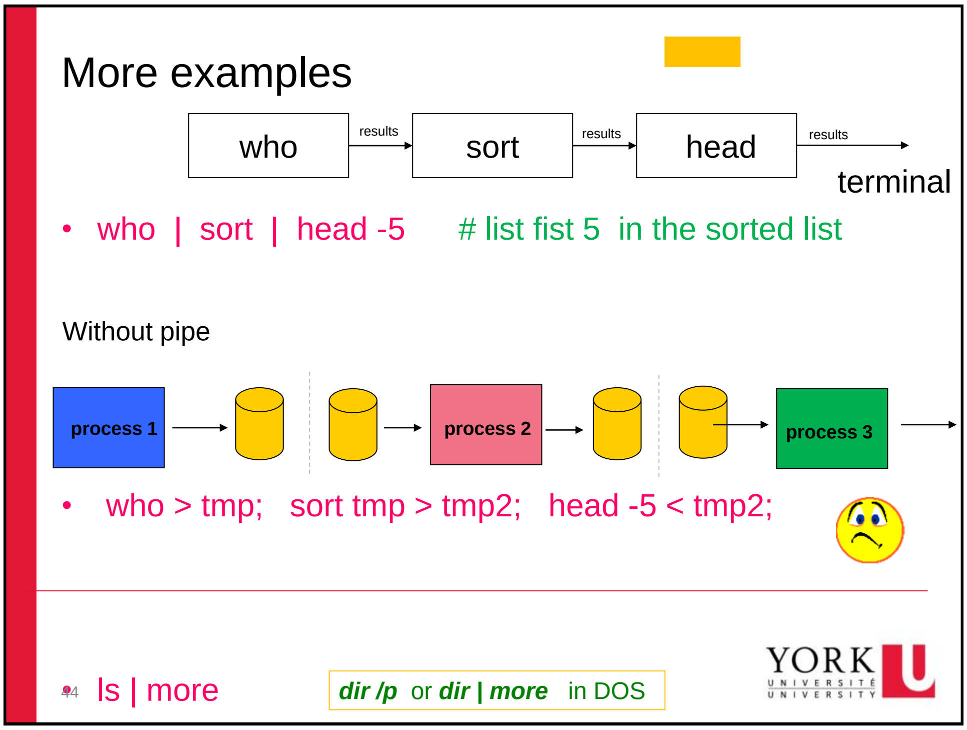
tail -n { fileName }*

- The tail utility displays the last n lines of a file.
  - If n is not specified, it defaults to 10.

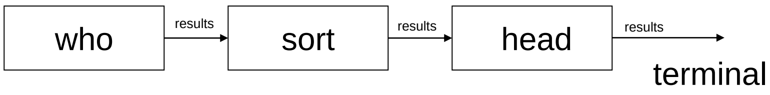
```
$ head -2 heart.txt                         #  list the first two lines.
I hear her breathing,
I'm surrounded by the sound.

$ tail -2 heart.txt                         #  list the last two lines.
Floating in this secret place,
I never shall be found.

$ who | head -3               # use pipe
$ who | sort | tail -3        # see what happens
```

43

43

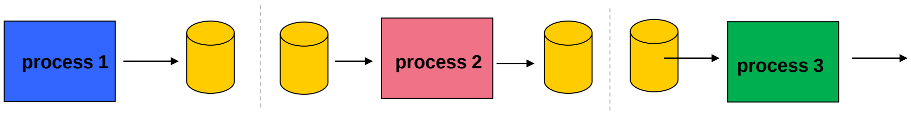


More examples



- who | sort | head -5   # list fist 5 in the sorted list

Without pipe



- who > tmp; sort tmp > tmp2; head -5 < tmp2;

- ls | more   *dir /p* or *dir | more* in DOS

44

44

So far: Displaying a file: cat, more/less, head/tail

Q: How to copy a file? How to copy a directory?

Q: How to rename a file? How to rename a directory?
Copy and remove old?

Q: How to remove a file? How to remove a directory?

File
cat
more less
head tail
cp
mv
rm
wc
file
chmod
chgrp

46

Copy a file/dir: cp

copy in DOS

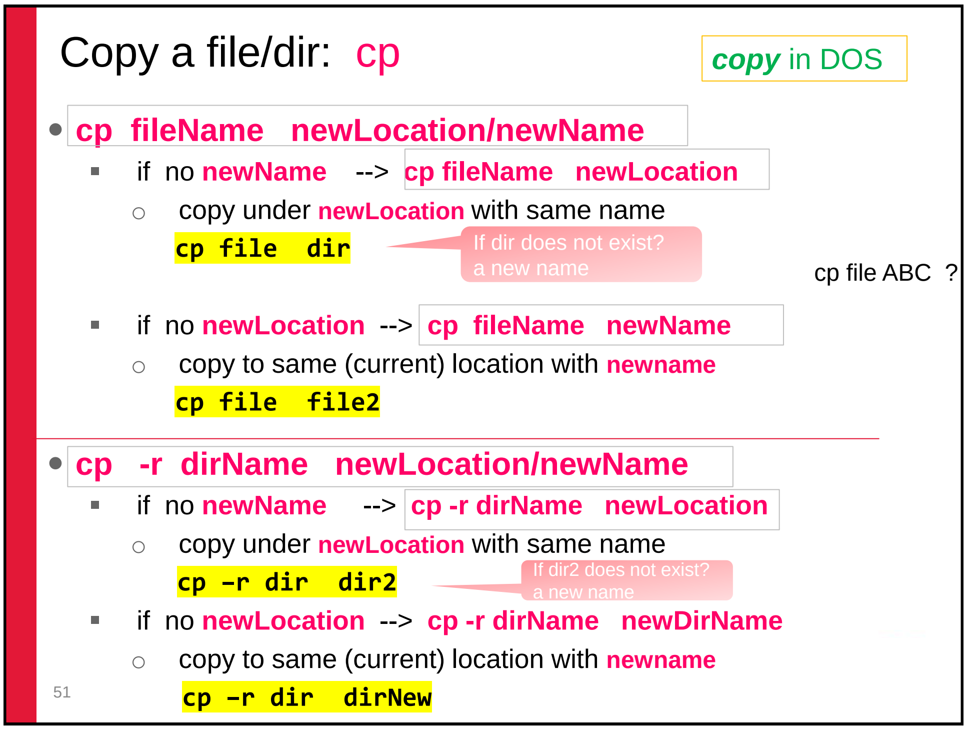
- **cp -i location/fileName   newLocation/newName**
  - `cp dir1/file1   dir2/file2`
- The -i option prompts you for confirmation if newName already exists so that you do not accidentally replace its contents.

- **cp   -r location/dirName   newLocation/newName**
  - copy directory !!!

- Now assume in the directory where the source file/directory exists

```
cp      fileName   newLocation/newName
cp  -r dirName    newLocation/newName
```

49



Copy a file/dir: cp

copy in DOS

- cp fileName newLocation/newName
  - if no newName --> cp fileName newLocation
    - copy under newLocation with same name

```
cp file  dir
```

If dir does not exist?
a new name

cp file ABC ?

  - if no newLocation --> cp fileName newName
    - copy to same (current) location with newname

```
cp file  file2
```

- cp -r dirName newLocation/newName
  - if no newName --> cp -r dirName newLocation
    - copy under newLocation with same name

```
cp –r dir  dir2
```

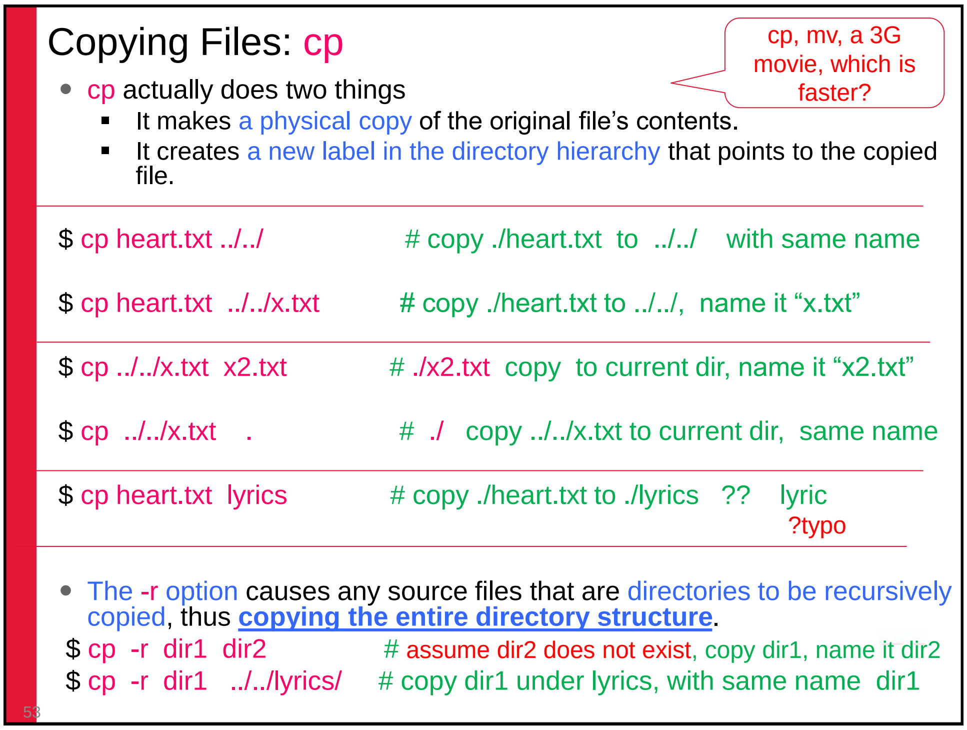
If dir2 does not exist?
a new name

  - if no newLocation --> cp -r dirName newDirName
    - copy to same (current) location with newname

```
cp –r dir  dirNew
```

51

51



Copying Files: cp

cp, mv, a 3G movie, which is faster?

- cp actually does two things
  - It makes a physical copy of the original file’s contents.
  - It creates a new label in the directory hierarchy that points to the copied file.

```
$ cp heart.txt ../../            # copy ./heart.txt  to  ../../    with same name

$ cp heart.txt  ../../x.txt      # copy ./heart.txt to ../../,  name it “x.txt”

$ cp ../../x.txt  x2.txt         # ./x2.txt  copy  to current dir, name it “x2.txt”

$ cp  ../../x.txt    .           #  ./   copy ../../x.txt to current dir,  same name

$ cp heart.txt  lyrics           # copy ./heart.txt to ./lyrics   ??    lyric
                                                                          ?typo
```

- The -r option causes any source files that are directories to be recursively copied, thus **copying the entire directory structure**.

```
$ cp -r dir1 dir2            # assume dir2 does not exist, copy dir1, name it dir2
$ cp -r dir1  ../../lyrics/  # copy dir1 under lyrics, with same name  dir1
```

53

53

Renaming/Moving a file/directory: mv

- **mv -i location/fileOrDirName newLocation/newName**
  - Move to **newLocation** with **newName**

```
mv dir1/file-or-dir  dir2/new-file-or-dir
```

no -r

- The -i option prompts you for confirmation if newName already exists so that you do not accidentally replace its contents.

- Now assume in the directory where source fileorDirName exist
- **mv fileOrDirName newLocation/newName**

56

56

Renaming/Moving a file/directory: mv

- **mv fileOrDirName newLocation/newName**
  - Move to **newLocation** with **newName**
  - if no **newName** --> **mv fileOrDirName newLocation**
    - **move** to **newLocation** with same name

```
mv file-or-dir  dir2
```

***move* in DOS**

  - if no **newLocation** --> **mv fileOrDirName newName**
    - move to same (current) location with **newName**
    - actually **rename**

```
mv file-or-dir  new-file-or-dir
```

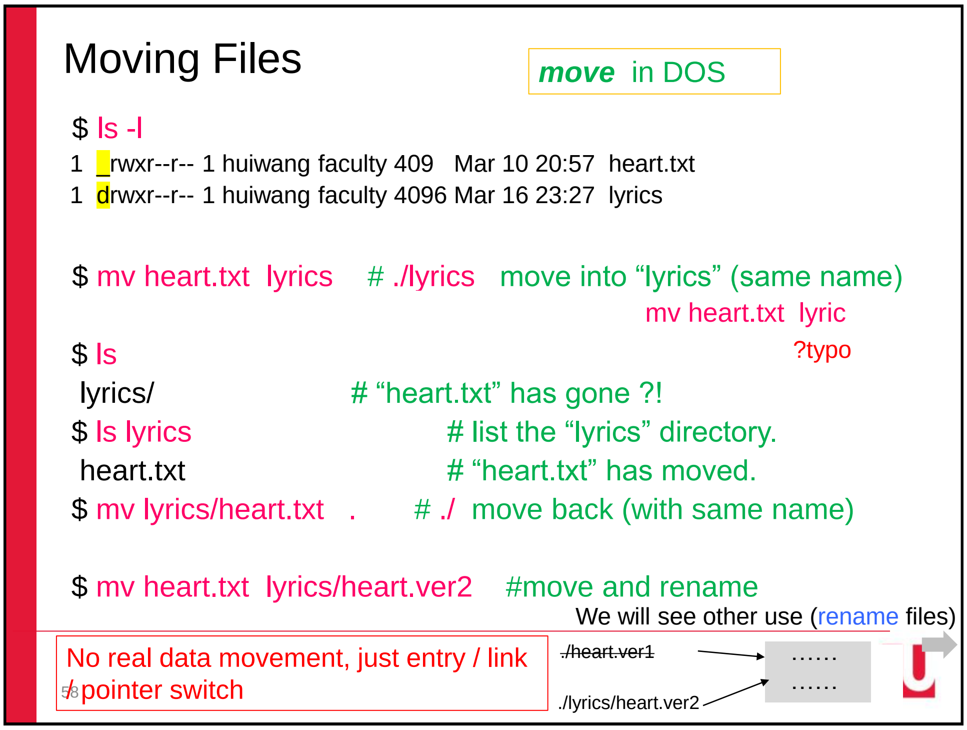
***rename* in DOS**

If dir2 does not exist?
a new name

cp directory need -r
mv directory does not

mv file ABC ?

57



Moving Files

move in DOS

```
$ ls -l
1  _rwxr--r-- 1 huiwang faculty 409   Mar 10 20:57  heart.txt
1  drwxr--r-- 1 huiwang faculty 4096 Mar 16 23:27  lyrics

$ mv heart.txt  lyrics    # ./lyrics   move into “lyrics” (same name)
                                              mv heart.txt  lyric
$ ls                                                      ?typo
 lyrics/                 # “heart.txt” has gone ?!
$ ls lyrics                         # list the “lyrics” directory.
 heart.txt                          # “heart.txt” has moved.
$ mv lyrics/heart.txt  .       # ./  move back (with same name)

$ mv heart.txt  lyrics/heart.ver2    #move and rename
```

We will see other use (rename files)

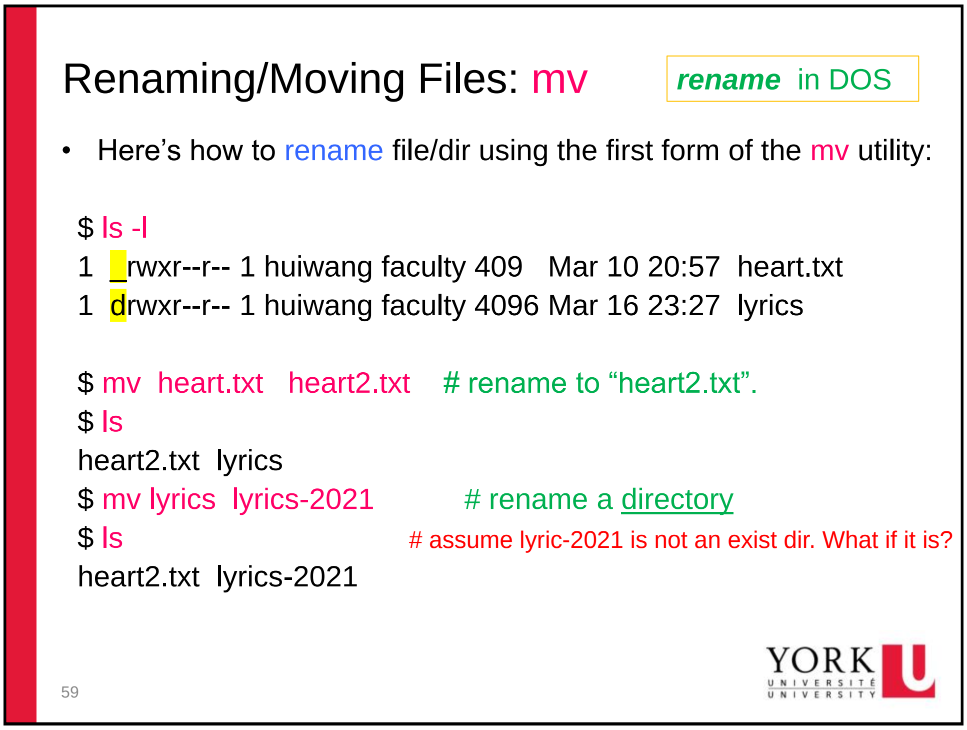
No real data movement, just entry / link / pointer switch

~~./heart.ver1~~ → ……

./lyrics/heart.ver2 → ……

58

58



Renaming/Moving Files: mv

rename in DOS

- Here’s how to rename file/dir using the first form of the mv utility:

```
$ ls -l
1  _rwxr--r-- 1 huiwang faculty 409   Mar 10 20:57  heart.txt
1  drwxr--r-- 1 huiwang faculty 4096 Mar 16 23:27  lyrics

$ mv  heart.txt   heart2.txt   # rename to “heart2.txt”.
$ ls
heart2.txt  lyrics
$ mv lyrics  lyrics-2021         # rename a directory
$ ls                        # assume lyric-2021 is not an exist dir. What if it is?
heart2.txt  lyrics-2021
```

59

59

Deleting files: rm

del in DOS

- The rm utility removes a file's label from the hierarchy.

```
rm -fir {fileName}*
```

- If the filename doesn't exist, an error message is displayed.
- The -i option prompts the user for confirmation before deleting a filename. It is a very good idea to use this option
  - Open in *tcsh*, not in *sh bash*
- The -f option inhibits all error messages and prompts. It overrides the -i option

This is dangerous!

- If fileName is a directory, the -r option causes all of its contents, including subdirectories, to be recursively deleted.
  - Really used for deleting directories.

60

60

Removing Directories with Files

- The -r option of rm can be used to delete the "lab1" directory and all of its contents with just one command:

```
$ rm  a.out            #  remove a file.

$ rm lab1              # cannot remove lab1: Is a directory
$ rm  -r  lab1         # recursively delete directory.
```

cp directory needs -r
mv directory does not
rm directory needs -r

```
$ rm  -f -r  *
```

recursively delete everything without any confirmation!!!

61

In sh, no -f needed,. Just rm -r * will remove everything

61

cp vs mv

- cp file1 abc   cp -r dir   abc
  mv file1 abc   mv dir1 abc

- Copy (copy+paste) and move (cut+paste) a 3G movie, which is faster?
- Below will both rename file1 to file2, what is the difference?

```
cp file1 file2
rm file1

mv file1 file2
```

62

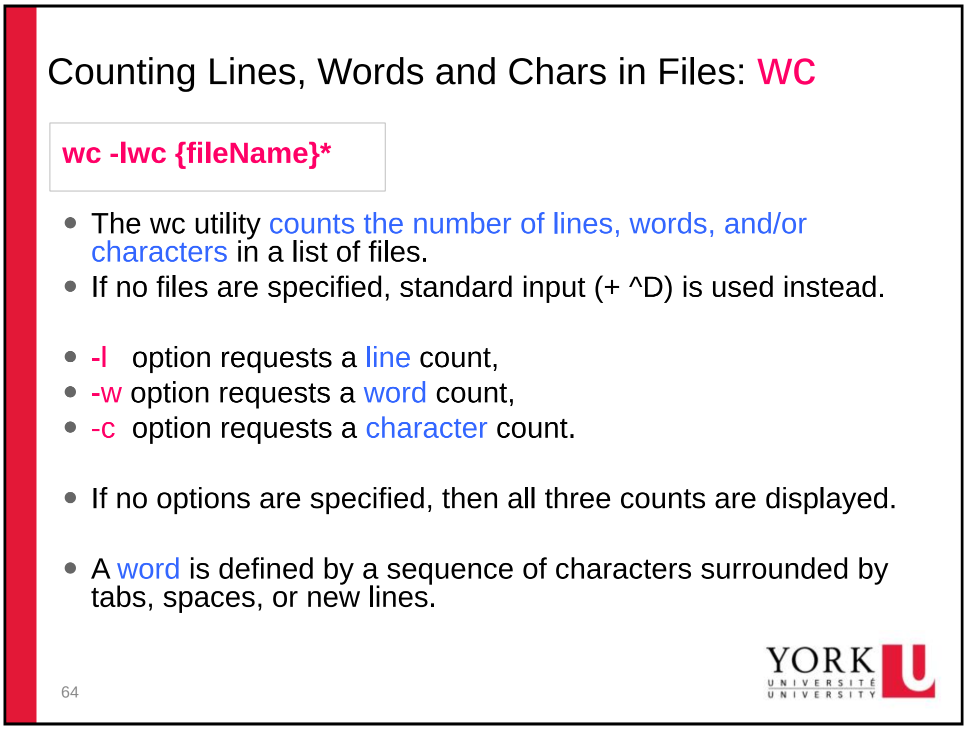
Basic utilities/commands

Introduced the following utilities, listed in in groups:

General	**Directory**	**File**	**File print**
man	**pwd**	**cat**	lp
clear	**mkdir -p**	**more less**	lpr
echo	**ls -l -d -a -R -S -t -r**	**head tail**	lprm
date	**cd**	**cp -r**	lpq
	rmdir must be empty	**mv** move and/or rename	lpstat
		rm -r	
		wc	
		file	
		chmod	
		chgrp	
		chown	
		newgrp	

cp directory needs -r
mv directory does not
rm directory needs -r

63



Counting Lines, Words and Chars in Files: WC

wc -lwc {fileName}*

- The wc utility counts the number of lines, words, and/or characters in a list of files.
- If no files are specified, standard input (+ ^D) is used instead.

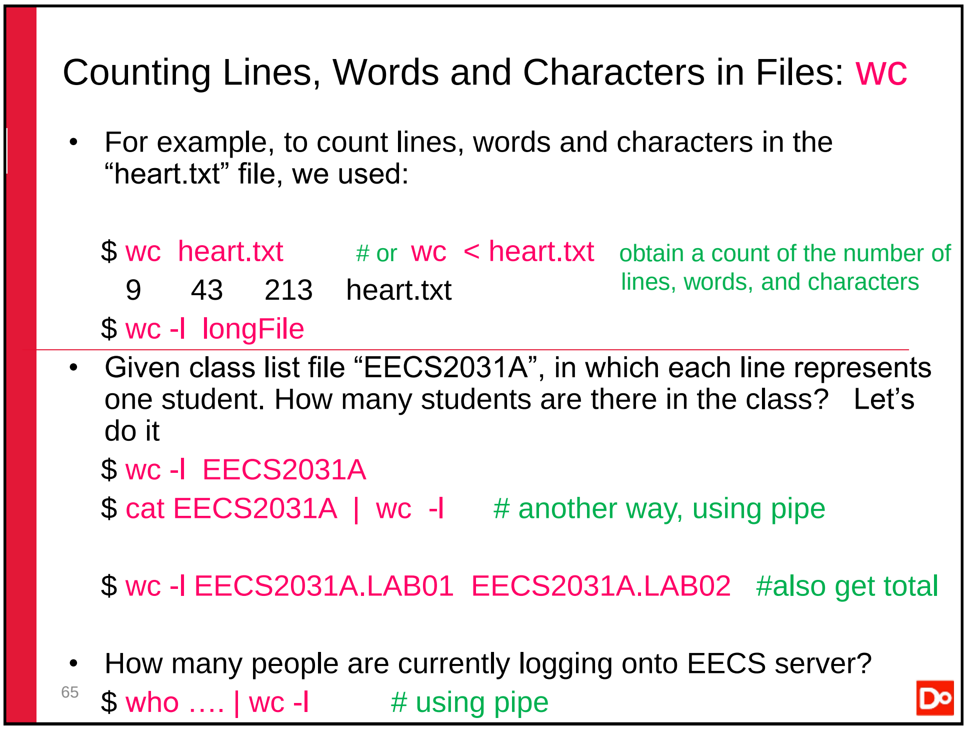
- -l option requests a line count,
- -w option requests a word count,
- -c option requests a character count.

- If no options are specified, then all three counts are displayed.

- A word is defined by a sequence of characters surrounded by tabs, spaces, or new lines.

64

64



Counting Lines, Words and Characters in Files: wc

- For example, to count lines, words and characters in the "heart.txt" file, we used:

```
$ wc  heart.txt        # or  wc  < heart.txt
   9      43     213    heart.txt
$ wc -l  longFile
```

obtain a count of the number of lines, words, and characters

- Given class list file "EECS2031A", in which each line represents one student. How many students are there in the class?   Let's do it

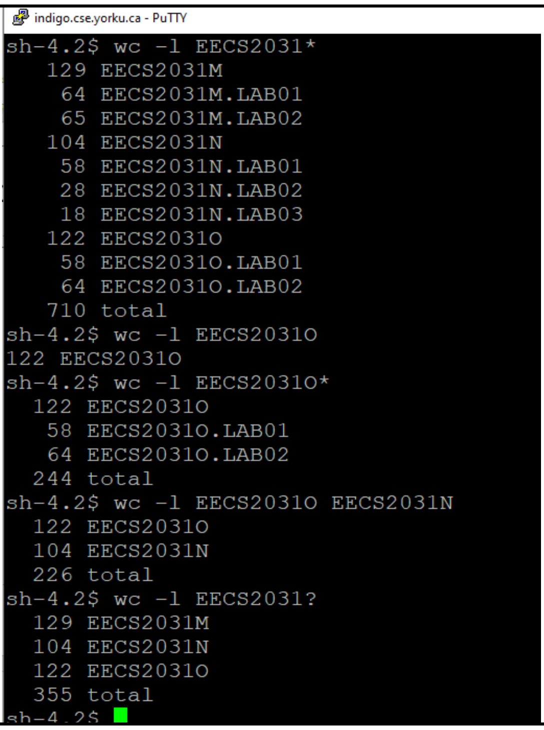
```
$ wc -l  EECS2031A
$ cat EECS2031A  |  wc  -l       # another way, using pipe

$ wc -l EECS2031A.LAB01  EECS2031A.LAB02    #also get total
```

- How many people are currently logging onto EECS server?

```
$ who …. | wc -l          # using pipe
```

65

65



```
sh-4.2$ wc -l EECS2031*
   129 EECS2031M
    64 EECS2031M.LAB01
    65 EECS2031M.LAB02
   104 EECS2031N
    58 EECS2031N.LAB01
    28 EECS2031N.LAB02
    18 EECS2031N.LAB03
   122 EECS2031O
    58 EECS2031O.LAB01
    64 EECS2031O.LAB02
   710 total
sh-4.2$ wc -l EECS2031O
122 EECS2031O
sh-4.2$ wc -l EECS2031O*
  122 EECS2031O
   58 EECS2031O.LAB01
   64 EECS2031O.LAB02
  244 total
sh-4.2$ wc -l EECS2031O EECS2031N
  122 EECS2031O
  104 EECS2031N
  226 total
sh-4.2$ wc -l EECS2031?
  129 EECS2031M
  104 EECS2031N
  122 EECS2031O
  355 total
sh-4.2$
```

cat EECS2031* | wc -l

cat EECS2031O | wc -l

cat EECS2031O* | wc -l

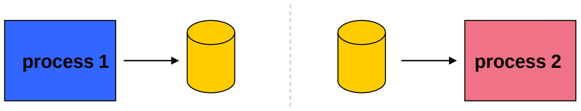
cat EECS2031O EECS2031N | wc -l

cat EECS2031? | wc -l

67

Pipeline Example2

How many users are logged in?



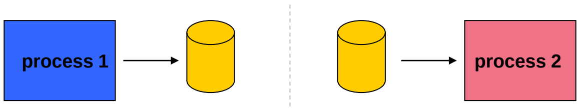
who > tmp.txt   wc -l tmp.txt

Another way

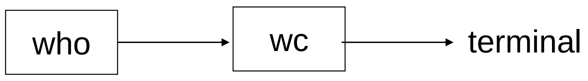
68

Pipeline Example2

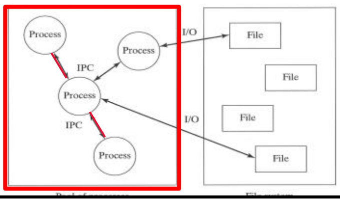
How many users are logged in?

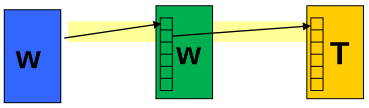


who > tmp.txt   wc -l tmp.txt



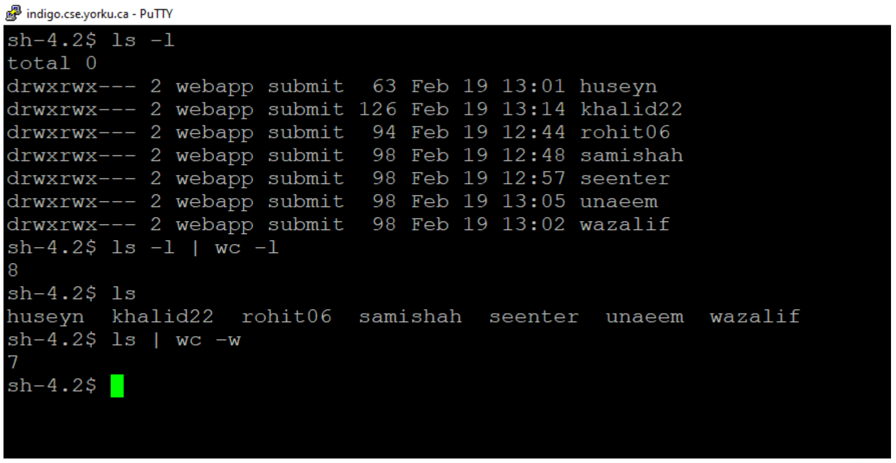
who | wc -l





69

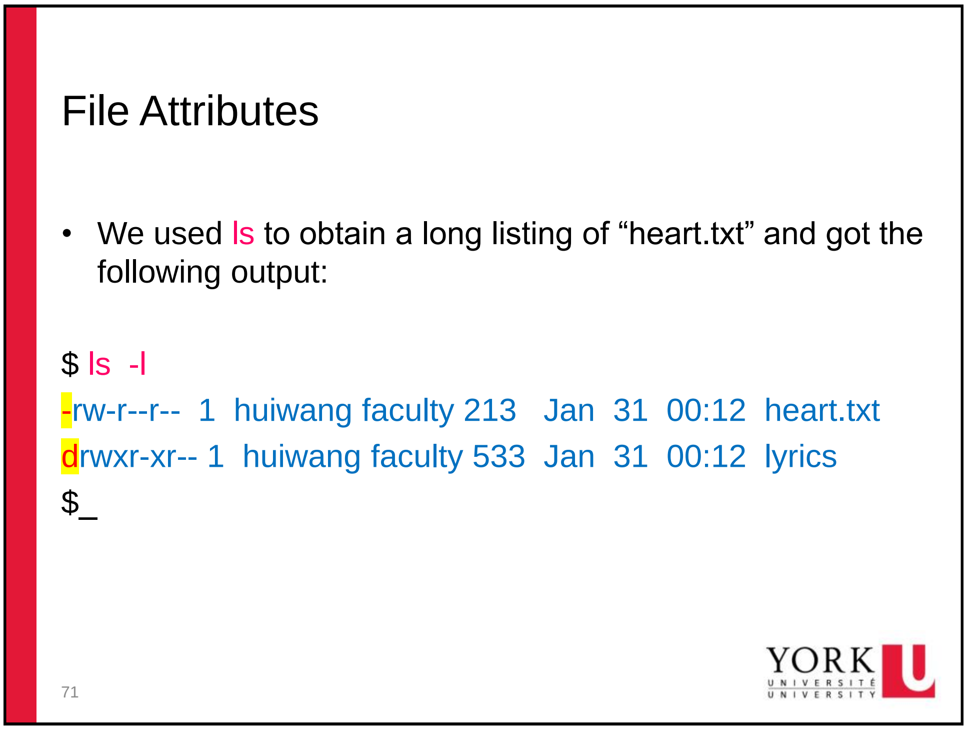
# of entries in a directory?



```
sh-4.2$ ls -l
total 0
drwxrwx--- 2 webapp submit  63 Feb 19 13:01 huseyn
drwxrwx--- 2 webapp submit 126 Feb 19 13:14 khalid22
drwxrwx--- 2 webapp submit  94 Feb 19 12:44 rohit06
drwxrwx--- 2 webapp submit  98 Feb 19 12:48 samishah
drwxrwx--- 2 webapp submit  98 Feb 19 12:57 seenter
drwxrwx--- 2 webapp submit  98 Feb 19 13:05 unaeem
drwxrwx--- 2 webapp submit  98 Feb 19 13:02 wazalif
sh-4.2$ ls -l | wc -l
8
sh-4.2$ ls
huseyn  khalid22  rohit06  samishah  seenter  unaeem  wazalif
sh-4.2$ ls | wc -w
7
sh-4.2$
```

70 Extend lab, check how many submitted

70



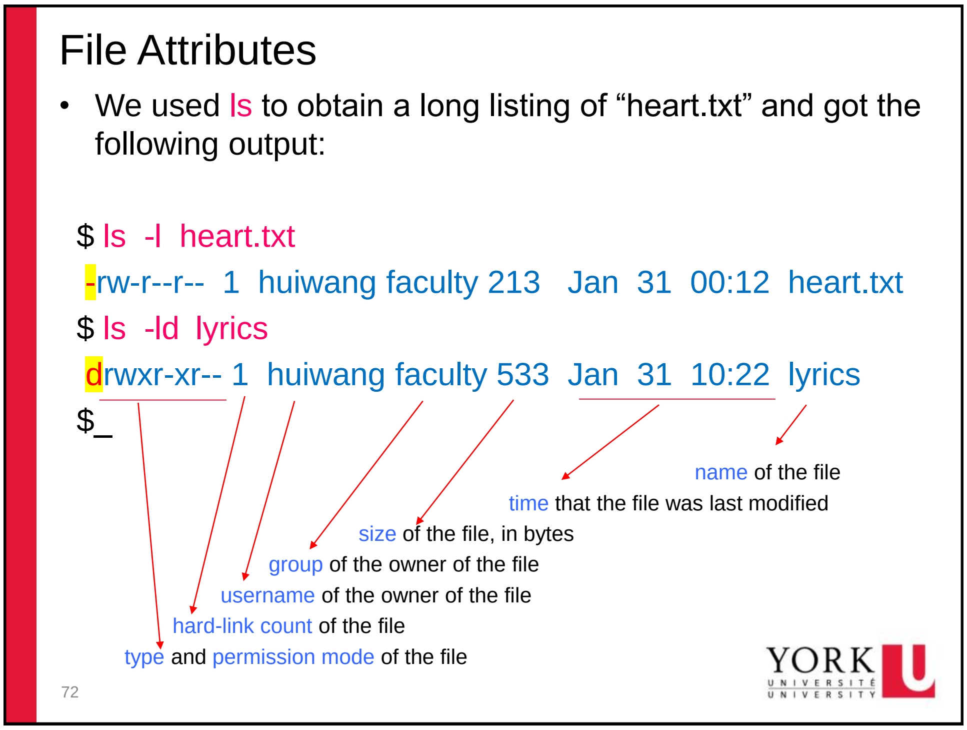
File Attributes

- We used ls to obtain a long listing of “heart.txt” and got the following output:

```
$ ls  -l
-rw-r--r--  1  huiwang faculty 213   Jan  31  00:12  heart.txt
drwxr-xr-- 1  huiwang faculty 533  Jan  31  00:12  lyrics
$_
```

71

71



File Attributes

- We used ls to obtain a long listing of “heart.txt” and got the following output:

```
 $ ls  -l  heart.txt
  -rw-r--r--  1  huiwang faculty 213   Jan  31  00:12  heart.txt
 $ ls  -ld  lyrics
  drwxr-xr-- 1  huiwang faculty 533  Jan  31  10:22  lyrics
 $_
```

name of the file
time that the file was last modified
size of the file, in bytes
group of the owner of the file
username of the owner of the file
hard-link count of the file
type and permission mode of the file

72

72

File Attributes

- **File Types**
  - first field describes the file's type and permission settings.

```
drwxr-xr-- 1  huiwang   faculty  533  Jan  31  10:22  lyrics

- rw-r--r--  1  huiwang   faculty  213  Jan  31  00:12  heart.txt
```

- The first character indicates the type of file, which is encoded as follows :

character	File Type
-	regular file
d	directory file
b	buffered special file(such as a disk drive)
c	unbuffered special file(such as a terminal)
l	symbolic link
p	pipe
s	socket

73

Determining Type of a File: file

file fileName(s)

- The file utility attempts to describe the contents of the fileName argument(s), including the language in which any of the text is written.
- not reliable; it may get confused.

```
$ file heart.txt            # determine the file type.
heart.txt: ASCII text
$ file lab5B.c
lab5B.c: C source, ASCII text
$ file a.out
a.out: ELF 64-bit LSB executable, x86-64, version 1 (SYSV) …..
```

File Permissions (Security) -- revisit

- File permissions are the basis for file security. They are given in three clusters.

$ ls -l heart.txt

```
- rw- r-x r--  1  huiwang   faculty   213   Jan  31  00:12   heart.txt
```

User (owner)	Group	Others
r w -	r - x	r - -

clusters

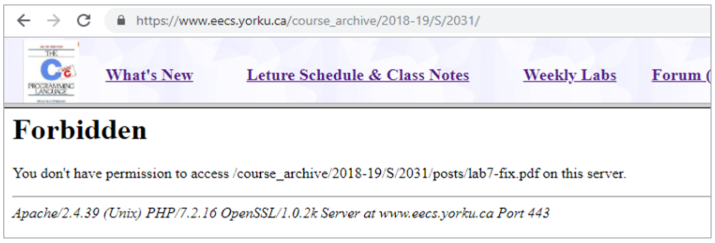
Each cluster of three letters has the same format:

Read permission	Write permission	Execute permission
r	w	x

e.g., webfile: others need to have r permission
submit dir: group need to have w permission

Permission examples

Webfile accessible: others must has r permission



https://www.eecs.yorku.ca/course_archive/2018-19/S/2031/

What's New Leture Schedule & Class Notes Weekly Labs Forum (

Forbidden

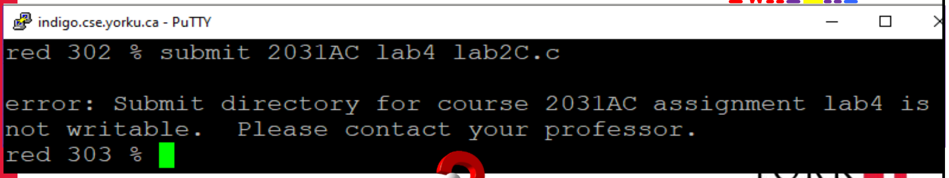
You don't have permission to access /course_archive/2018-19/S/2031/posts/lab7-fix.pdf on this server.

Apache/2.4.39 (Unix) PHP/7.2.16 OpenSSL/1.0.2k Server at www.eecs.yorku.ca Port 443

-rwxr-x---

submit directory open: group must has w permission

-rwxr-xr--



indigo.cse.yorku.ca - PuTTY

```
red 302 % submit 2031AC lab4 lab2C.c

error: Submit directory for course 2031AC assignment lab4 is
not writable.  Please contact your professor.
red 303 %
```

How to set/change permission? chmod

Change File Permissions: chmod

Only owner and admin can change

```
chmod -R change{, change}* {fileName }+
```

- The **chmod** utility changes the **modes (permissions)** of the specified files according to the change parameters, which may take the following forms:

clusterSelection + newPermissions	(add permissions)
clusterSelection - newPermissions	(subtract permissions)
clusterSelection = newPermissions	(assign permissions absolutely)

where **clusterSelection** is any combination of:

- **u** (user/owner)
- **g** (group)
- **o** (others)
- **a** (all)

$$\underbrace{\underbrace{\text{rw-}}_{u}\ \underbrace{\text{r-x}}_{g}\ \underbrace{\text{r--}}_{o}}_{a}$$

newPermissions is any combination of

r (read) **w (write)** **x (execute)**

78

- The **-R** option recursively changes the modes of the files in directories.

$$\underbrace{\underbrace{\text{rw-}}_{u}\ \underbrace{\text{r-x}}_{g}\ \underbrace{\text{r--}}_{o}}_{a}$$

Changing File Permissions: examples

Requirement	Change parameters
<u>Add</u> group a write permission	**chmod g+w file/dir**
<u>Remove</u> user (owner) a write permission	**u-w**
<u>Remove</u> other's read and write permission	**o-rw o-wr o-r,o-w**
<u>Add</u> execute permission for user, group, and others.	**a+x u+x,g+x,o+x ugo+x**
<u>Give</u> the group read permission only.	**g=r**
<u>Add</u> write permission for user, and <u>remove</u> group read permission.	**u+w, g-r**
<u>Give</u> the other read and execute permission	**o=wx o=xw**

overwrite old

79

Changing File Permission: examples

$\underbrace{\underbrace{\text{rw-}}_{u}\ \underbrace{\text{r-x}}_{g}\ \underbrace{\text{r--}}_{o}}_{a}$

- Here's an example of how to set these permissions:

```
$ ls -l lab4.pdf              # not accessible on web
 -rw-r-----   1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf
$ chmod o+r lab4.pdf       # accessible now
$ ls -l lab4.pdf
 -rw-r--r--   1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf
$ chmod a+x  lab4.pdf
$ ls -l lab4.pdf
 -rwxr-xr-x   1   huiwang    faculty  213  Jan  31  00:12  lab4.pdf
```

fix

```
$ ls  -ld  2031          # list attributes of directory 2031 itslef.
 drwxr-xr-x   45   huiwang  faculty 4096   Apr    29    14:35
$ chmod   o-rx 2031            # other more  rx    o-r, o-x
$ ls  -ld   2031
 drwxr-x---   45    huiwang  faculty  4096  Apr  29  14:35
$
```

Changing Permissions Using Numbers   chmod 761

- The chmod utility allows you to specify the new permission setting of a file as 3 octal numbers (0~7) .
- Each octal digit (0~7) represents a permission triplet.

  binary 1/0 1/0 1/0

  r   w   x

For example, if you wanted a file to have the permission settings of

rwx r-x ---   # owner: rwx, group: r x   → chmod u=rwx, g=rw,o=r

then the octal permission setting would be 750, calculated as follows:

	User	Group	Others
setting	rwx	rw-	r--
binary	111	110	100
octal	7	6	1

Changing Permissions Using Numbers chmod 750

- The chmod utility allows you to specify the new permission setting of a file as 3 octal numbers (0~7) .
- Each octal digit (0~7) represents a permission triplet.
  binary 1/0 1/0 1/0
  r w x

For example, if you wanted a file to have the permission settings of

rwx r-x --- # owner: rwx, group: r x → chmod u=rwx, g=rx,o=

then the octal permission setting would be 750, calculated as follows:

	User	Group	Others
setting	rwx	r-x	---
binary	111	101	000
octal	7	5	0

82

82

Changing File Permissions Using Octal Numbers

- The octal permission setting would be supplied to chmod as follows:

```
$ chmod  750  lab4.pdf          # or  chmod u=rwx, g=rx,o=    lab4.pdf
$ ls  -l  lab4.pdf              # confirm.
  - rwx r-x ---      45   huiwang faculty    4096    Apr  29 14:35 lab4.pdf
$ _
```

7 5 0

⇩

111 101 000

⇩

rwx r-x - - -

83

83

Changing Permissions Using Octal Numbers

- The chmod utility allows you to specify the new permission setting of a file as an octal number.

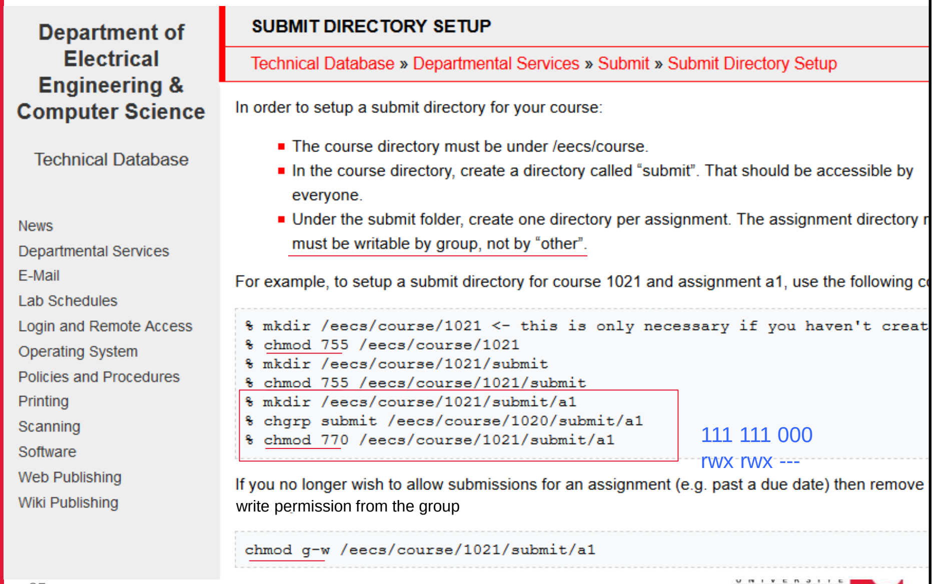
```
+-----+---+--------------------------+
| rwx | 7 | Read, write and execute  | 111
| rw- | 6 | Read, write              | 110
| r-x | 5 | Read, and execute        | 101
| r-- | 4 | Read,                    | 100
| -wx | 3 | Write and execute        | 011
| -w- | 2 | Write                    | 010
| --x | 1 | Execute                  | 001
| --- | 0 | no permissions           | 000
+------------------------------------+
```

```
+------------------------+-----------+
| chmod u=rwx,g=rwx,o=rx | chmod 775 |
| chmod u=rwx,g=rx,o=    | chmod 750 |
| chmod u=rw,g=r,o=r     | chmod 644 |
| chmod u=rw,g=r,o=      | chmod 640 |
| chmod u=rw,go=         | chmod 600 |
| chmod u=rwx,go=        | chmod 700 |
+------------------------+-----------+
```

84

An example: setting up submit directory using chmod

- https://wiki.eecs.yorku.ca/dept/tdb/services:submit:submit-setup



Department of Electrical Engineering & Computer Science

Technical Database

News
Departmental Services
E-Mail
Lab Schedules
Login and Remote Access
Operating System
Policies and Procedures
Printing
Scanning
Software
Web Publishing
Wiki Publishing

SUBMIT DIRECTORY SETUP

Technical Database » Departmental Services » Submit » Submit Directory Setup

In order to setup a submit directory for your course:

- The course directory must be under /eecs/course.
- In the course directory, create a directory called "submit". That should be accessible by everyone.
- Under the submit folder, create one directory per assignment. The assignment directory r must be writable by group, not by "other".

For example, to setup a submit directory for course 1021 and assignment a1, use the following c

```
% mkdir /eecs/course/1021 <- this is only necessary if you haven't creat
% chmod 755 /eecs/course/1021
% mkdir /eecs/course/1021/submit
% chmod 755 /eecs/course/1021/submit
% mkdir /eecs/course/1021/submit/a1
% chgrp submit /eecs/course/1020/submit/a1
% chmod 770 /eecs/course/1021/submit/a1
```

111 111 000
rwx rwx ---

If you no longer wish to allow submissions for an assignment (e.g. past a due date) then remove write permission from the group

```
chmod g-w /eecs/course/1021/submit/a1
```

85

- Also next page

Let's do it

85

Example of chmod, chgrp

- Create a submission directory for weekly lab
  - mkdir lab7
  - chgrp submit lab7   # change group to 'submit'
  - chmod 770 lab7   # or chmod u=rwx, g=rwx,o=   lab7

```
$ mkdir lab7
$ ls -ld lab7
  drwx------  2 huiwang faculty 4096 Jul 11 16:39 lab7

$ chgrp submit lab7
$ ls -ld lab7
  drwx------  2 huiwang submit 4096 Jul 11 16:39 lab7

$ chmod 770 lab7
$ ls -ld lab7
  drwxrwx---  2 huiwang submit 4096 Jul 11 16:39 lab7
```

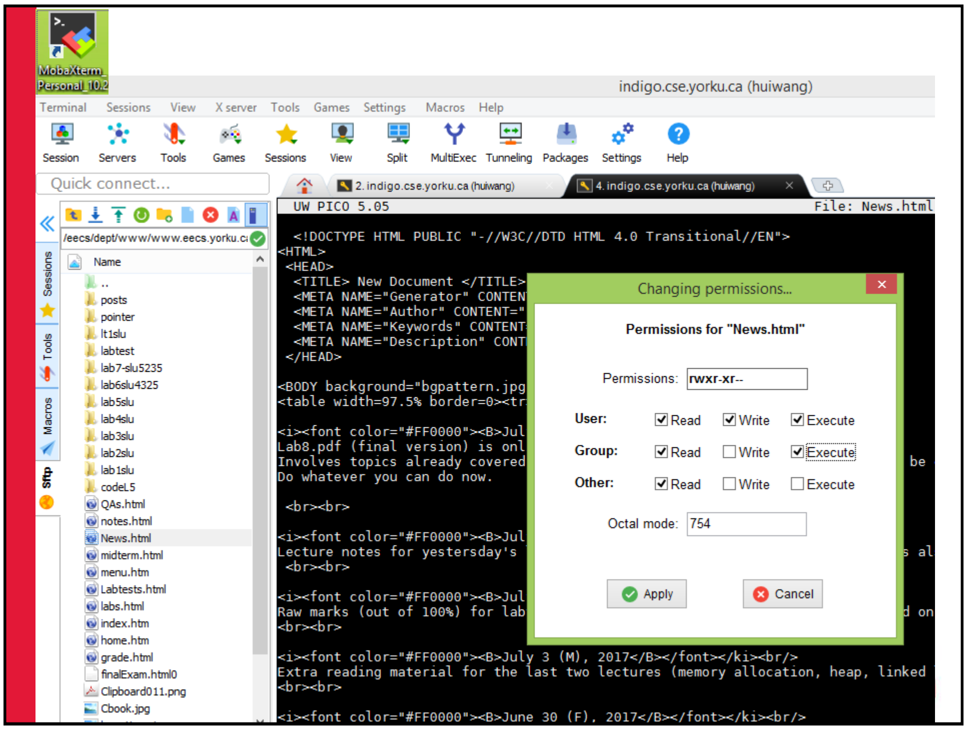
$d\underbrace{rwx}_{7}\underbrace{rwx}_{7}\underbrace{---}_{0}$

- After due time, close the submission

```
  $ chmod g-w lab7     # or  chmod 750  lab7
  drwxr-x---  2 huiwang submit 4096 Jul 11 16:39 lab7
```

86

86


MobaXterm Personal 10.2
indigo.cse.yorku.ca (huiwang)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. indigo.cse.yorku.ca (huiwang)
4. indigo.cse.yorku.ca (huiwang)
Sessions Tools Macros Sftp
/eecs/dept/www/www.eecs.yorku.ca
Name
..
posts
pointer
lt1slu
labtest
lab7-slu5235
lab6slu4325
lab5slu
lab4slu
lab3slu
lab2slu
lab1slu
codeL5
QAs.html
notes.html
News.html
midterm.html
menu.htm
Labtests.html
labs.html
index.htm
home.htm
grade.html
finalExam.html0
Clipboard011.png
Cbook.jpg
Apply
Cancel

```
  UW PICO 5.05                                                    File: News.html

  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> New Document </TITLE>
  <META NAME="Generator" CONTEN
  <META NAME="Author" CONTENT="
  <META NAME="Keywords" CONTENT
  <META NAME="Description" CONT
 </HEAD>

<BODY background="bgpattern.jpg
<table width=97.5% border=0><tr

<i><font color="#FF0000"><B>Jul
Lab8.pdf (final version) is onl
Involves topics already covered                                            be
Do whatever you can do now.

 <br><br>

<i><font color="#FF0000"><B>Jul
Lecture notes for yesterday's                                              s al
 <br><br>

<i><font color="#FF0000"><B>Jul
Raw marks (out of 100%) for lab                                            d on
<br><br>

<i><font color="#FF0000"><B>July 3 (M), 2017</B></font></ki><br/>
Extra reading material for the last two lectures (memory allocation, heap, linked
<br><br>

<i><font color="#FF0000"><B>June 30 (F), 2017</B></font></ki><br/>
```

Changing permissions...

Permissions for "News.html"

Permissions: rwxr-xr--

User:	☑ Read	☑ Write	☑ Execute
Group:	☑ Read	☐ Write	☑ Execute
Other:	☑ Read	☐ Write	☐ Execute

Octal mode: 754

87

Basic utilities/commands

Introduced the following utilities, listed in in groups:

General	**Directory**	**File**	**File print**
man	**pwd**	**cat**	lp
clear	**mkdir -p**	**more less**	lpr
echo	**ls -l -d -a -R -S -t -r**	**head tail**	lprm
date	**cd**	**cp -r**	lpq
	rmdir must be empty	**mv** move and/or rename	lpstat
		rm -r	
		wc -l -c -w	
		file	
		chmod g+w 750	
		chgrp	
		chown	
		newgrp	

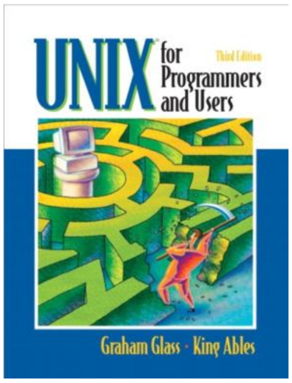
cp directory needs -r
mv directory does not
rm directory needs -r

88

Contents

- Overview of UNIX
  - Structures
  - File systems
    - Pathname: absolute vs relative
    - Security `-rwx-rw--x`
  - Process:
    - Exit code ≥ 0
    - IPC: Pipes

      who | sort who | sort | head -3



today

- **Utilities/commands**
  - Basic: **pwd, ls, rmdir, mkdir, cat, more, mv, cp, rm, wc, chmod**
  - Advanced: grep/egrep, sort, cut, find ….
- Shell and shell scripting language

89

- Unix tutorials on eClass
- SMQ4 tomorrow

90

90