The File system

The File

- A file is a sequence of bytes
- In UNIX system, everything is represented in the form of file
- UNIX file does not contain the eof (end-of-file) mark
- A file attributes like name, size etc are kept in a separate area if the hard disk, not directly accessible to humans, but only to kernel

Categories of files

- Files in UNIX are 3 types
  - Ordinary (Regular) File
  - Directory File
  - Device File

Ordinary File

- It contains only data as a stream of characters
- These files are created, changed or deleted by the user
- An ordinary file can be divided into
  - Text File

    It contains only printable characters

    Every line is terminated with the newline character
  - Binary File

    It contains both printable and unprintable characters. Most unix commands are binary files

Directory File

- UNIX uses directories to organize the files
- Directories are known as folders under the windows environment
- The UNIX file system is organized as directories, where each directory can contain sub-directories and/or files

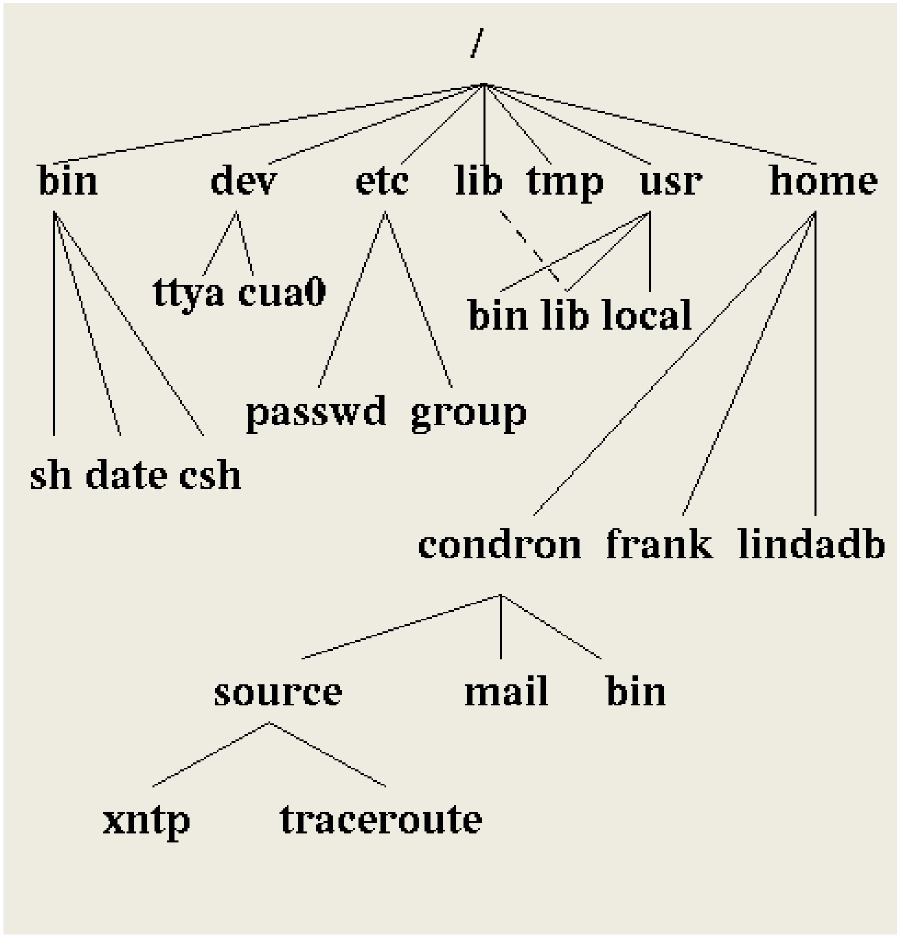
Device File

- All devices and peripherals are represented by files. To read or write a device, you have to perform these operations on its associated file

The Parent-Child Relationship

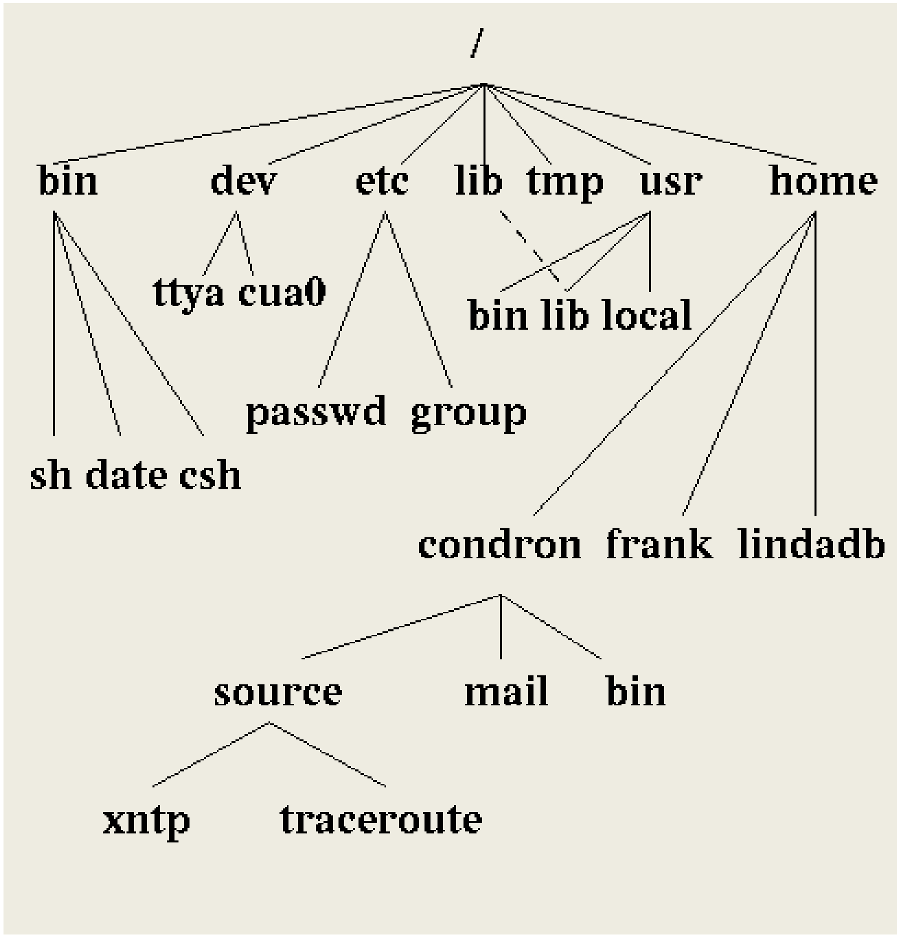
- All files in UNIX are

  Related to one another
- The file system in UNIX is a collection of all of these related files (Ordinary, directory and device file)
- They are organized in a hierarchical structure



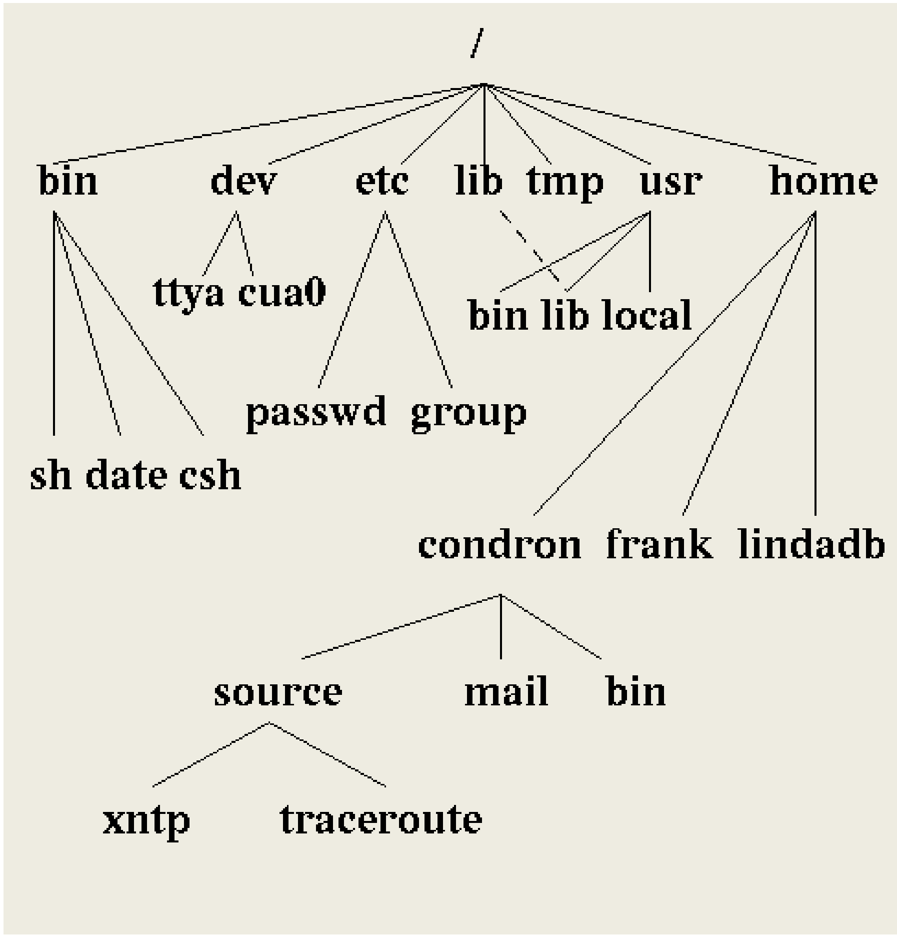
The Parent-Child Relationship

- UNIX file system has reference point for all files called as top
- This top is called root and is represented by a / (front slash)
- root is actually a directory
- The root directory (/) has a number of subdirectories under it



The Parent-Child Relationship

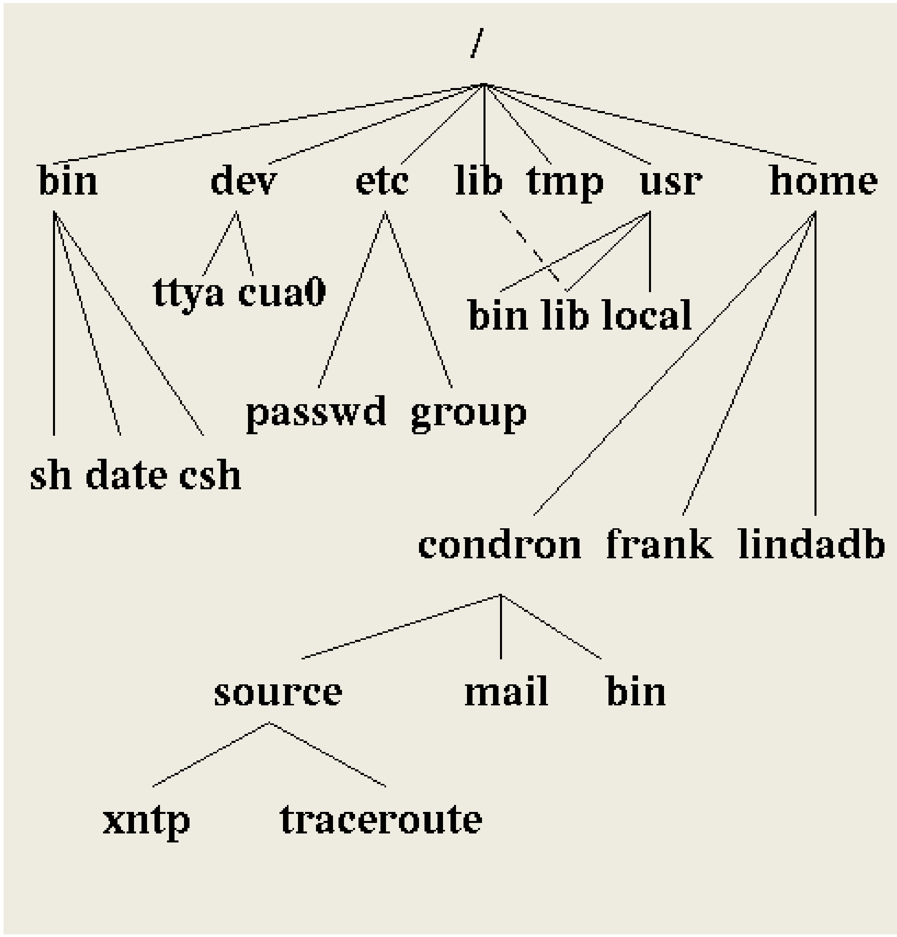
- Every file, apart from root, must have a parent
- The file system follows parent – child relationship, in that the parent is always a directory



The HOME Variable: The Home Directory

- When you log on to the system, UNIX automatically places you in a directory called home directory
- It created by the system when a user account is opened
- The shell variable HOME knows your home directory

  echo $HOME



Where am I?
Checking Your Current Directory

`pwd`

- Print name of the “current working directory”

Example:

```
$pwd
/home/kayal
```

Changing the Current Directory

`cd`

- It is used to move around in the file system
- It changes the current directory to the directory specified as argument

```
$ cd test    (or) cd /test/prog
```

`cd -`	= go back to where you just were
`cd`	= no arguments, go back "home". "home" is where your login starts

Making Directories

`mkdir`

- Create a new directory.
- The command is followed by name of the directory to be created
- **Example:**

```
$mkdir test
$mkdir sem sem/unix
```

Removing Directories

`rmdir`

- This command removes directories
- You can’t delete a directory with rmdir unless it is empty

Absolute Pathnames

- Many UNIX commands use file and directory names as arguments, which are presumed to exist in the current directory
- If the file is not in the current directory then specifying absolute path is necessary
- An absolute path refers to the complete details needed to locate a file or folder, starting from the root element and ending with the other subdirectories

Example

/home/kayar/prog/a.txt

Any command that resides in the directories specified in the PATH variable, you don't need to use the absolute pathname

Relative Pathnames

- A relative path refers to a location that is relative to a current directory or parent directory as reference
- Relative paths make use of two special symbols, a dot (.) and a double-dot (..), which translate into the current directory and the parent directory

Example (Current Directory)

/home/kayar/prog

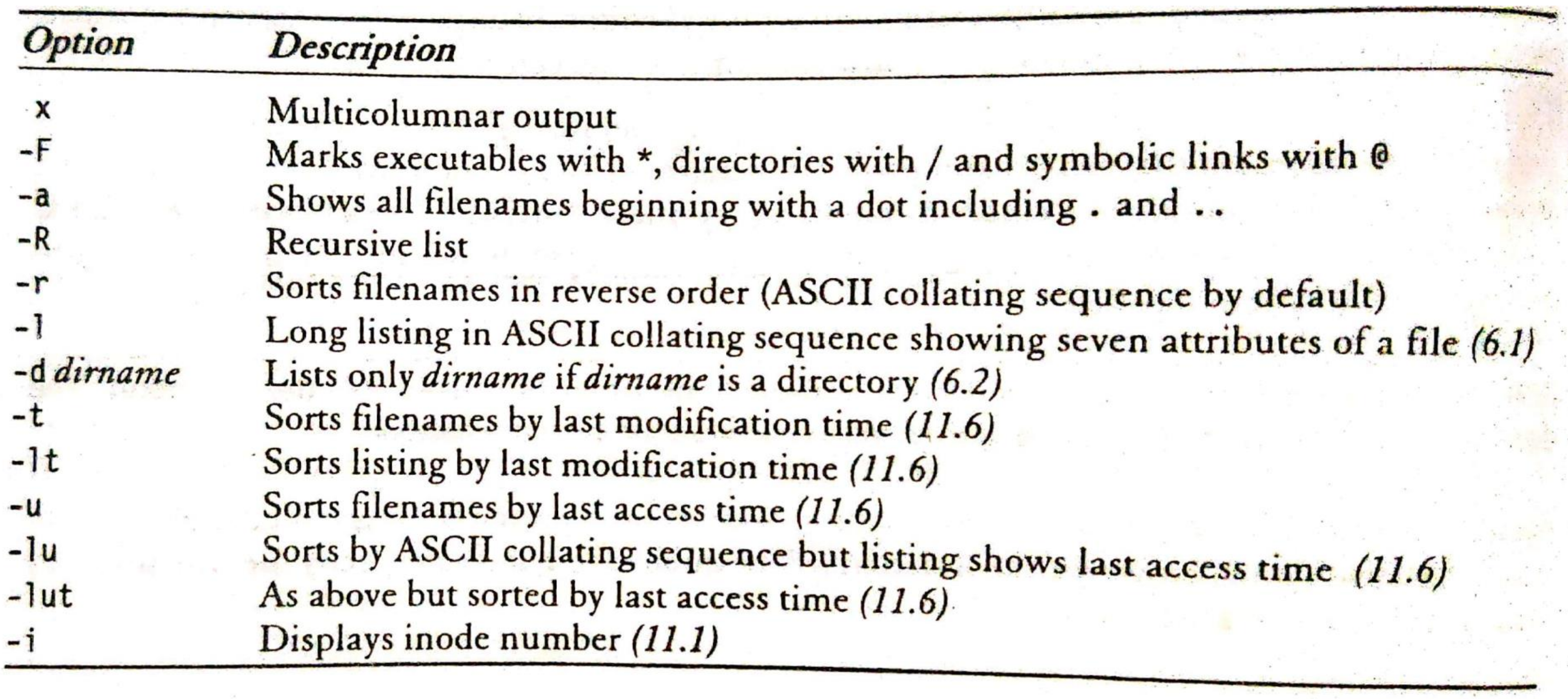
```
cd ../..    =>To move to home directory
cp ../simon/s.txt  .   => copy to current directory
cd ./prog
```

Listing Directory Content

`ls`

- List contents of the current working directory
- **ls –l** - long listing, with dates, owners (File attributes)
- **ls –lrt** - above, but sorted by time

Options to ls command



Option	*Description*
x	Multicolumnar output
-F	Marks executables with *, directories with / and symbolic links with @
-a	Shows all filenames beginning with a dot including . and ..
-R	Recursive list
-r	Sorts filenames in reverse order (ASCII collating sequence by default)
-l	Long listing in ASCII collating sequence showing seven attributes of a file *(6.1)*
-d *dirname*	Lists only *dirname* if *dirname* is a directory *(6.2)*
-t	Sorts filenames by last modification time *(11.6)*
-lt	Sorts listing by last modification time *(11.6)*
-u	Sorts filenames by last access time *(11.6)*
-lu	Sorts by ASCII collating sequence but listing shows last access time *(11.6)*
-lut	As above but sorted by last access time *(11.6)*
-i	Displays inode number *(11.1)*

How do I get help?

`man`

Display the manual for a given program

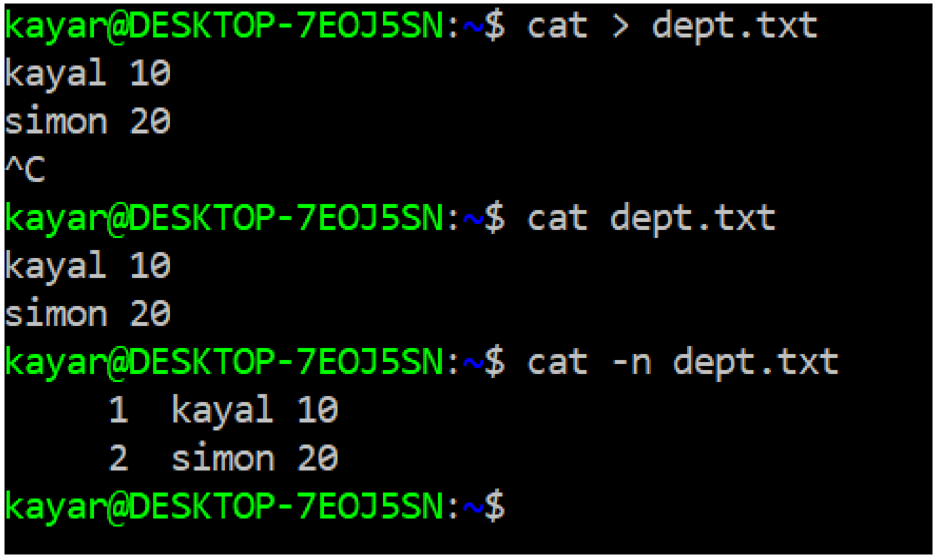
`man ls`	- see manual for the "`ls`" command
`man tcsh`	- learn about the C shell
`man bash`	- learn about that other shell
`man man`	- read the manual for the manual

to return to the command prompt, type "`q`"

Handling Ordinary Files

cat - Displaying and Creating Files

It is used to display the contents of a small file on the terminal



```
kayar@DESKTOP-7EOJ5SN:~$ cat > dept.txt
kayal 10
simon 20
^C
kayar@DESKTOP-7EOJ5SN:~$ cat dept.txt
kayal 10
simon 20
kayar@DESKTOP-7EOJ5SN:~$ cat -n dept.txt
     1  kayal 10
     2  simon 20
kayar@DESKTOP-7EOJ5SN:~$
```

Example : To Create a file

$ cat > file1.txt

This is sample content

[Ctrl+C]

Example : To display the file content

$ cat file1.txt

mv :Renaming files

The **mv** command renames (moves) files. It has two distinct functions

- It renames a file (or directory)
- It moves a group of files to a different directory

Example (Change Name)

```
mv file1.txt file2.txt
```

- It does not create a copy of file. It merely renames it
- If the destination file does not exist, it will be created
- If the destination file exists then it will be overwritten

mv :Renaming files

- A group of files can be moved from one directory to another

Example (same name, different directory)

```
mv file1.txt file2.txt ../simon
```

- mv can be used to rename a directory

Example

```
mv kayal kayar
```

cp : Copying a File

Copy a file. This is just like “mv” except it does not delete the original.

```
cp stupidname.txt betterame.txt
```

- change name, keep original

```
cp chap01 prog/unit1
cp chap01 chap02 chap03 prog
```

- now this is the same as “mv”

cp options

- **Interactive Copying (-i)** – It warns the user before overwriting the destination file

Example

```
cp –i one.txt  two.txt
```

- **Copying Directory Structure (-R)**

Example

```
cp –R progs newprogs
```

rm :Deleting Files

- Remove a file forever. There is no “trash” or “undelete” in unix.

```
rm unwanted_file.txt
```

- delete file with that name

```
rm –f /tmp/yourname/*
```

- forcefully remove everything in your temporary directory.

Will not prompt for confirmation!

```
rm *
```

rm options

- **Interactive Copying (-i)** – It ask the user before for confirmation before removing each file

Example

```
rm –i  chap01 chap02 chap03
```

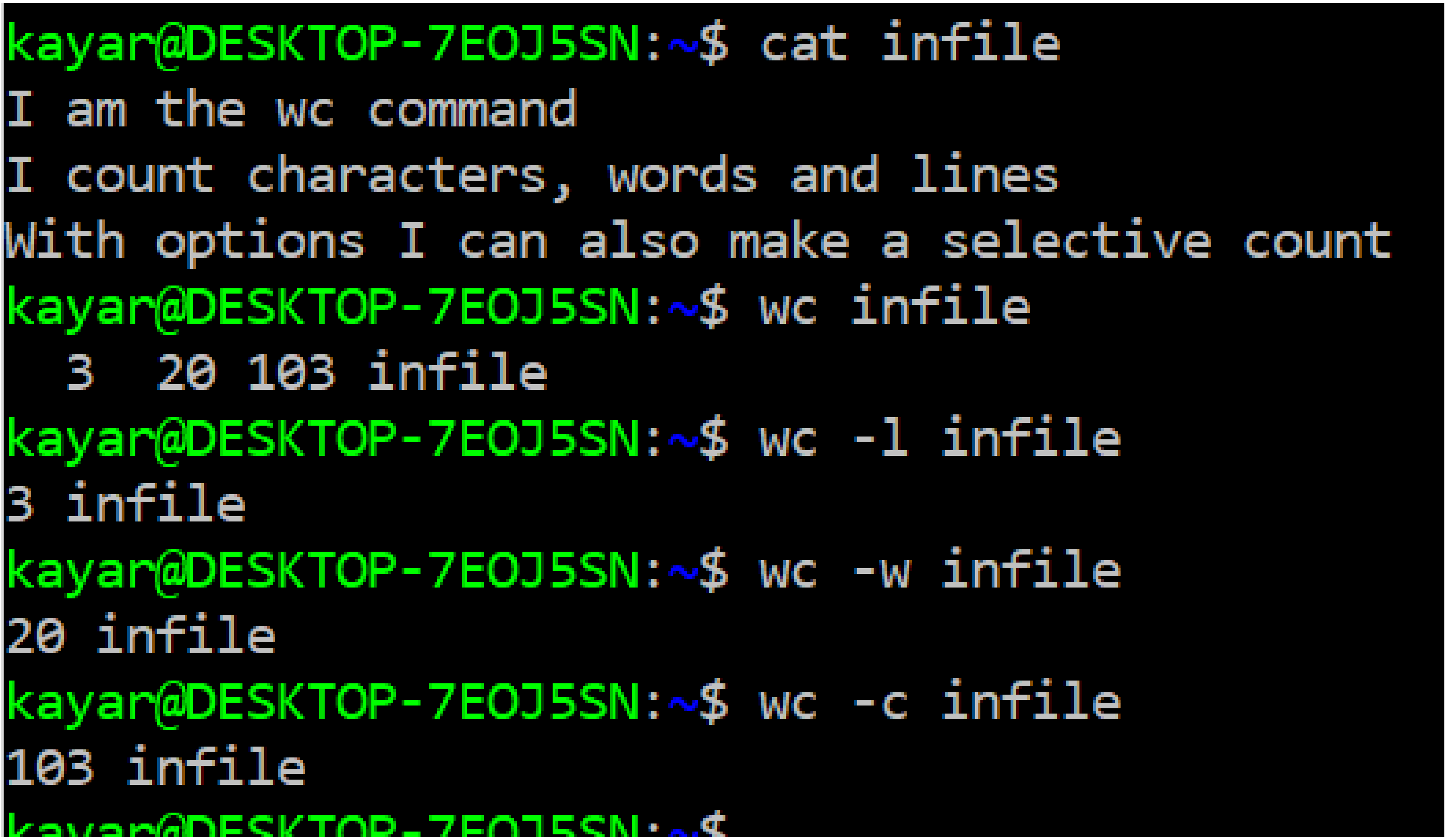
- **Recursive Deletion (-r or –R) –** Recursively deletes for all subdirectories and files

```
rm –r *
```

- **Forcing Removal (-f) –** rm prompts for removal if a file is write protected

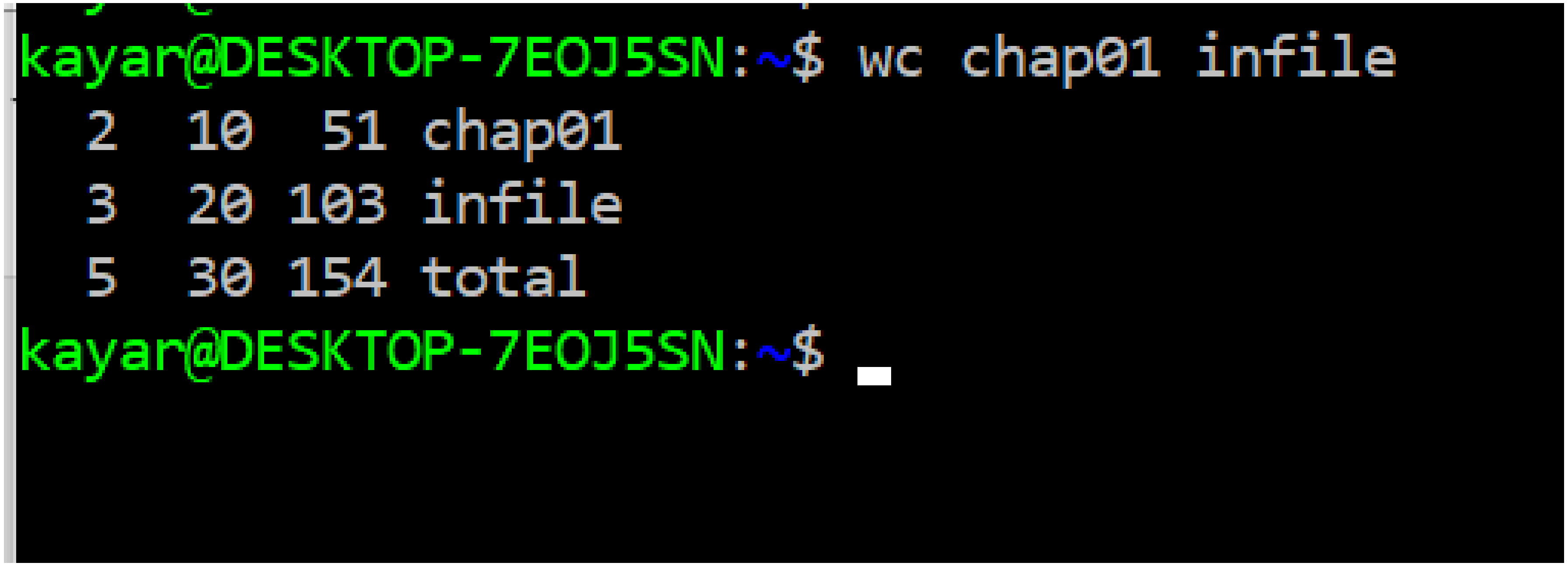
```
rm –rf *
```

wc : Counting Lines, words and Characters



```
kayar@DESKTOP-7EOJ5SN:~$ cat infile
I am the wc command
I count characters, words and lines
With options I can also make a selective count
kayar@DESKTOP-7EOJ5SN:~$ wc infile
  3  20 103 infile
kayar@DESKTOP-7EOJ5SN:~$ wc -l infile
3 infile
kayar@DESKTOP-7EOJ5SN:~$ wc -w infile
20 infile
kayar@DESKTOP-7EOJ5SN:~$ wc -c infile
103 infile
kayar@DESKTOP-7EOJ5SN:~$
```

wc : Counting Lines, words and Characters



```
kayar@DESKTOP-7EOJ5SN:~$ wc chap01 infile
  2  10  51 chap01
  3  20 103 infile
  5  30 154 total
kayar@DESKTOP-7EOJ5SN:~$ 
```

cmp: Comparing two files

- To identify identical files

Example

```
cmp chap01 infile
```

- Displays location of the first mismatch

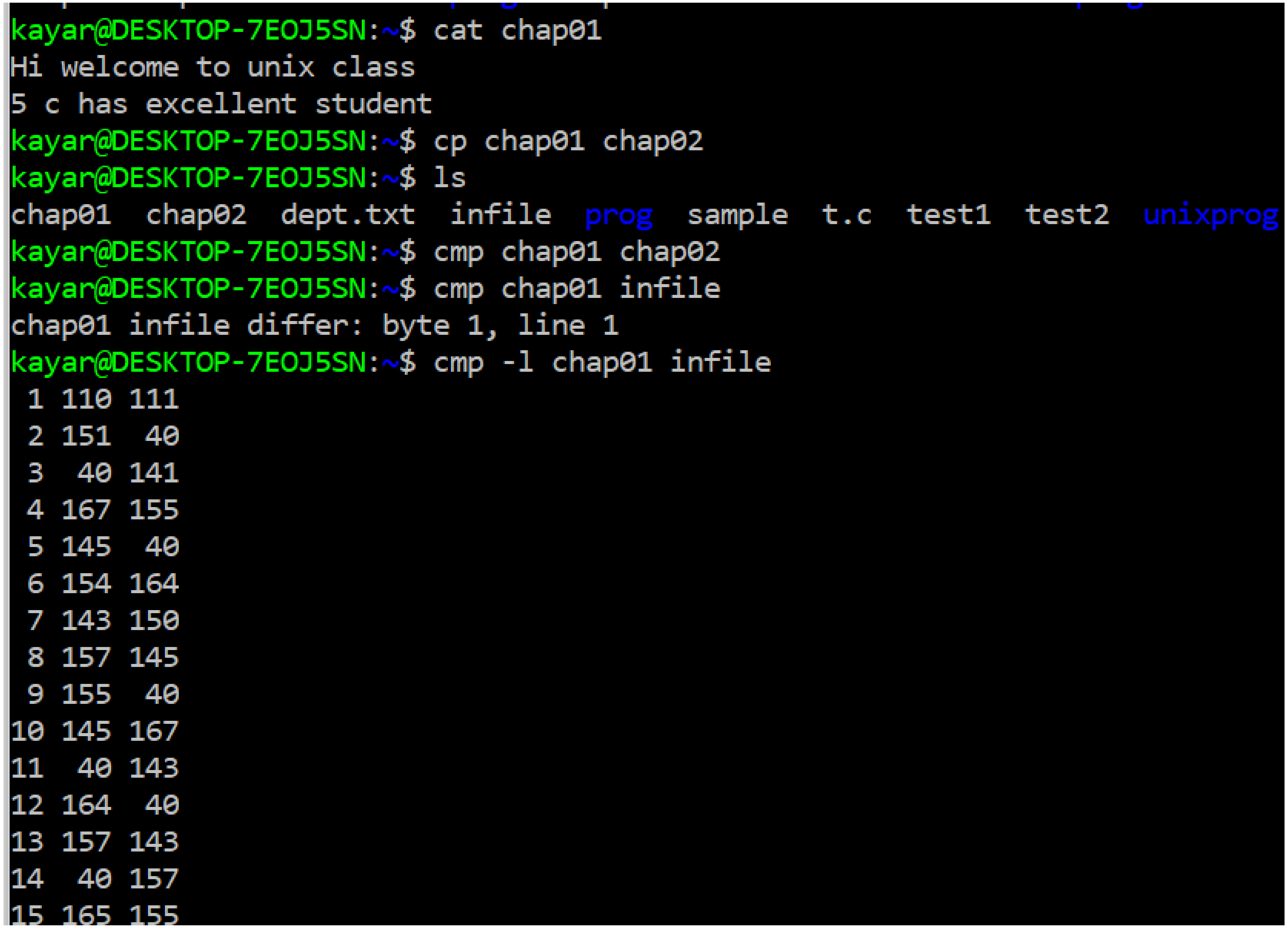
```
cmp chap01 chap02
```

- Displays nothing if it is a identical files

```
cmp -l chap01 chap02
```

- Complete mismatch information will be displayed

cmp: Comparing two files



```
kayar@DESKTOP-7EOJ5SN:~$ cat chap01
Hi welcome to unix class
5 c has excellent student
kayar@DESKTOP-7EOJ5SN:~$ cp chap01 chap02
kayar@DESKTOP-7EOJ5SN:~$ ls
chap01  chap02  dept.txt  infile  prog  sample  t.c  test1  test2  unixprog
kayar@DESKTOP-7EOJ5SN:~$ cmp chap01 chap02
kayar@DESKTOP-7EOJ5SN:~$ cmp chap01 infile
chap01 infile differ: byte 1, line 1
kayar@DESKTOP-7EOJ5SN:~$ cmp -l chap01 infile
 1 110 111
 2 151  40
 3  40 141
 4 167 155
 5 145  40
 6 154 164
 7 143 150
 8 157 145
 9 155  40
10 145 167
11  40 143
12 164  40
13 157 143
14  40 157
15 165 155
```

comm : What is common?

- It is used to identify common content between two files

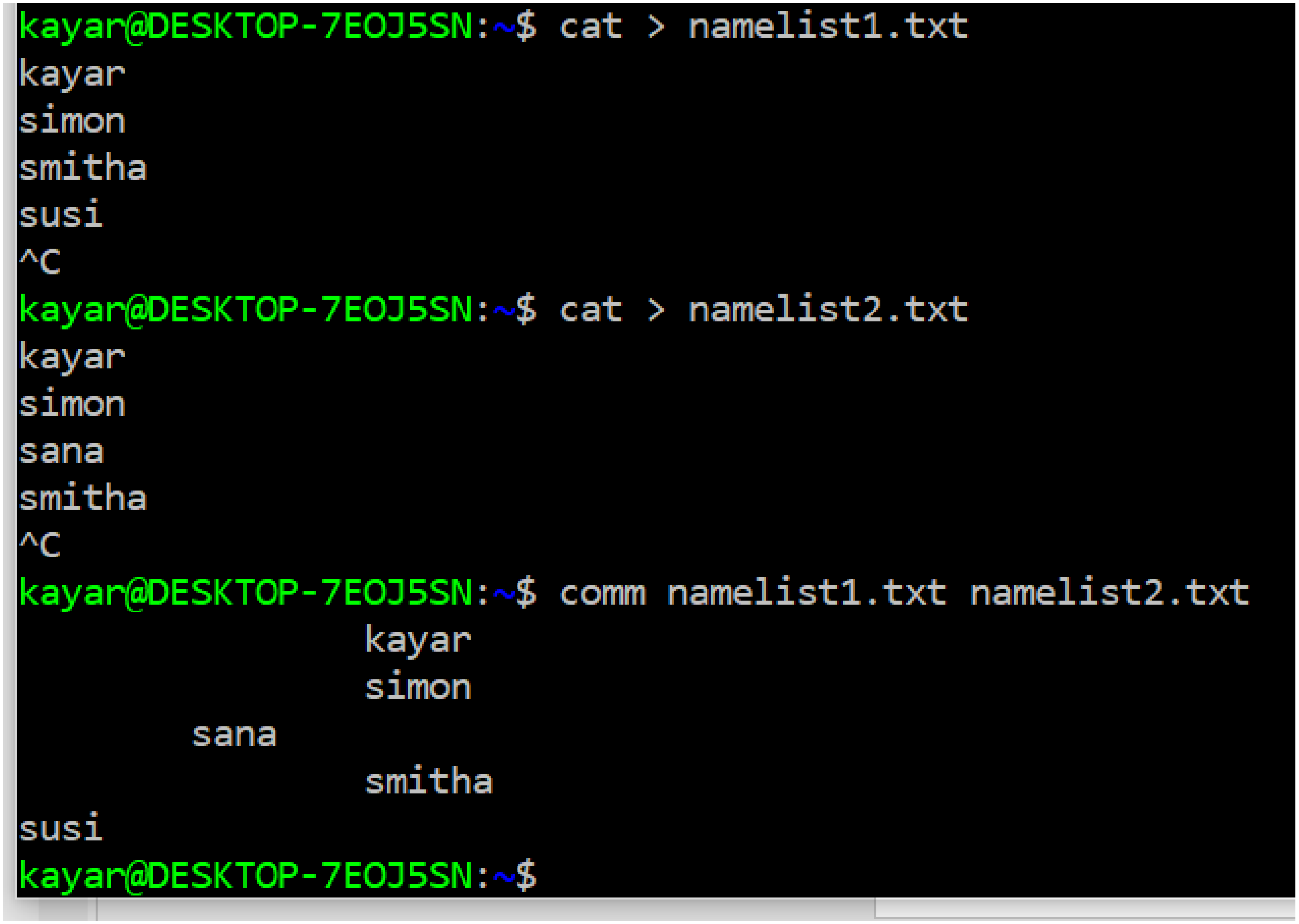
Example

```
comm namelist1.txt namelist2.txt
```

It displays three column output

- The first column contains lines unique to the first file, and the second column shows unique to second file. The third column displays common lines to both files

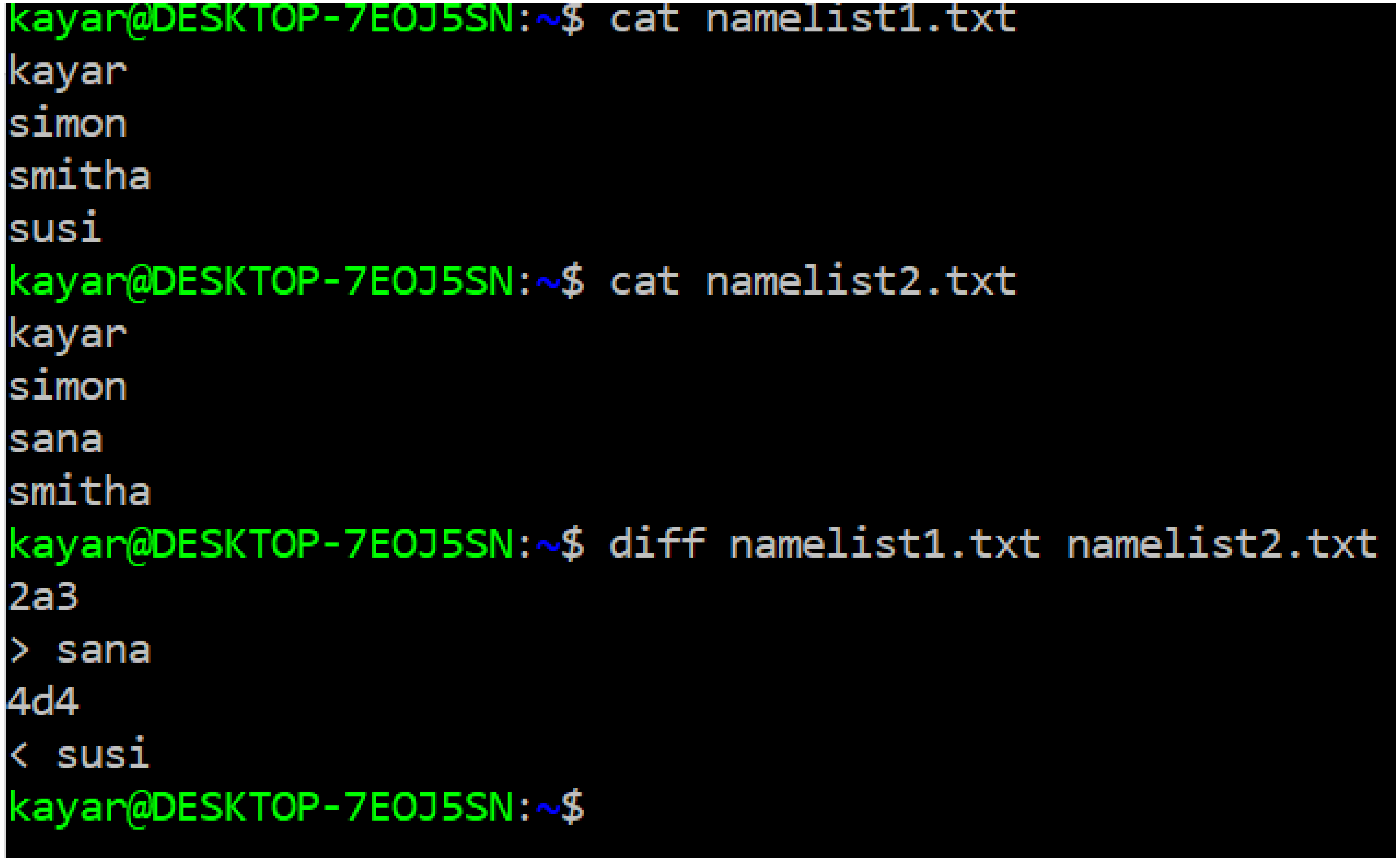
comm : What is common?



```
kayar@DESKTOP-7EOJ5SN:~$ cat > namelist1.txt
kayar
simon
smitha
susi
^C
kayar@DESKTOP-7EOJ5SN:~$ cat > namelist2.txt
kayar
simon
sana
smitha
^C
kayar@DESKTOP-7EOJ5SN:~$ comm namelist1.txt namelist2.txt
                kayar
                simon
        sana
                smitha
susi
kayar@DESKTOP-7EOJ5SN:~$
```

diff : file differences

It tells you which lines in one file have to be changed to make the two files identical



```
kayar@DESKTOP-7EOJ5SN:~$ cat namelist1.txt
kayar
simon
smitha
susi
kayar@DESKTOP-7EOJ5SN:~$ cat namelist2.txt
kayar
simon
sana
smitha
kayar@DESKTOP-7EOJ5SN:~$ diff namelist1.txt namelist2.txt
2a3
> sana
4d4
< susi
kayar@DESKTOP-7EOJ5SN:~$
```

More :Paging out

- The man command displays its output a page at a time
- This is possible because it sends its output to a pager program
- Unix offers standard pager (more and less)

Example

more chap01

The content of chap01 will appear one page at a time. At the bottom of the screen, you will also see the filename and percentage of the file

```
--more -- (17%)
```

Navigation

- more uses the **spacebar (or) f** to scroll forward a page at a time
- **b** to move back one page

The Repeat Features

- **Repeating the Last Command (.)** – the dot command is used to repeat the last command you used

Searching for a pattern

/ command is used to search for a pattern

Example

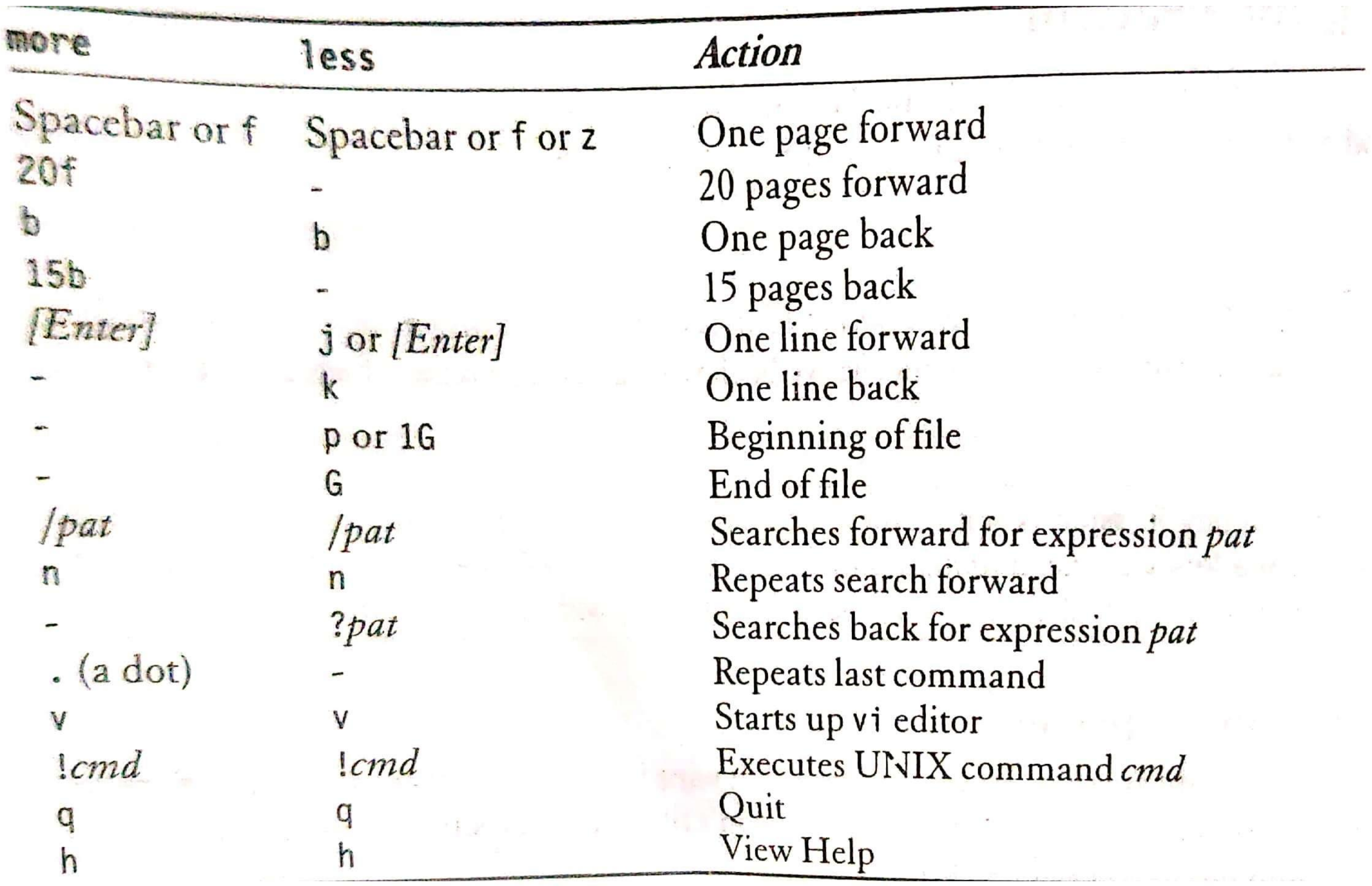
/while

Using more in a Pipeline

man ls | more

man ls command output wont fit on the screen so by pipelining more command helps to navigate a page at a time

Internal Commands of more or less



more	less	*Action*
Spacebar or f	Spacebar or f or z	One page forward
20f	-	20 pages forward
b	b	One page back
15b	-	15 pages back
[Enter]	j or *[Enter]*	One line forward
-	k	One line back
-	p or 1G	Beginning of file
-	G	End of file
/pat	*/pat*	Searches forward for expression *pat*
n	n	Repeats search forward
-	*?pat*	Searches back for expression *pat*
. (a dot)	-	Repeats last command
v	v	Starts up vi editor
!*cmd*	!*cmd*	Executes UNIX command *cmd*
q	q	Quit
h	h	View Help

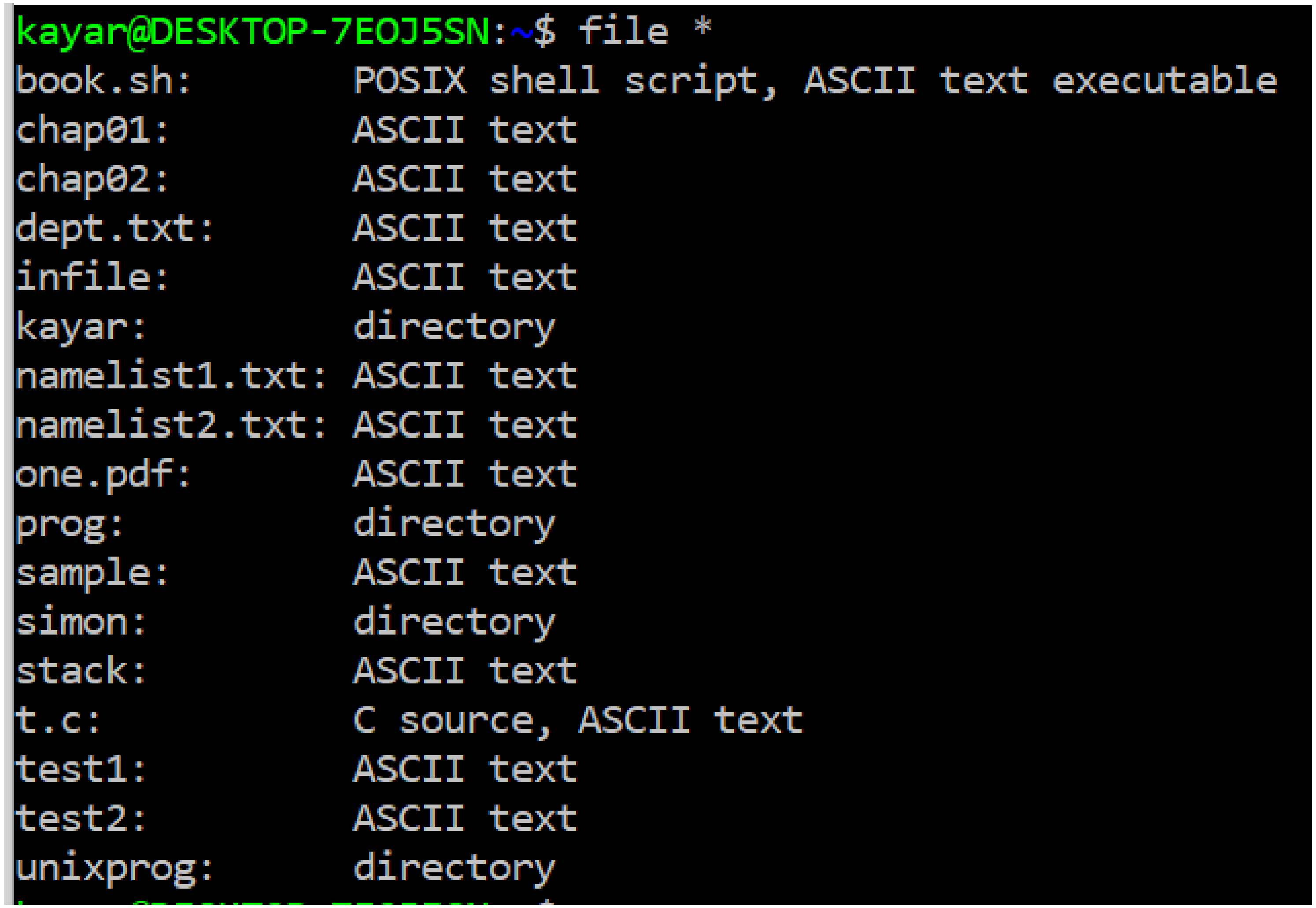
file: Knowing the file type

- **file command to determine the type of file**, especially of an ordinary file
- file correctly identifies the basic file type (regular, directory or device)

Example

```
file *
```

file: Knowing the file type



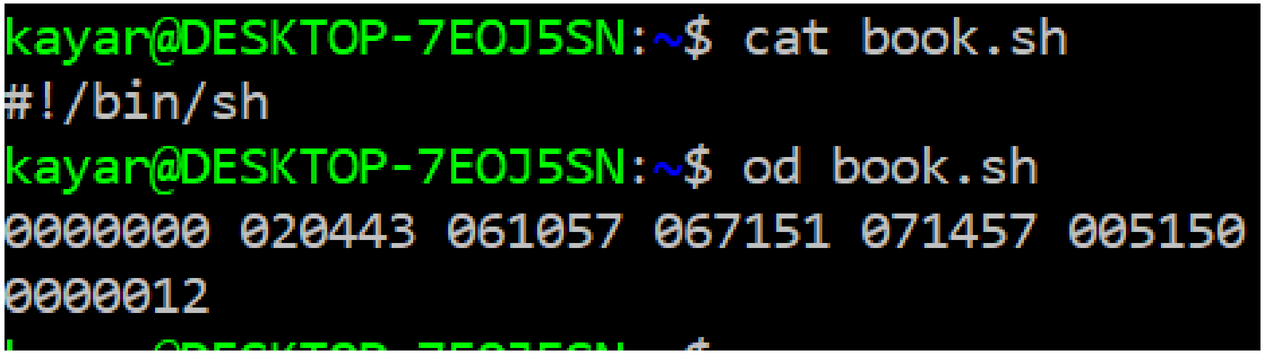
```
kayar@DESKTOP-7EOJ5SN:~$ file *
book.sh:       POSIX shell script, ASCII text executable
chap01:        ASCII text
chap02:        ASCII text
dept.txt:      ASCII text
infile:        ASCII text
kayar:         directory
namelist1.txt: ASCII text
namelist2.txt: ASCII text
one.pdf:       ASCII text
prog:          directory
sample:        ASCII text
simon:         directory
stack:         ASCII text
t.c:           C source, ASCII text
test1:         ASCII text
test2:         ASCII text
unixprog:      directory
```

od: Display Data in Octal

- Executable files contain nonprinting characters.
- UNIX commands don't display them properly

Example

od book.sh



```
kayar@DESKTOP-7EOJ5SN:~$ cat book.sh
#!/bin/sh
kayar@DESKTOP-7EOJ5SN:~$ od book.sh
0000000 020443 061057 067151 071457 005150
0000012
```

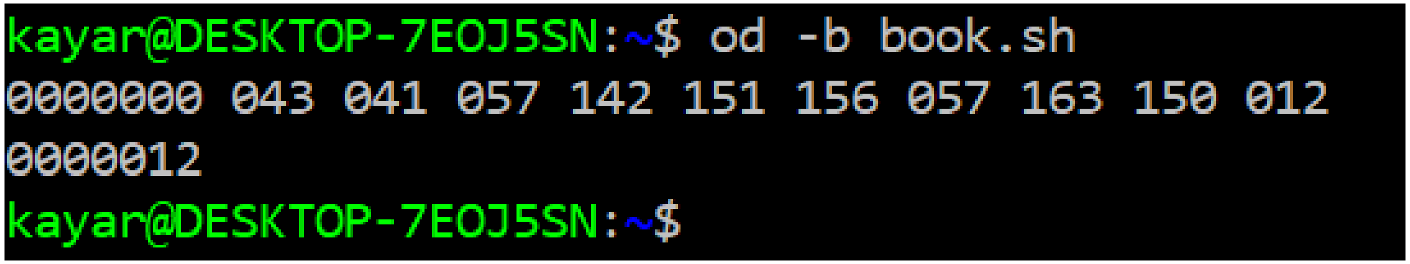
Each line displays 16 bytes of data in octal, preceded by the offset in the file of the first byte in the line

od: Display Data in Octal

- The –b option displays octal values for each character separately

Example

od –b book.sh



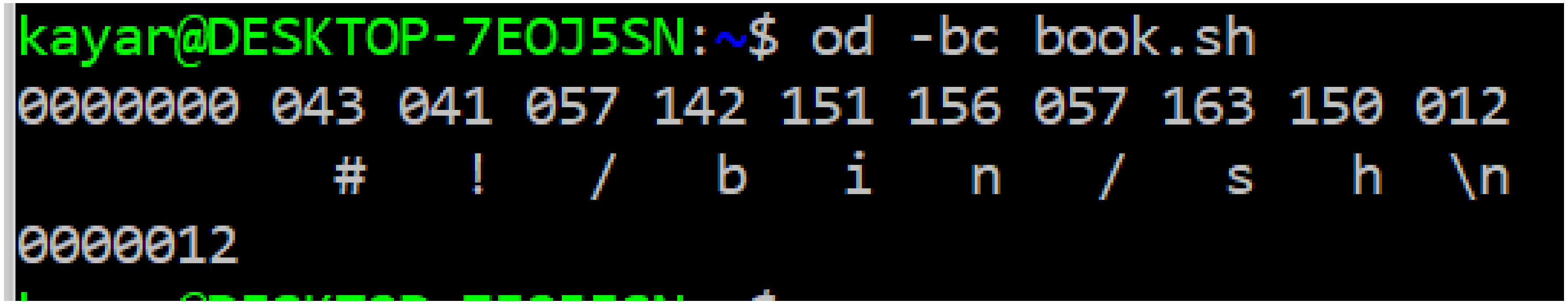
```
kayar@DESKTOP-7EOJ5SN:~$ od -b book.sh
0000000 043 041 057 142 151 156 057 163 150 012
0000012
kayar@DESKTOP-7EOJ5SN:~$
```

od: Display Data in Octal

- The –bc option displays octal value and its corresponding printable character

Example

od –bc book.sh



```
kayar@DESKTOP-7EOJ5SN:~$ od -bc book.sh
0000000 043 041 057 142 151 156 057 163 150 012
          #   !   /   b   i   n   /   s   h  \n
0000012
```

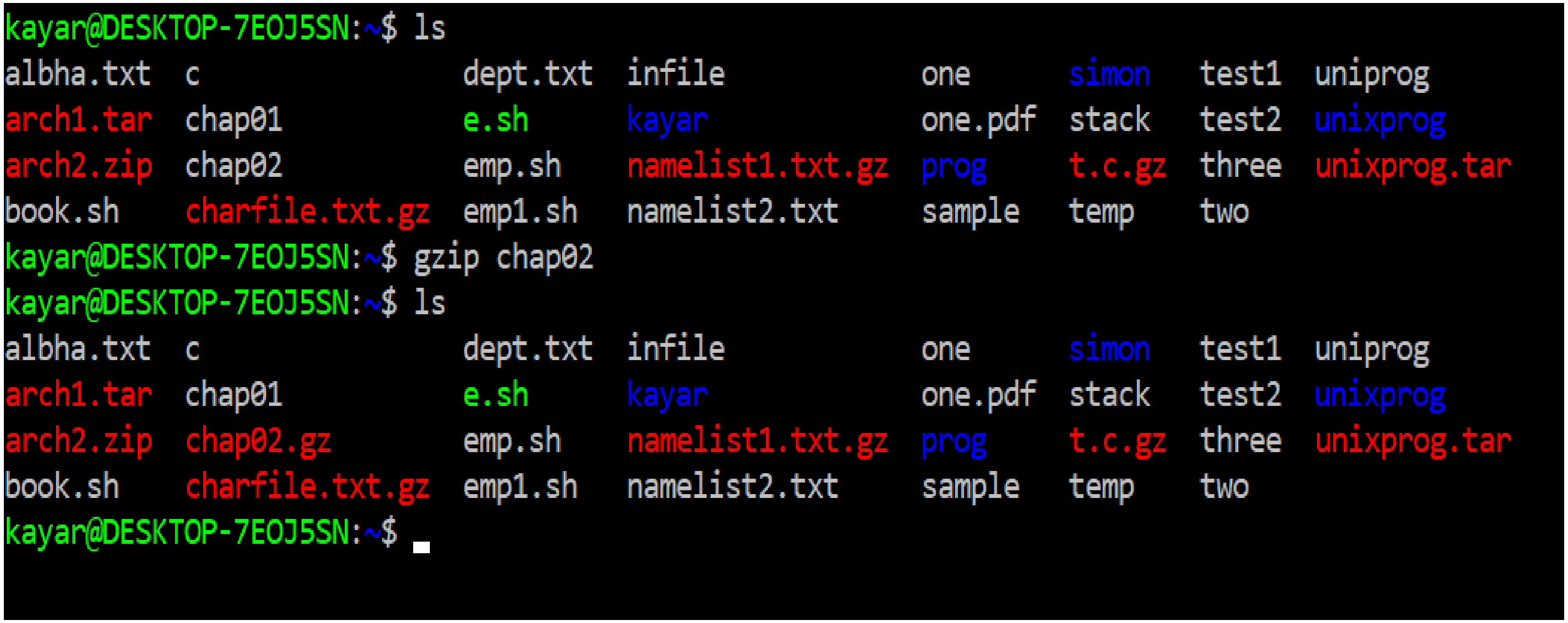
Compressing and Archiving Files

- UNIX system provides compression and decompression utilities
  - gzip and guzip (.gz)
  - Bzip2 and bunzip2(.bz2)
  - zip and unzip(.zip)

gzip AND gunzip
Compressing and Decompressing Files

gzip chap02

gzip stack test1



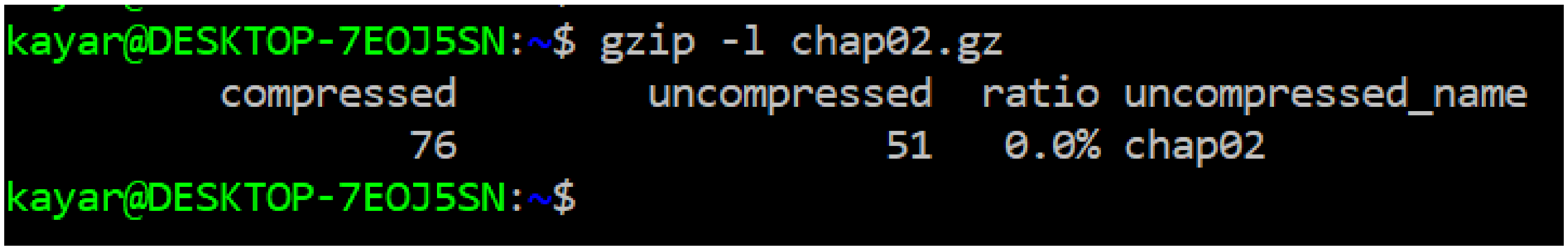
```
kayar@DESKTOP-7EOJ5SN:~$ ls
albha.txt  c                dept.txt  infile            one      simon   test1  uniprog
arch1.tar  chap01           e.sh      kayar             one.pdf  stack   test2  unixprog
arch2.zip  chap02           emp.sh    namelist1.txt.gz  prog     t.c.gz  three  unixprog.tar
book.sh    charfile.txt.gz  emp1.sh   namelist2.txt     sample   temp    two
kayar@DESKTOP-7EOJ5SN:~$ gzip chap02
kayar@DESKTOP-7EOJ5SN:~$ ls
albha.txt  c                dept.txt  infile            one      simon   test1  uniprog
arch1.tar  chap01           e.sh      kayar             one.pdf  stack   test2  unixprog
arch2.zip  chap02.gz        emp.sh    namelist1.txt.gz  prog     t.c.gz  three  unixprog.tar
book.sh    charfile.txt.gz  emp1.sh   namelist2.txt     sample   temp    two
kayar@DESKTOP-7EOJ5SN:~$ _
```

gzip AND gunzip

- Use –l option to find compression and un-compression ratio

Example

gzip – l chap02.gz



```
kayar@DESKTOP-7EOJ5SN:~$ gzip -l chap02.gz
         compressed        uncompressed  ratio uncompressed_name
                 76                  51   0.0% chap02
kayar@DESKTOP-7EOJ5SN:~$
```

gzip AND gunzip

gzip options

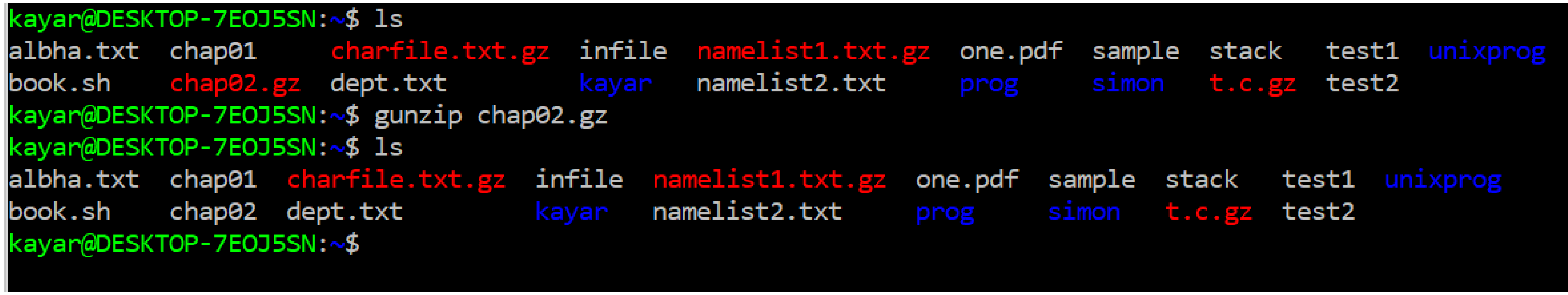
- Uncompressing a “gzipped” File (-d)

Example

gunzip chap02.gz

(Or)

gzip -d namelist1.gz



```
kayar@DESKTOP-7EOJ5SN:~$ ls
albha.txt   chap01      charfile.txt.gz  infile   namelist1.txt.gz  one.pdf  sample  stack    test1  unixprog
book.sh     chap02.gz   dept.txt         kayar    namelist2.txt     prog     simon   t.c.gz   test2
kayar@DESKTOP-7EOJ5SN:~$ gunzip chap02.gz
kayar@DESKTOP-7EOJ5SN:~$ ls
albha.txt   chap01   charfile.txt.gz  infile   namelist1.txt.gz  one.pdf  sample  stack    test1  unixprog
book.sh     chap02   dept.txt         kayar    namelist2.txt     prog     simon   t.c.gz   test2
kayar@DESKTOP-7EOJ5SN:~$
```

gzip AND gunzip

- Recursive Compression (-r)

Example

gzip -r unixprog

- Recursive Un-Compression (-r)

Example

gunzip -r unixprog

(or)

gzip -dr unixprog

tar: The Archival Program

- **tar** command is used to create a disk achieve that contains a group of files

Options

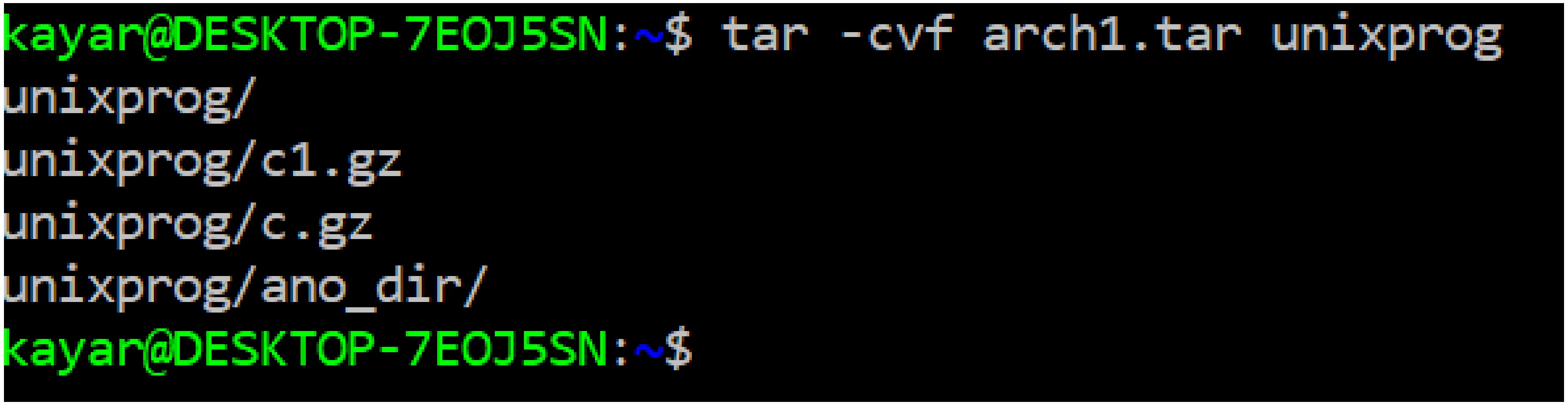
-c	Create an archive
-x	Extract files from archive
-t	Display files in archive
-f arch	Specify the archive arch

tar: The Archival Program

- Creating an Archive (-c)

Example

tar -cvf arch1.tar unixprog



```
kayar@DESKTOP-7EOJ5SN:~$ tar -cvf arch1.tar unixprog
unixprog/
unixprog/c1.gz
unixprog/c.gz
unixprog/ano_dir/
kayar@DESKTOP-7EOJ5SN:~$
```

tar: The Archival Program

- Extracting files from Archive (-x)

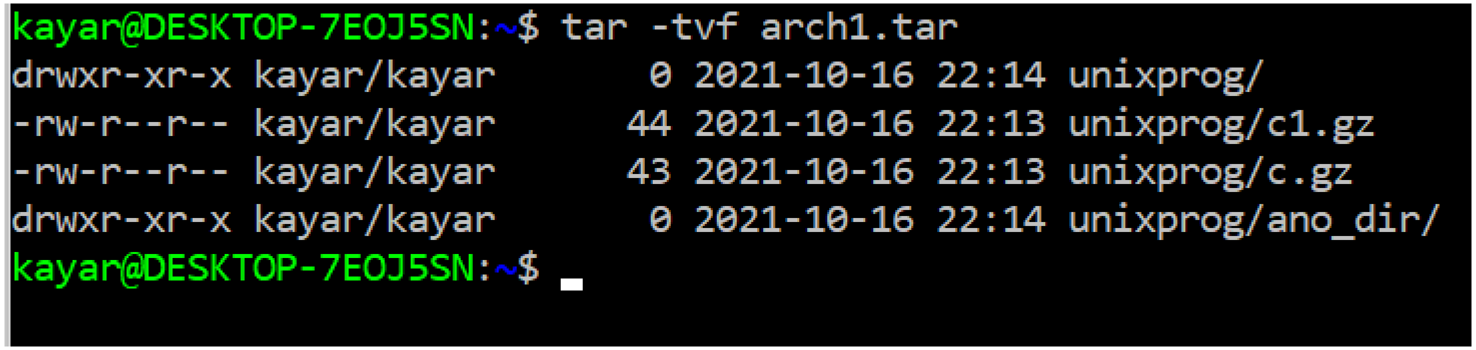
Example

tar -xvf arch1.tar

- Viewing the Archive (-t)

Example

tar –tvf arch1.tar



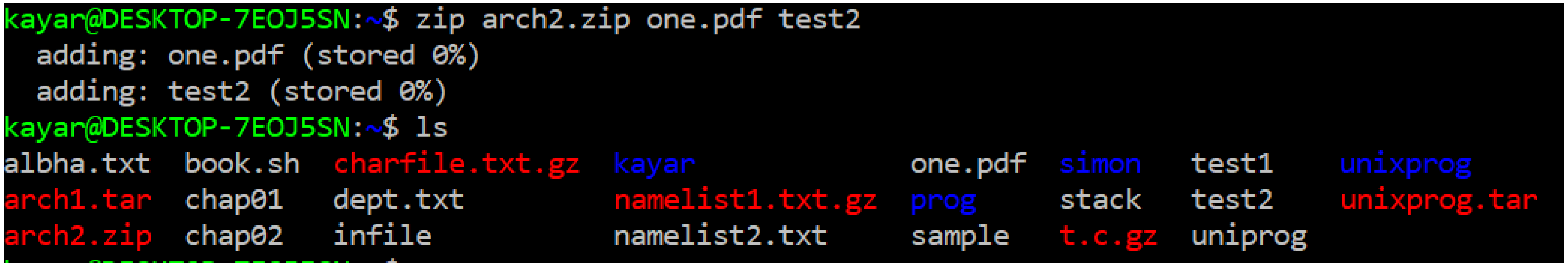
```
kayar@DESKTOP-7EOJ5SN:~$ tar -tvf arch1.tar
drwxr-xr-x kayar/kayar       0 2021-10-16 22:14 unixprog/
-rw-r--r-- kayar/kayar      44 2021-10-16 22:13 unixprog/c1.gz
-rw-r--r-- kayar/kayar      43 2021-10-16 22:13 unixprog/c.gz
drwxr-xr-x kayar/kayar       0 2021-10-16 22:14 unixprog/ano_dir/
kayar@DESKTOP-7EOJ5SN:~$ _
```

zip and unzip
Compressing and Archiving Together

- Zip combines the compressing function of gzip with the archival function of tar

Example

zip arch2.zip one.pdf test2



```
kayar@DESKTOP-7EOJ5SN:~$ zip arch2.zip one.pdf test2
  adding: one.pdf (stored 0%)
  adding: test2 (stored 0%)
kayar@DESKTOP-7EOJ5SN:~$ ls
albha.txt  book.sh   charfile.txt.gz  kayar             one.pdf  simon   test1    unixprog
arch1.tar  chap01    dept.txt         namelist1.txt.gz  prog     stack   test2    unixprog.tar
arch2.zip  chap02    infile           namelist2.txt     sample   t.c.gz  uniprog
```

zip and unzip

- Files are restored with unzip command

```
unzip arch2.zip
```

- Viewing the Archive (-v)

```
unzip –v archi2.zip
```

lp: Printing a File

- **lp:** submits files for printing or alters a pending job.