SECURITY AND FILE PERMISSIONS

MODULE 2

USERS AND GROUPS

- Everyone who logs on to the system is called **user**. Users are known to the system by their **user-ids**.
- Superusers have the maximum set of capabilities in the system, they can even change the system itself.
- Users can be organized into groups. The team members can have easy access to each others files while still protecting the files from users outside the group.
- Users can belong to multiple groups.

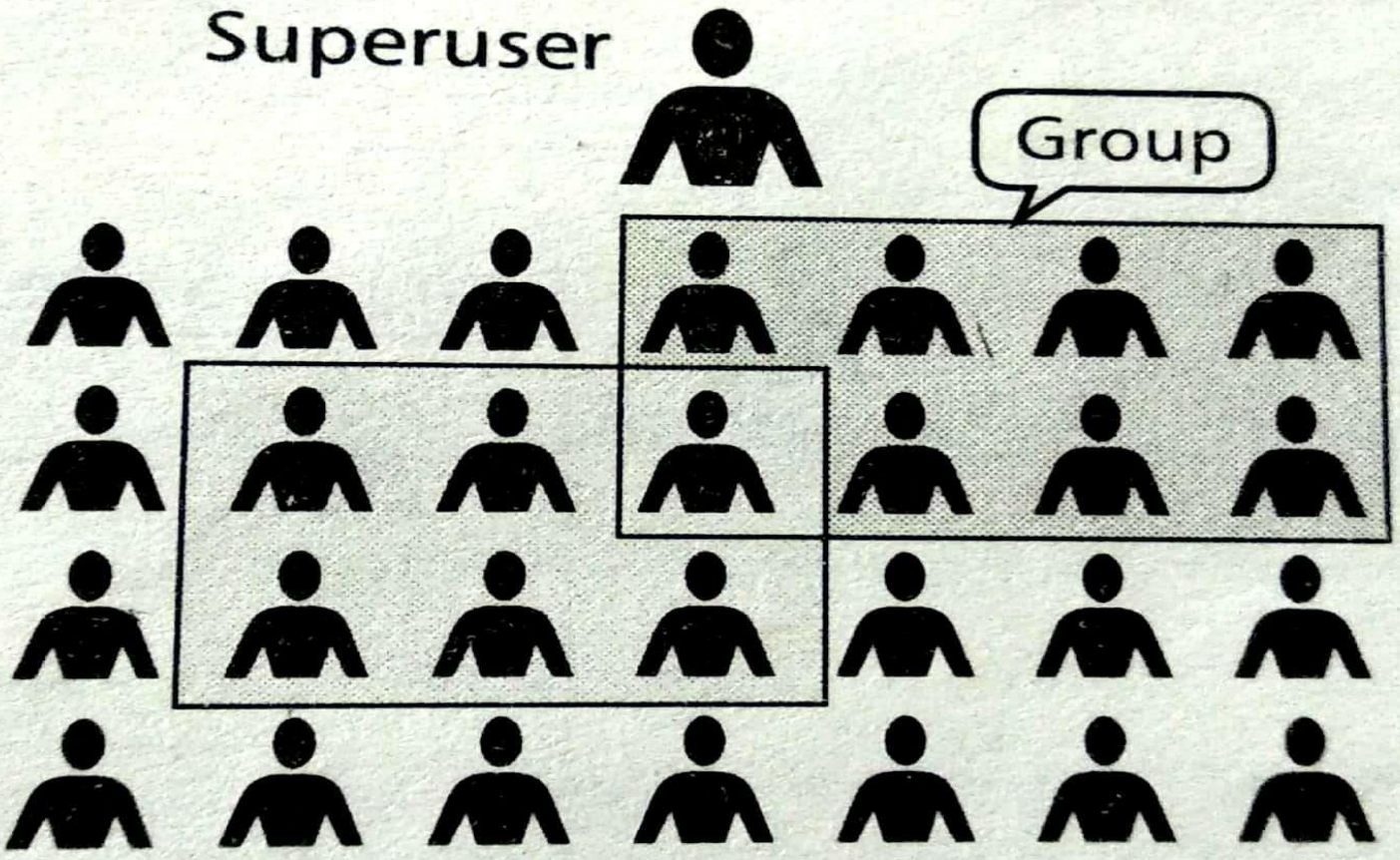


FIGURE 4.1 Users

GROUP (groups) COMMAND

- UNIX provides a command, **groups**, to determine a user's group.
- If you enter the command with no user id, the system responds with your group.
- If you enter the command with user id, it returns the user's group.

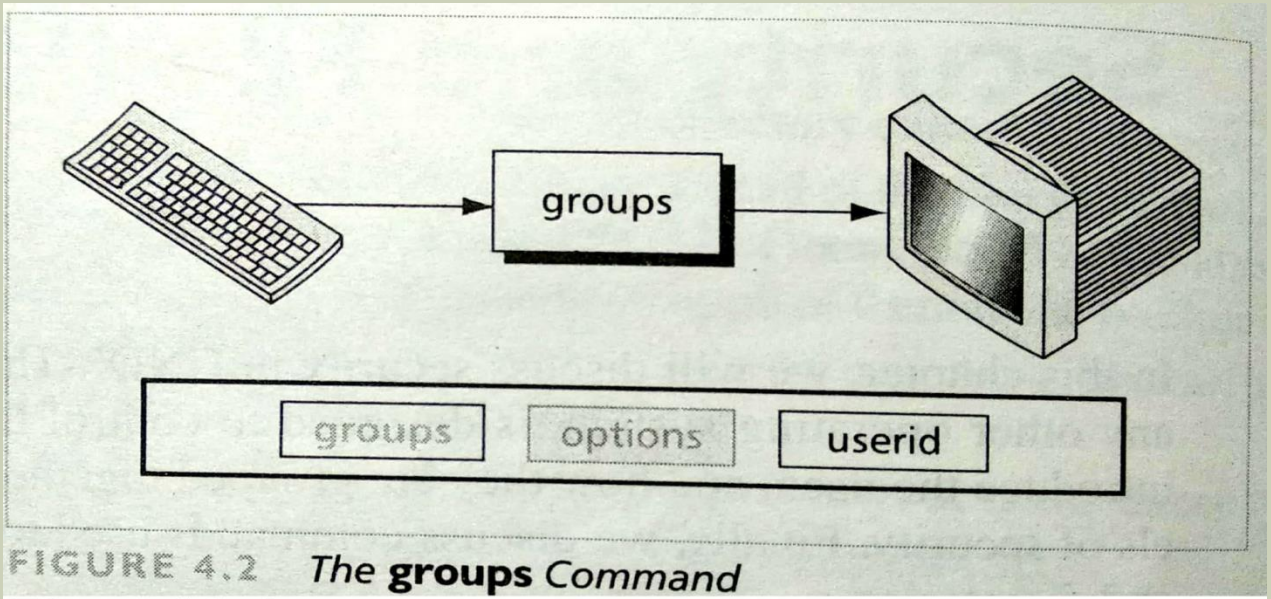


FIGURE 4.2 *The* **groups** *Command*

SECURITY LEVELS

- There are three levels of security in UNIX : system, directory, and file.
- The system security is controlled by the system admin, super user.
- The directory and file securities are controlled by the users who own them.

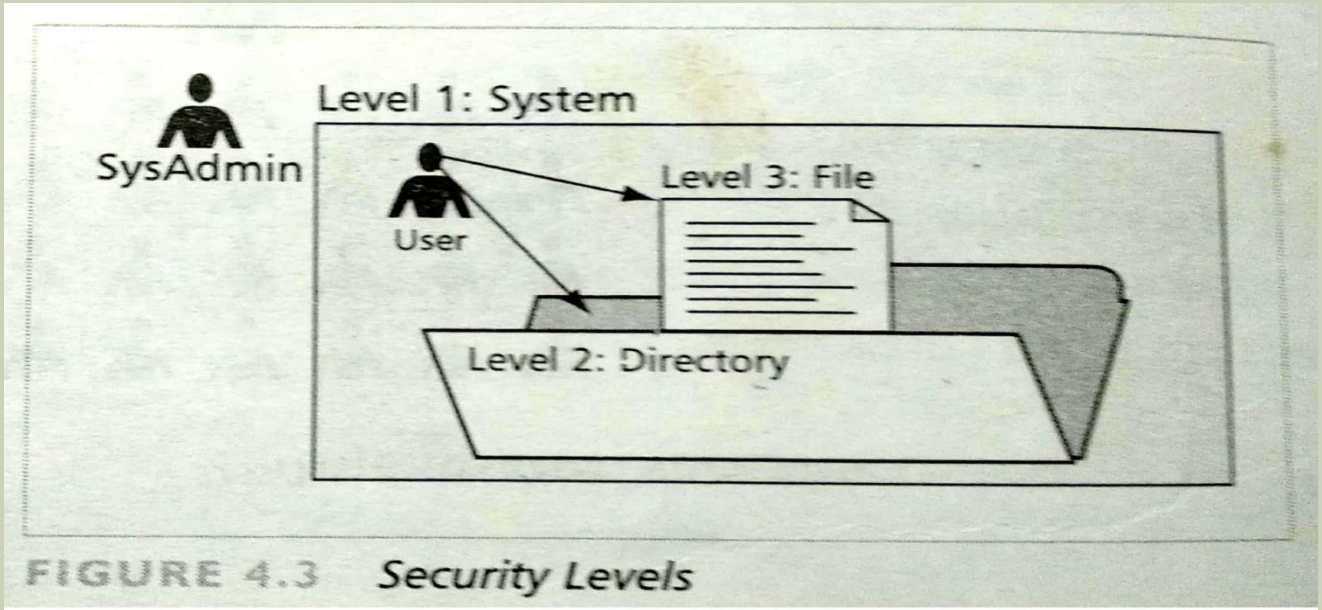


FIGURE 4.3 *Security Levels*

System Security

- System security controls who is allowed to access the system. It begins with your login id and password.
- The file, named /etc/passwd, is located in the etc directory and contains several important pieces of information about user.

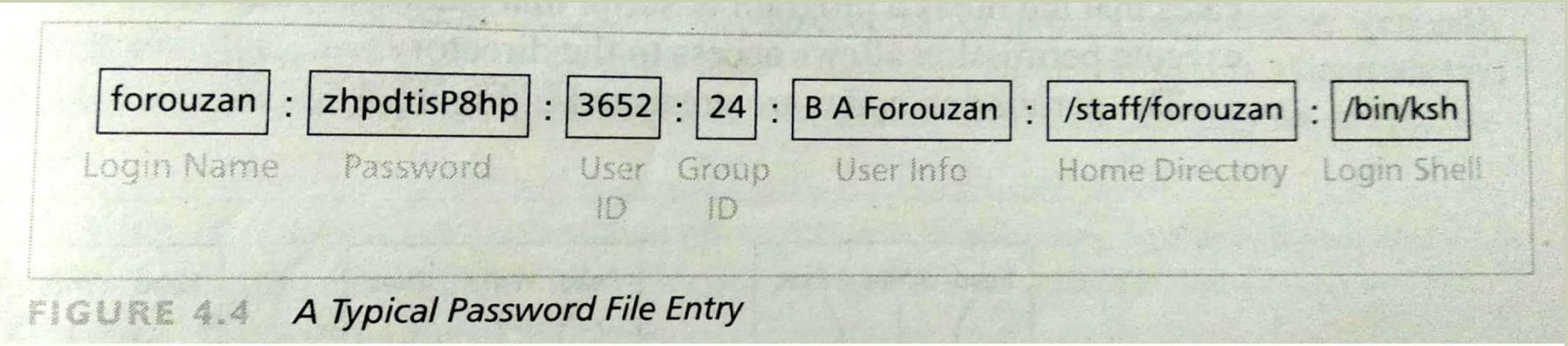


FIGURE 4.4 *A Typical Password File Entry*

- **Login Name** :- It uniquely identifies you as one of the users.
- **Password** :- the one-way encrypted password that identifies you to the system.
- **User id** :- It is a unique number between 0 and 65,535. user id zero is reserved for the superuser.
- **Group id** :- is a unique number between 100 and 65,535 that identifies users who have common access.
- **User information** :- is used to store data about the user.
- **Home Directory** :- the login or home directory when you first log into the system.
- **Login shell** :- identifies the shell that is loaded when you login.

Permission Codes

- Both the directory and file security levels use a set of permission codes to determine who can access and manipulate a directory or file.
- The permission codes are divided into three sets of codes.
- **First**, set contains the permission of the owner of the directory or file.
- **Second**, set contains the group permissions for members in a group as identified by the group id.
- **Third**, set contains the permission for everyone else that is ,the general public.

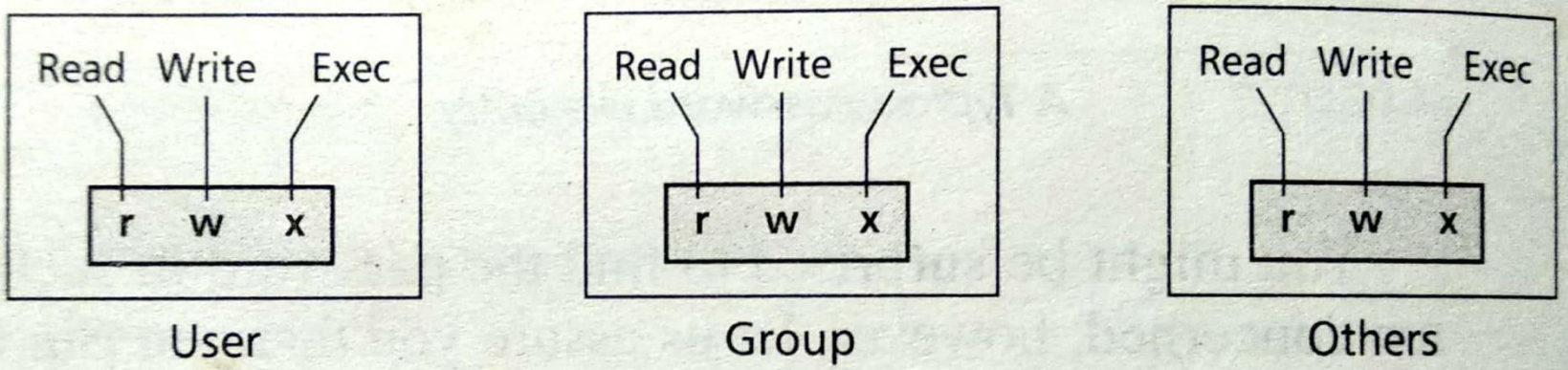


FIGURE 4.5 *Directory and File Permissions*

TABLE 4.1 *Summary of Permission Rules*

Permission	read (r)	write (w)	execute (x)
Directory	Read contents of directory	Add or delete entries (files) in directory using commands	Reference or move to directory
File Level	Read or copy files in directory	Change or delete files	Run executable files

Directory Level Permission

- **Read Permission** :- When users have read permission for a directory, they can read the directory, which contains the names of the files and subdirectories and all of their attributes.
- **Write Permission** :- When users have write permission, they can add or delete entries in a directory. This means that they can copy a file from another directory, move a file to or from the directory, or remove (delete) a file.
- **Execute Permission** :- This is sometimes called search permission, at the directory level allows you to reference a directory, as in pathname or file read, or move to a directory using the cd command. The user permissions for directories, therefore, generally include both read and execute.

File Level Permission

- **Read Permission** :- Users who have file read permission can read or copy a file. Private files, however , should be read only by the user (owner).
- **Write Permission** :- Files with write permission can be changed. They can also be deleted.
- **Execute Permission** :- With files, execute permission means that you can execute (run) programs, utilities, and scripts.

Checking Permission :- To check the permission of a file or directory, we use the long list command.

CHANGING PERMISSIONS

- When a directory or a file is created, the system automatically assigns default permissions. The owner of the directory or file can change them.
- To change the permissions, we use the **chmod** command.

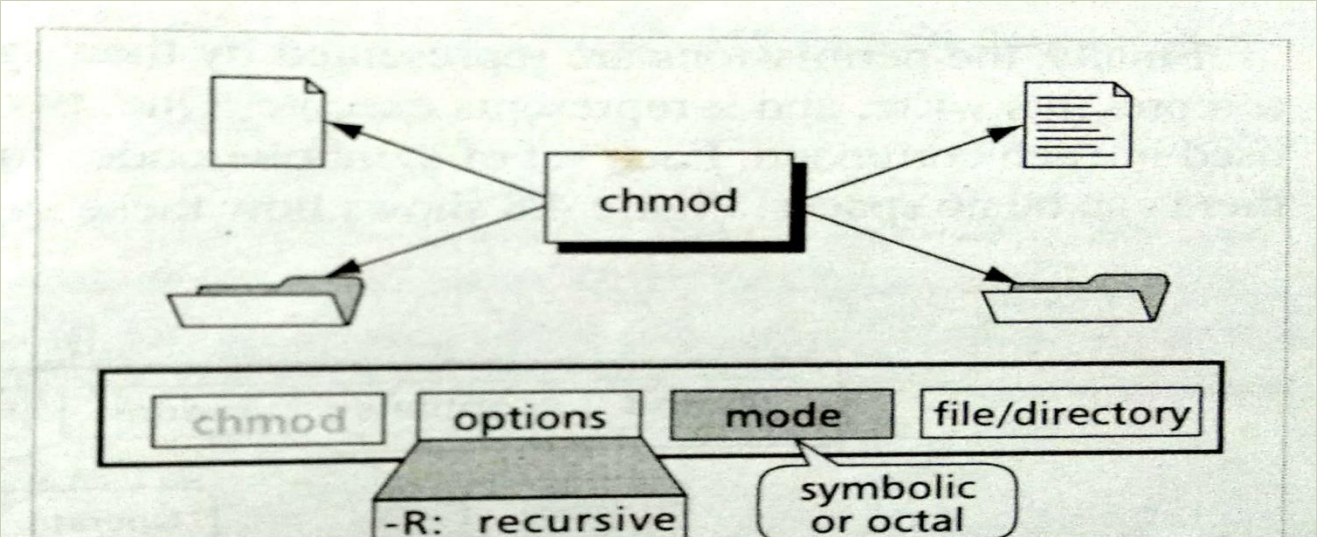


FIGURE 4.6 *The* **chmod** *Command*

- There are two ways to change the permissions :- Symbolic and Octal.

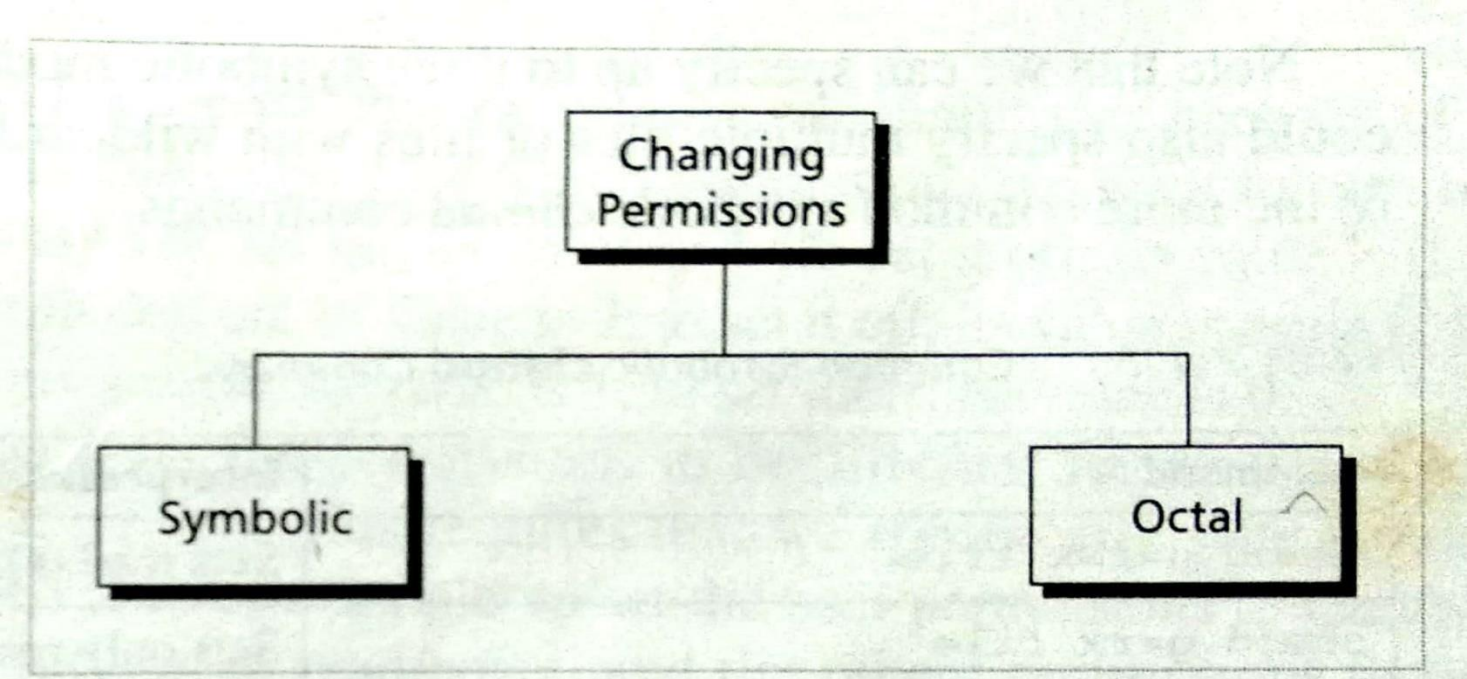
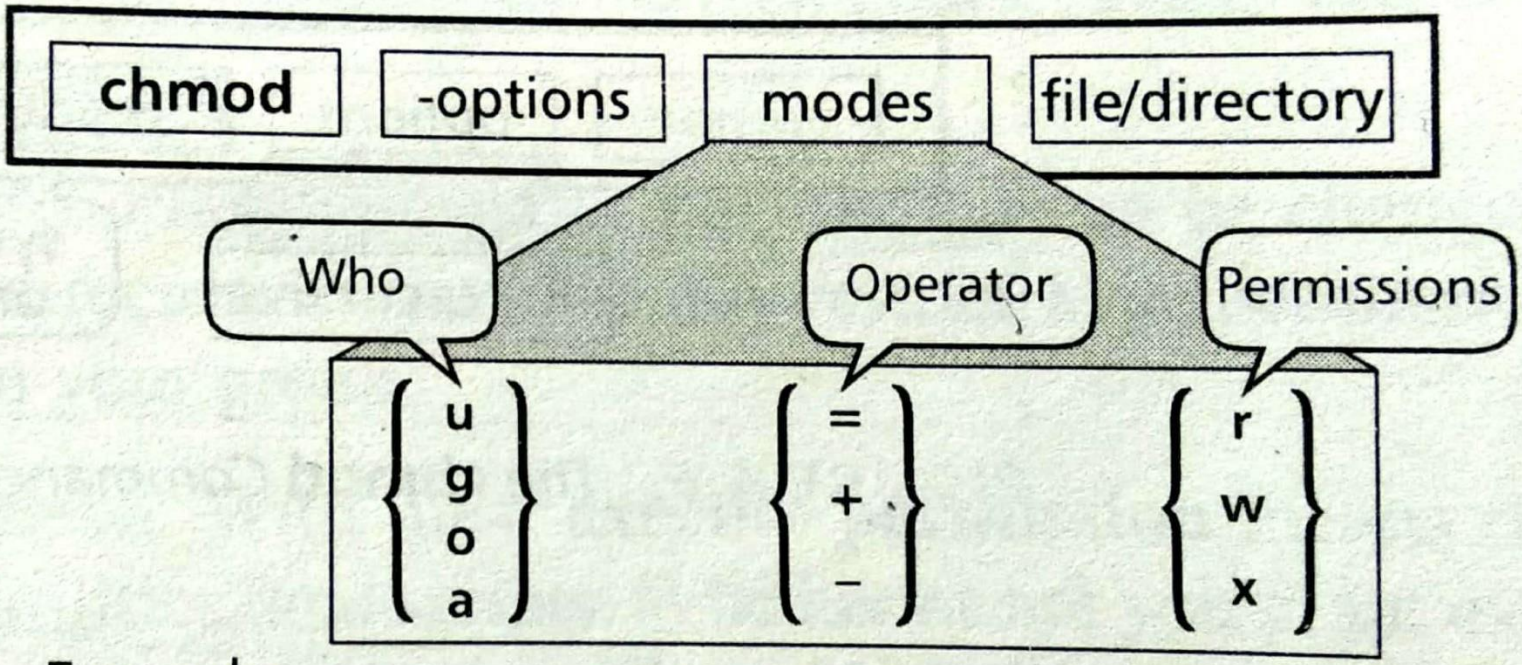


FIGURE 4.7 *Changing Permissions*

Symbolic Codes

- There are three sets of permissions: user, group, and other.
- Each set uses its first letter as a mnemonic identifier. Thus, **u** represents user, **g** represents group, and **o** represents others.
- There are three sets of operators. The assignment operator (=), this provides current permissions for a set are replaced by the new permissions.
- To change only one or two of the permissions in a set and leave the others as they are currently set, we use a plus sign (+) to add permissions.
- To remove one or two permissions and leave the others alone, we use a minus sign (-).



Example

```
chmod   u=rwx,g+w,o-w   memo.doc
```

FIGURE 4.8 *Symbolic* **chmod** *Codes*

TABLE 4.2 *Common Symbolic* **chmod** *Commands*

Command	Interpretation
`chmod u=rwx file`	Sets read (**r**), write (**w**), execute (**x**) for user.
`chmod g=rx file`	Sets only read (**r**) and execute (**x**) for group; write (**w**) denied.
`chmod g+x file`	Adds execute (**x**) permission for group; read and write unchanged.
`chmod a+r file`	Adds read (**r**) to all users; write and execute unchanged.
`chmod o-w file`	Removes others' write (**w**) permission; read and execute unchanged.

Octal Codes

- A faster way to enter permission is to use the octal equivalent of the codes.
- It is not like the symbolic modes where you need to specify only what you want to change.
- In an octal digit, there are three bit positions. The three different permissions for each set of codes correspond to the three different bit positions in an octal digit.
- The first bit represent read permission, second bit represent write permission, third bit represent execute permission.

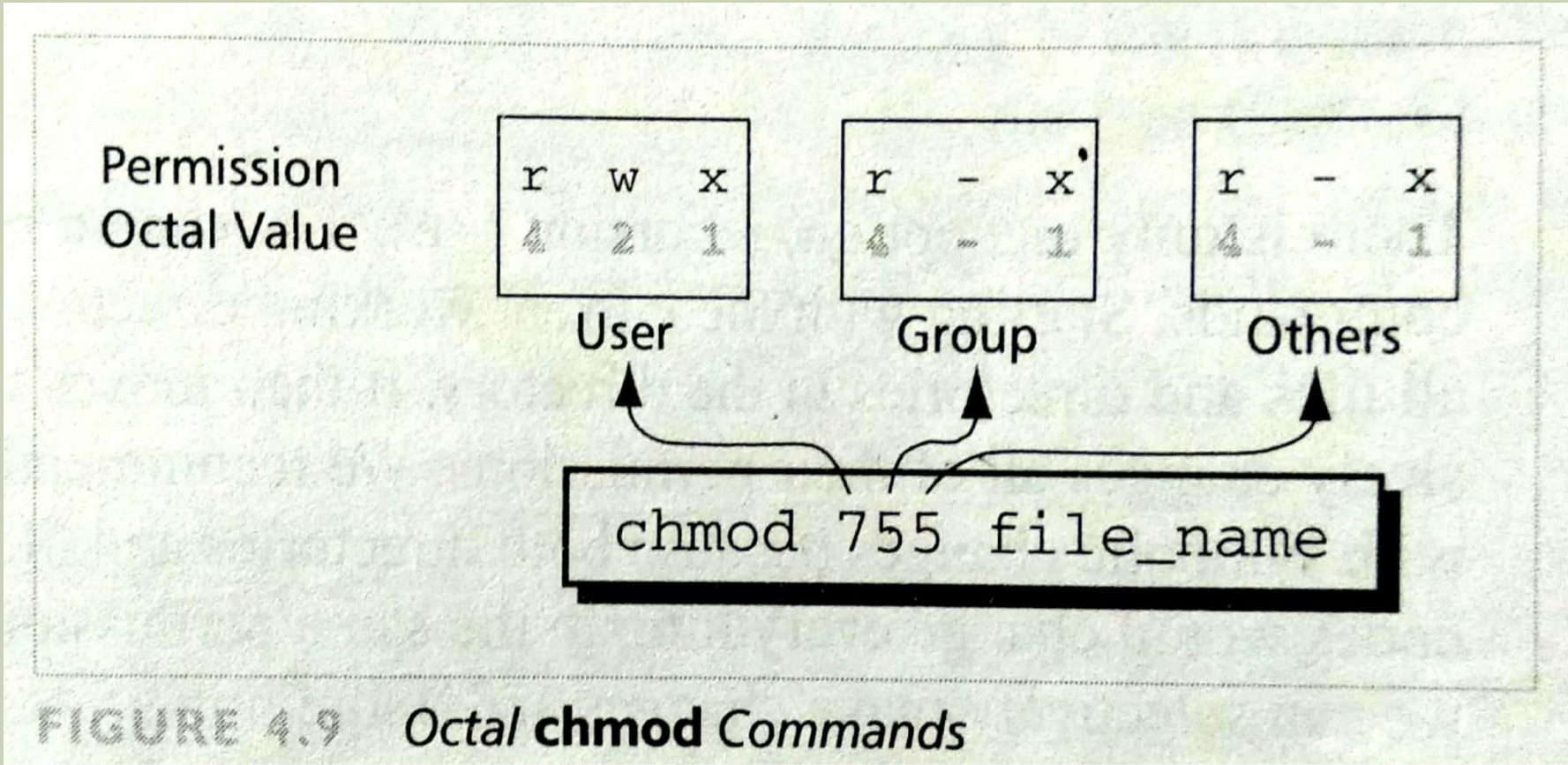


FIGURE 4.9 *Octal* **chmod** *Commands*

TABLE 4.3 *Common Symbolic* **chmod** *Commands*

Command	Description
`chmod 777 file`	All permissions on for all three settings
`chmod 754 directory`	User all, group read + execute; others read only
`chmod 664 file`	User and group read + write, others read only
`chmod 644 file`	User read + write, group and others read only
`chmod 711 program`	User all, group and others execute only

Option

- There is only one option, recursion (-R). The chmod recursion works just as in other commands. Starting with the current working directory, it changes the permissions of all files and directories in the directory.
- It then moves to the subdirectories and recursively changes all of their permissions.

USER MASKS

Basic Concept

- The permission are initially set for a directory or file using a three-digit octal system variable, the user mask.
- Defined initially by the system administrator when your account is created , the mask contains the octal settings for the permissions that are to be removed from the default when the directory or file is created.
- The default permission are **777 for a directory** and **666 for a file.**

TABLE 4.4 *User Mask Results*

mask	Directory Permission (Default 777)	File Permission (Default 666)[a]
0	7 (read/write/execute)	6 (read/write)
1	6 (read/write)	6 (read/write)
2	5 (read/execute)	4 (read)
3	4 (read)	4 (read)
4	3 (write/execute)	2 (write)
5	2 (write)	2 (write)
6	1 (execute)	0 (none)
7	0 (none)	0 (none)

[a]File default assumes file is data file. Executable file must be manually set.

TABLE 4.5 *Examples of Default Calculations for Permissions*

Mask	Directory Permissions '(Default 777)	File Permissions (Default 666)
000 (Public)	777 (rwx rwx rwx)	666 (rw– rw– rw-)
011 (Public)	766 (rwx rw– rw–)	666 (rw– rw– rw–)
022 (Write Protected)	755 (rwx r–x r–x)	644 (rw- r-- r--)

Continued

USER MASK (umask) COMMAND

- The user mask is displayed and set with the umask command.
- To display the current user mask setting, use the umask command with no arguments.

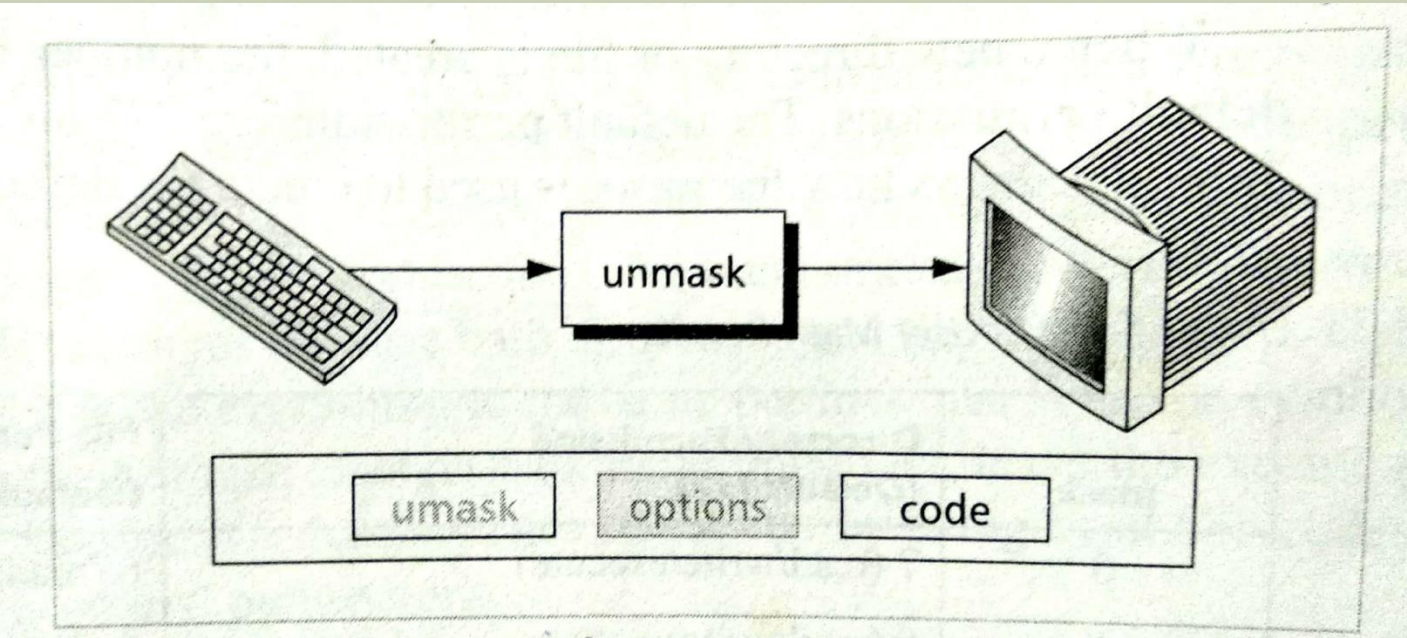


FIGURE 4.10 *The* **umask** *Command*

CHANGING OWNERSHIP AND GROUP

- When you create a directory or file, you are the owner and your group is the group.
- There are two command that allow the owner and group to be changed.
- The change ownership (chown) command can change the owner or the owner and the group.
- The change group (chgrp) command can change only the group.

CHANGE OWNERSHIP (chown) COMMAND

- The owner and optionally the group are changed with the change ownership (chown) command.
- The new owner may be a login name or used id.
- The group is optional. When it is used, it is separated from the owner by colon or period.

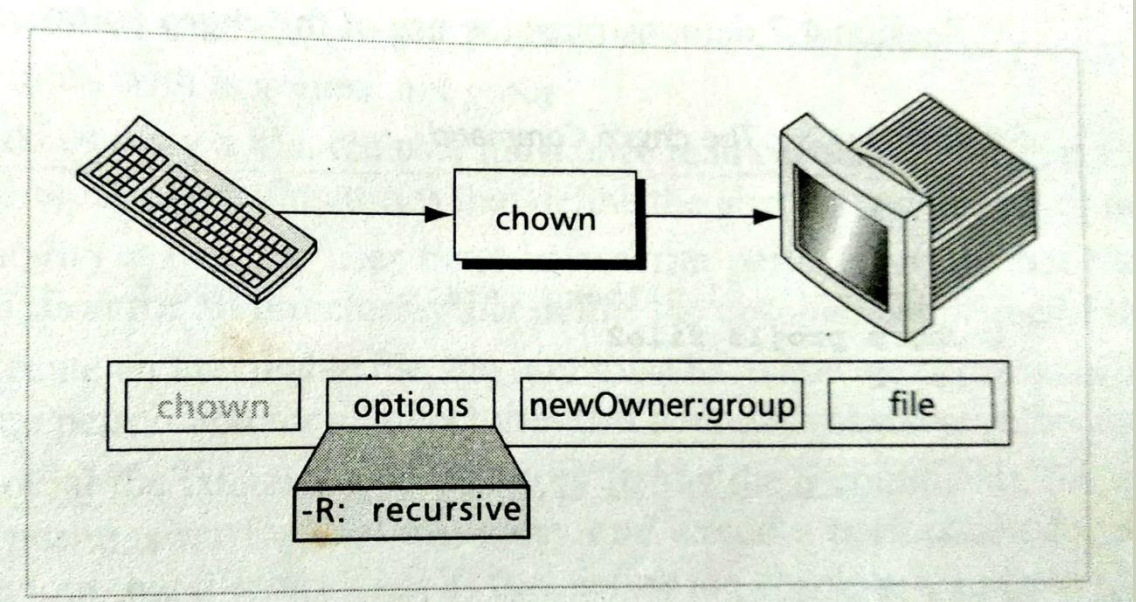


FIGURE 4.11 *The* **chown** *Command*

CHANGE GROUP (chgrp) COMMAND

- To change the group without changing the owner, you use the change group (chgrp) command.

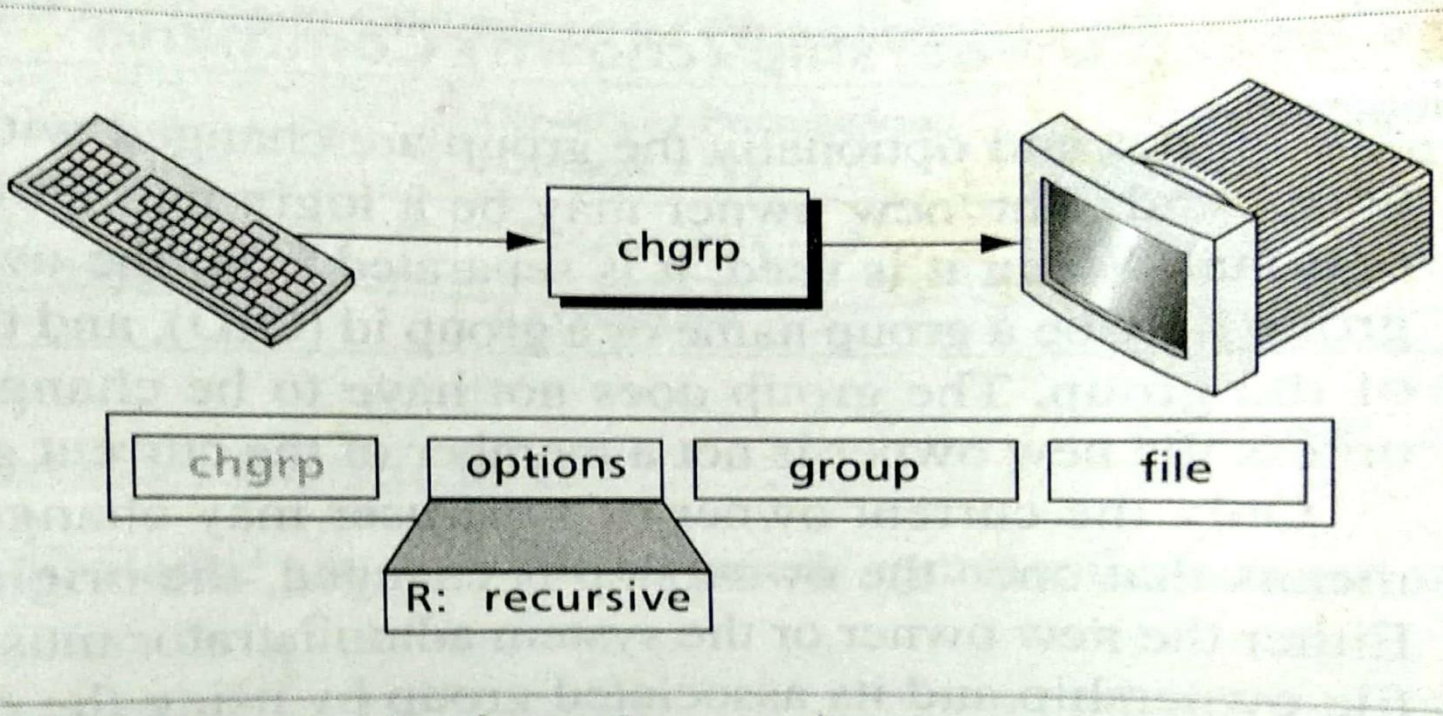


FIGURE 4.12 *The* **chgrp** *Command*

FILTERS AND PIPES

Filters

- In UNIX, a filter is any command that gets its input from the standard input stream, manipulates the input, and then sends the result to the standard output stream.
- One of the example for filter is more command.
- Three filters – grep, sed, awk.

TABLE 6.1 *Common Filters*

Filter	Action
Already Studied	
more	Passes all data from input to output, with pauses at the end of each screen of data.
Studied in This Chapter	
cat	Passes all data from input to output.
cmp	Compares two files.
comm	Identifies common lines in two files.
cut	Passes only specified columns.
diff	Identifies differences between two files or between common files in two directories.
head	Passes the number of specified lines at the beginning of the data.
paste	Combines columns.
sort	Arranges the data in sequence.
tail	Passes the number of specified lines at the end of the data.
tr	Translates one or more characters as specified.

Continued

TABLE 6.1 *Common Filters—Continued*

Filter	Action
Studied in This Chapter	
uniq	Deletes duplicate (repeated) lines.
wc	Counts characters, words, or lines.
Studied In Future Chapters	
grep	Passes only specified lines.
sed	Passes edited lines.
awk	Passes edited lines—parses lines.

FILTERS AND PIPES

- Filters work naturally with pipes. Because a filter can send its output to the monitor, it can be used on the left of a pipe.
- Because a filter can receive its input from the keyboard, it can be used on the right of a pipe.

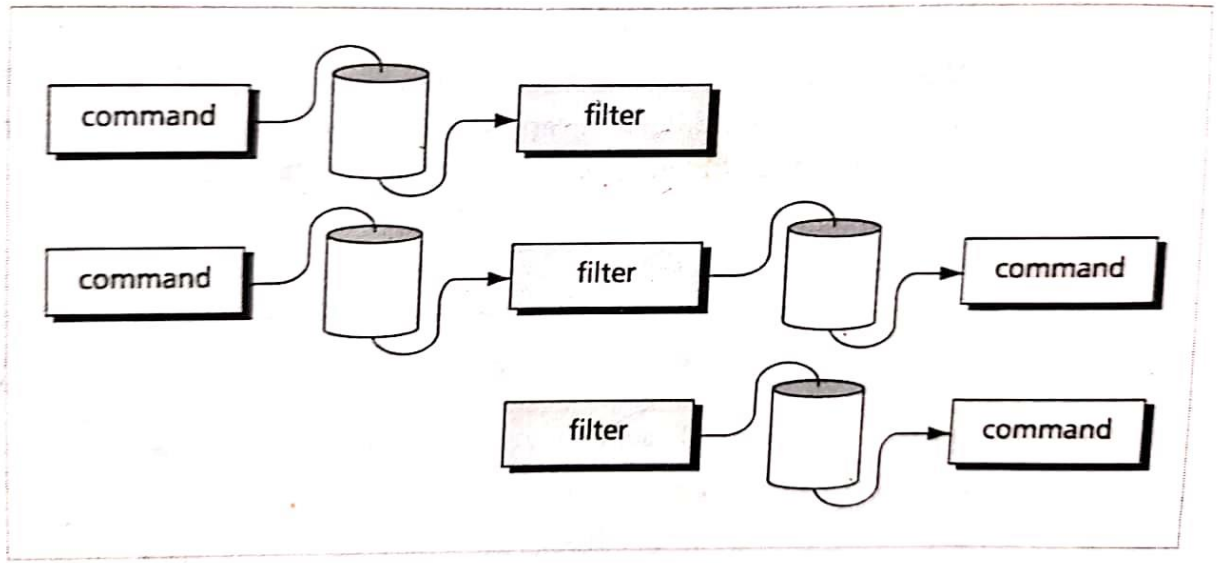


FIGURE 6.2 *Using Filters with Pipes*

CONCATENATING FILES

- UNIX provides a powerful utility to concatenate commands.
- Catenate (cat) command – The cat command writes them one after another to standard output.

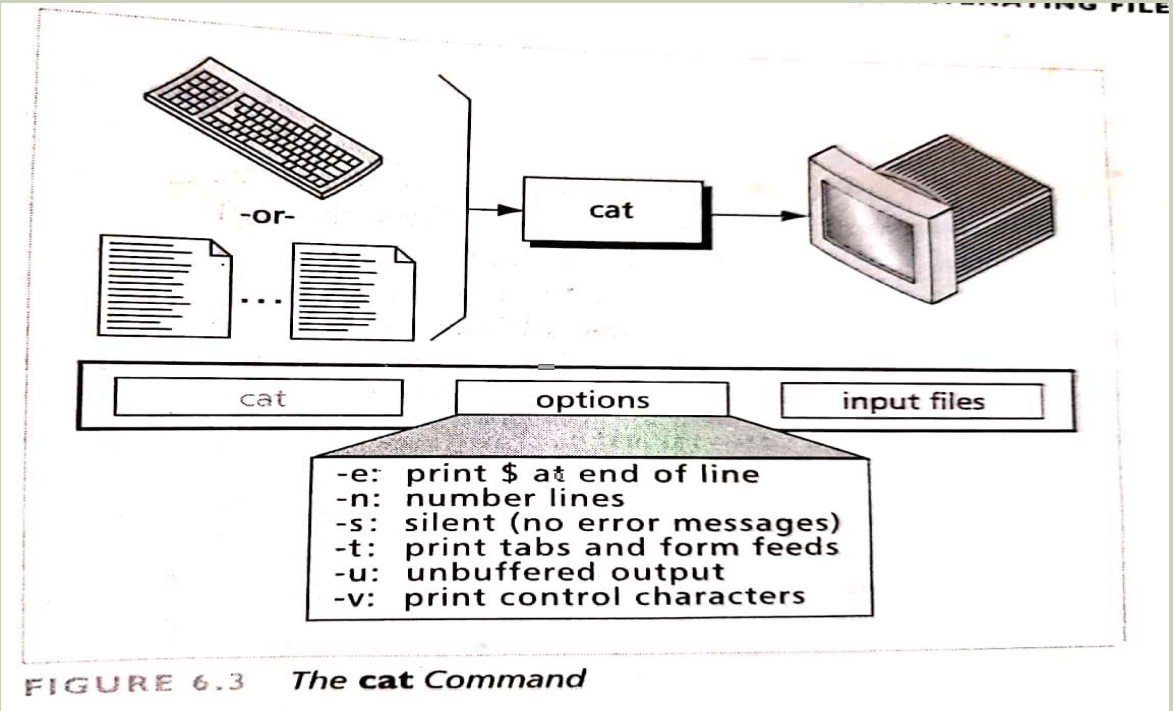


FIGURE 6.3 *The* **cat** *Command*

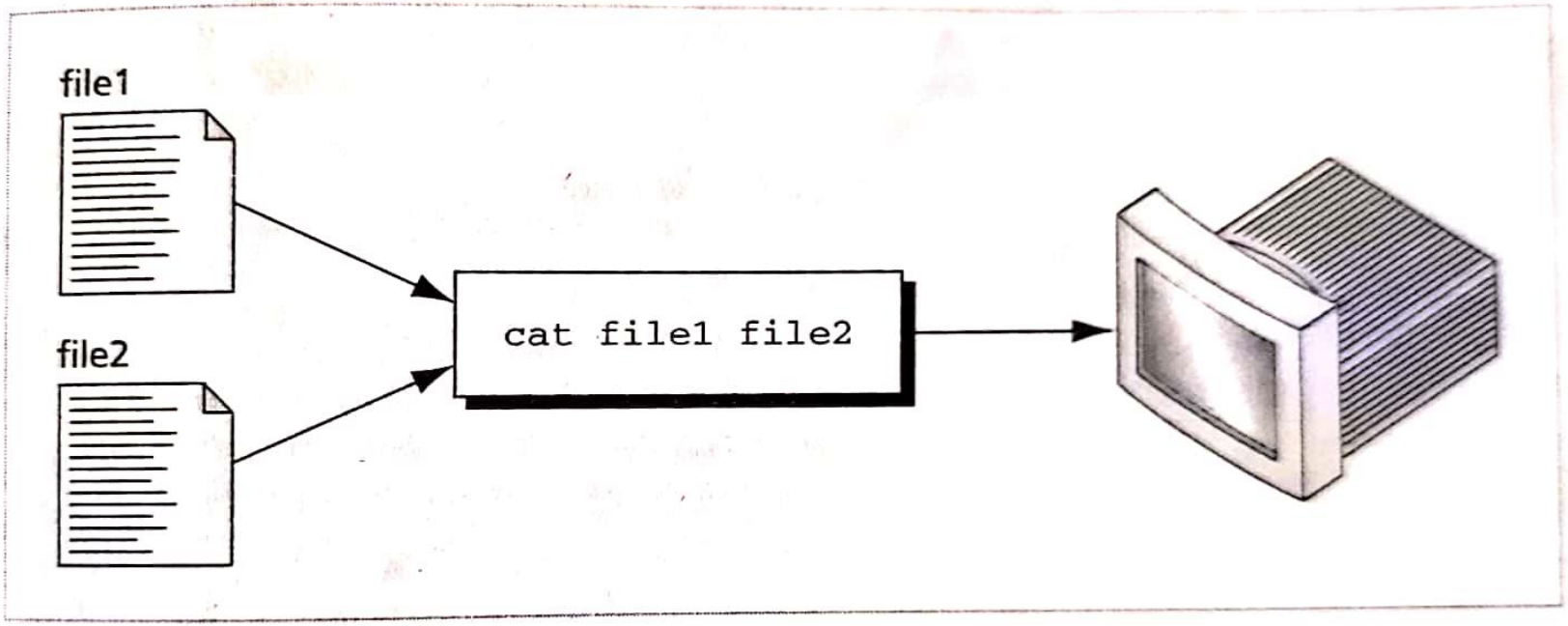


FIGURE 6.4 *Using* **cat** *to Combine Files*

USING CAT TO DISPLAY A FILE

- Its basic design makes cat a useful tool to display a file.
- First, there are no automatic pauses at the end of the screen like we saw with the more command.
- Second, because the primary use is to combine files an not to display them, there is no check to verify that the file contains text.

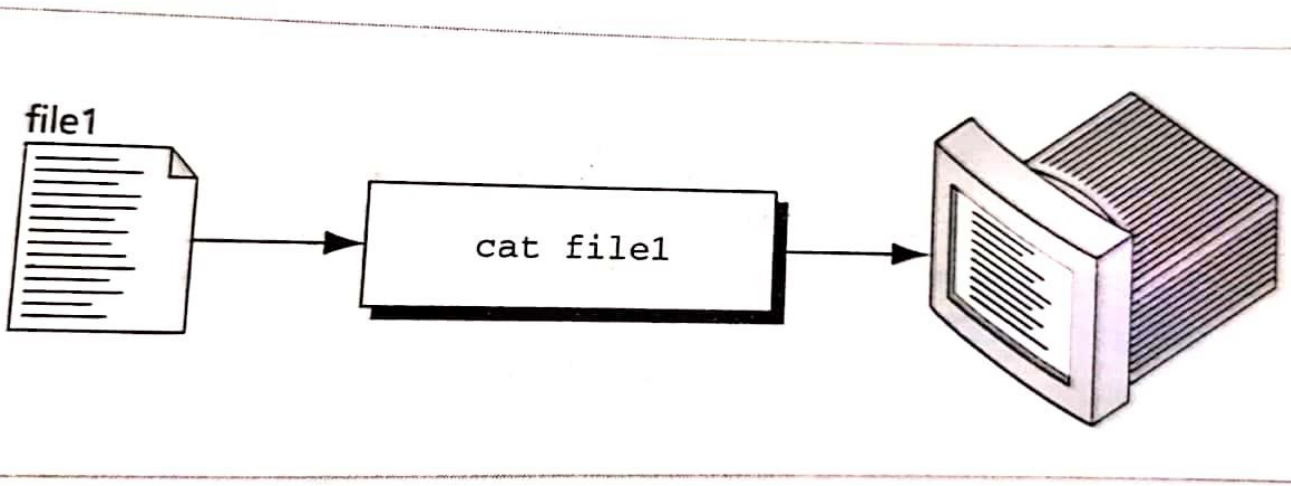


FIGURE 6.5 *Displaying a File with* **cat**

USING CAT TO CREATE A FILE

- The second special application uses cat to create a file.

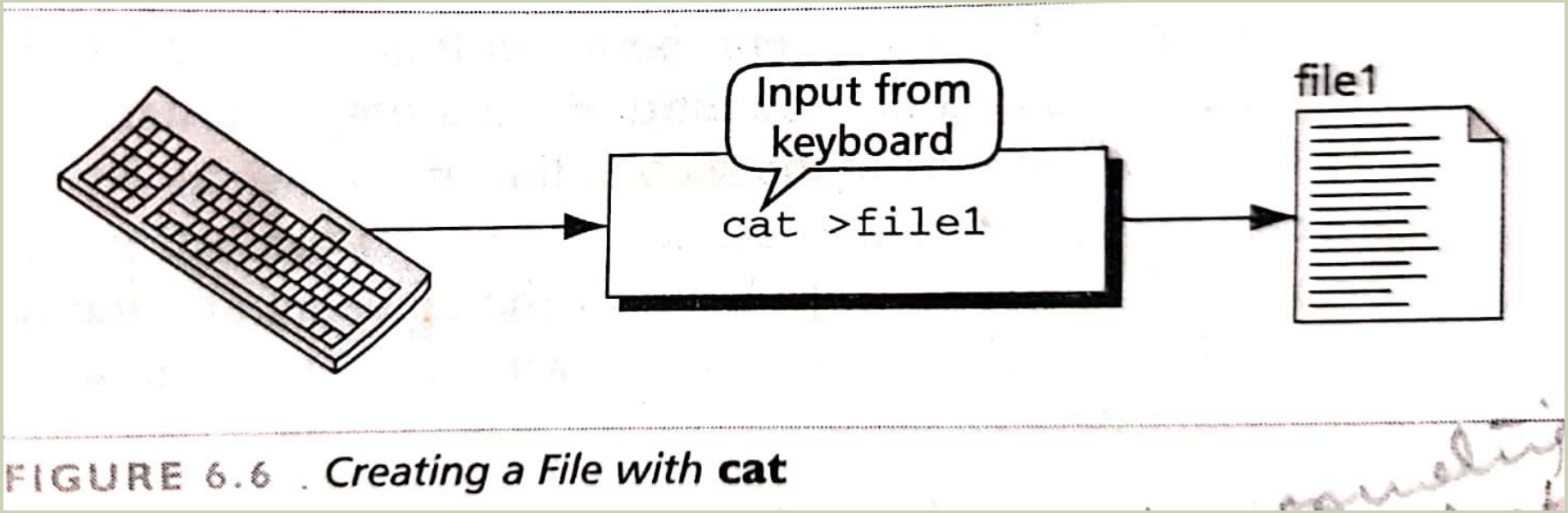


FIGURE 6.6 . *Creating a File with* **cat**

CAT OPTIONS

- There are six options available with cat. They can be grouped into four categories:-
  - Visual characters
  - Buffered output
  - Missing files
  - Numbered lines

- **Visual characters** :- The visual option (-v), allows us to see control character, with the exception of the tab, newline, and form feed characters.
- **Buffered output** :- When output is buffered, it is kept in the computer until the system has time to write it to a file. You can force output to be written to the file immediately by specifying the option –u for unbuffered.
- **Missing files** :- When you catenate several files together, if one of them is missing, the system displays a message. If you don't want to have the message in your output, you can specify that cat is to be silent when it cant find a file the option is –s.
- **Numbered lines** :- This option (-n) numbers each line in each file as the line is written to standard output.

DISPLAY BEGINNING AND END OF FILES

- **Head**
- The head command copies a specified number of lines from the beginning of one or more files to the standard output stream. If no files are specified, it gets the lines from the standard input.
- The option is used to specify the number of lines. If the number of lines is omitted, head assumes 10 lines, total file is used.

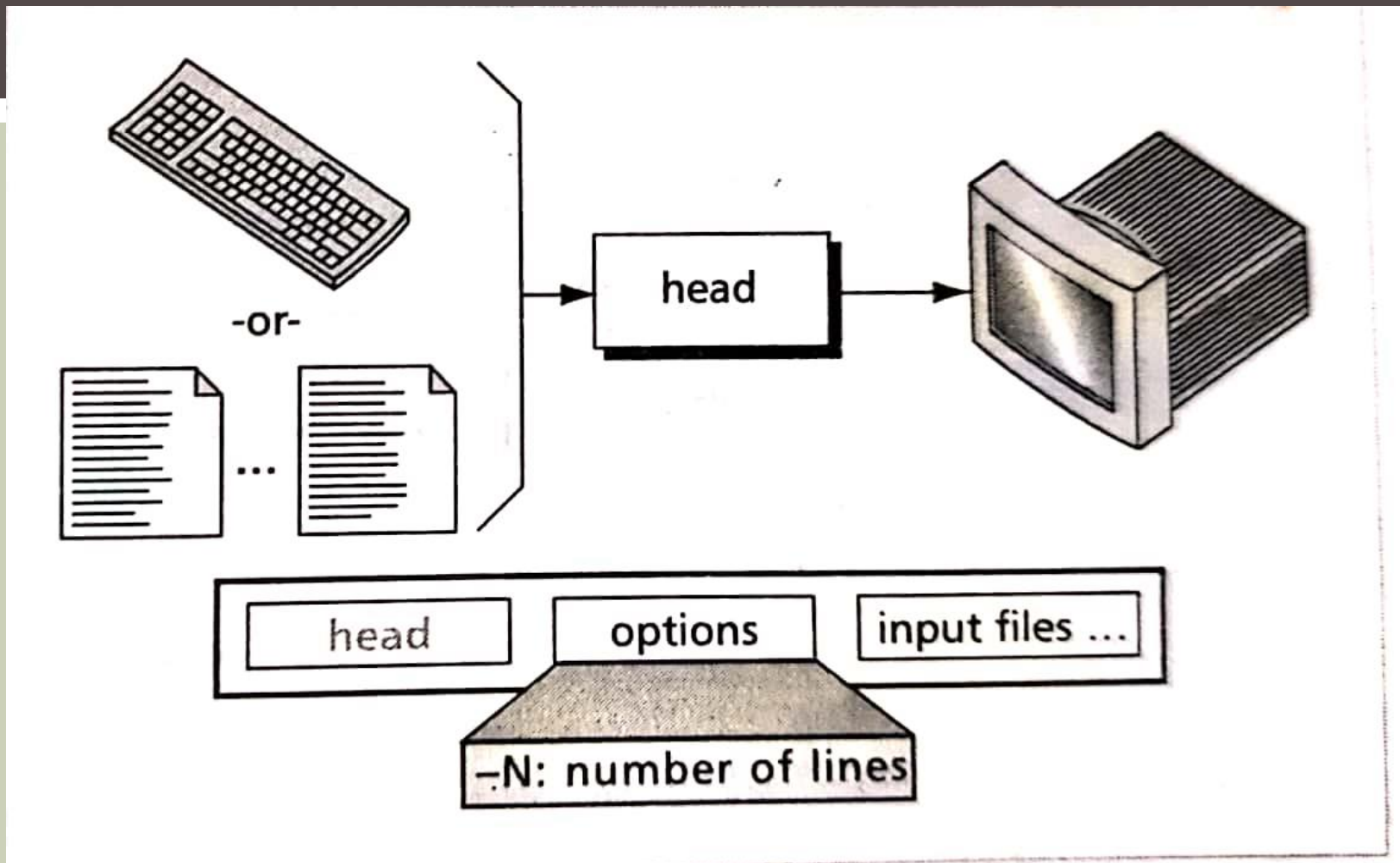


FIGURE 6.7 *The* **head** *Command*

- **Tail**
- The tail command also outputs data, only this time from the end of the file.
- The general format of the tail command is presented in figure.
- Its has many options to perform specific operation.

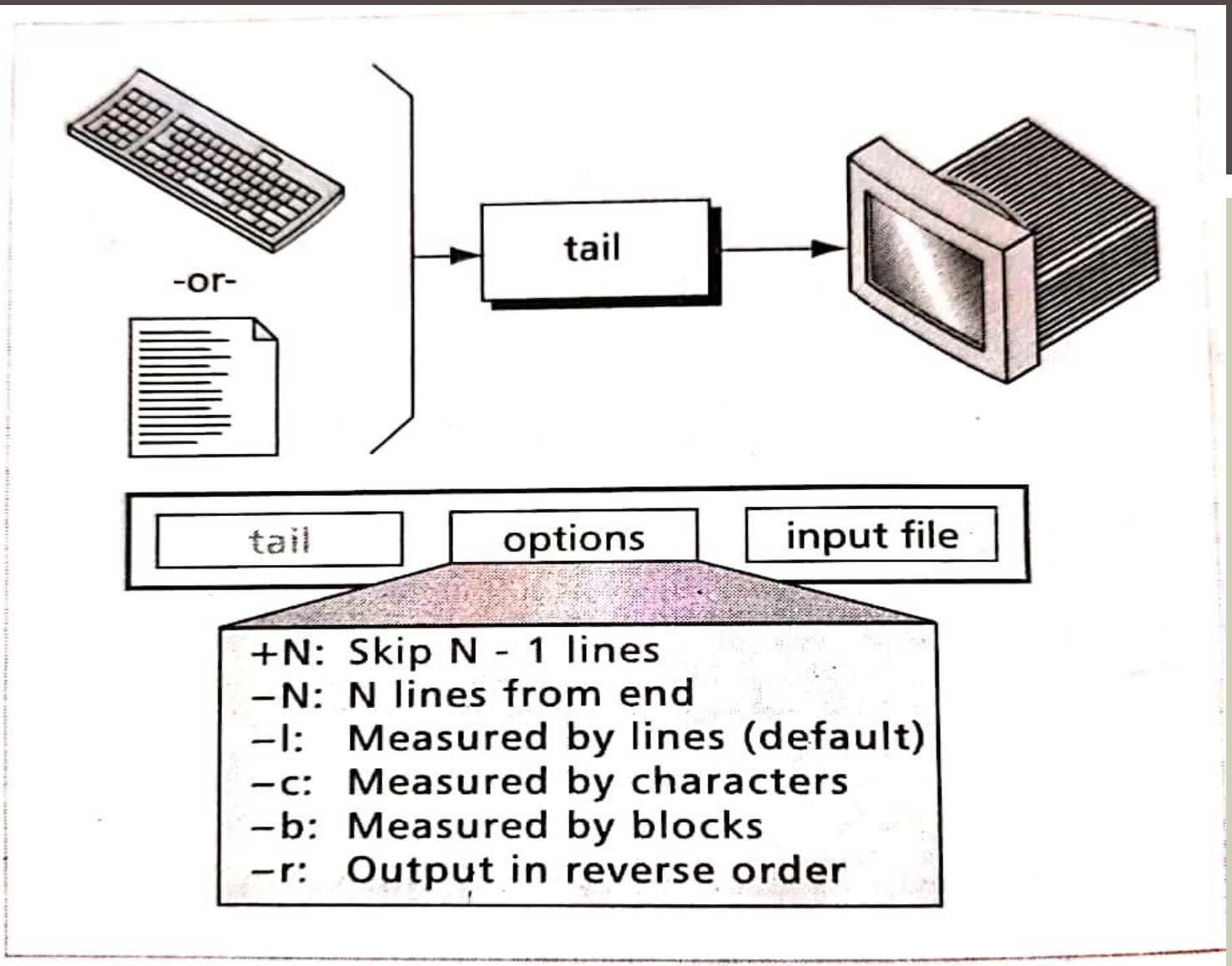


FIGURE 6.8 *The* **tail** *Command*

CUT AND PASTE

- The UNIX cut and paste commands performs operations on files. Cut removes columns of data from a file, and paste combines columns of data.
- Because the cut and paste commands work on columns, text files don't work well.

TABLE 6.3 *Data for the Five Largest Cities in the United States (1990 Census)*

Chicago	IL	2783726	3005072	1434029
Houston	TX	1630553	1595138	1049300
Los Angeles	CA	3485398	2968528	1791011
New York	NY	7322564	7071639	3314000
Philadelphia	PA	1585577	1688210	1736895

cut command

- The basic purpose of the cut command is to extract one or more columns of data from either standard input or from one or more files.

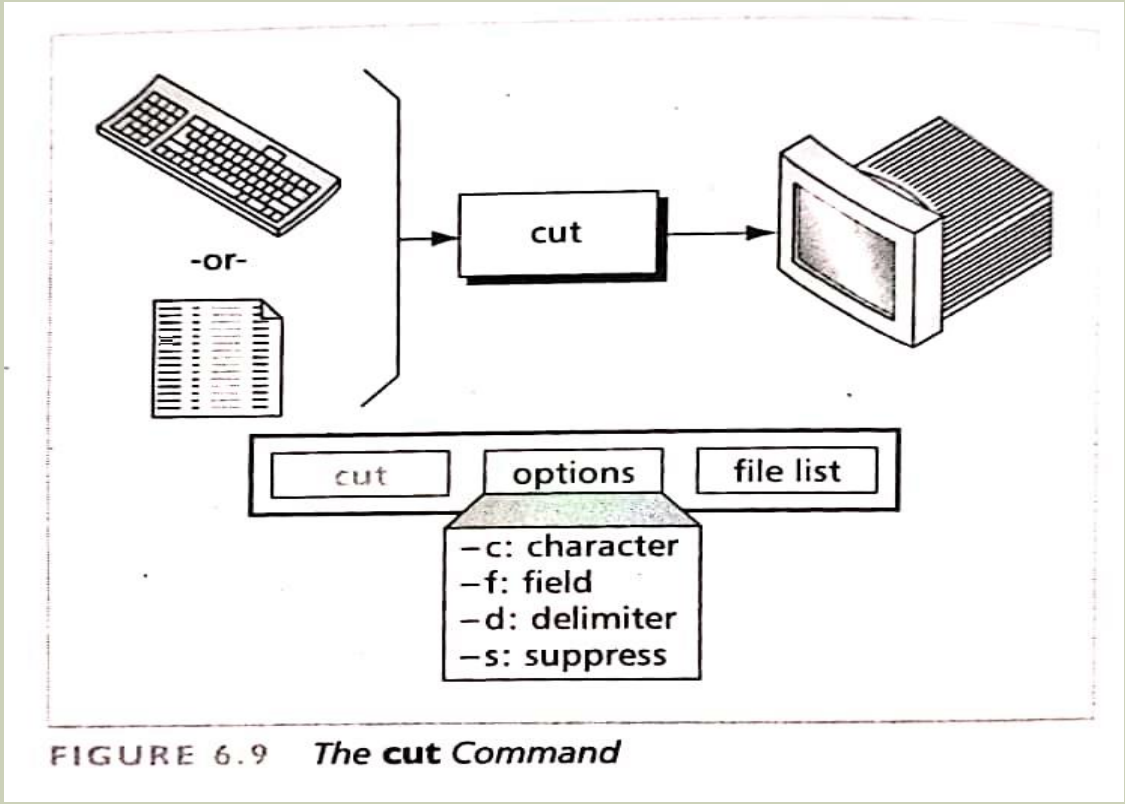


FIGURE 6.9 *The* **cut** *Command*

Specifying character positions

- Character positions work well when the data are aligned in fixed columns.
- To specify that the file formatted with fixed columns, we use the character option, -c, followed by one or more specifications.
- $ cut –c1-14,19-25 censusFixed

```
Chicago       2783726
Houston       1630553
Los angeles   3485398
New York      7322564
Philadelphia  1585577
```

- **Field Specification**
- While the columns specification works well when the data are organized around fixed columns, it doesn't work in other situations.
- We have indicated the locations of the tabs with the notation <tab> and have spaced the data to show how it would be displayed.

TABLE 6.4 *Data for Five Largest Cities in the United States (with Tabs)*

```
Chicago<tab>      IL<tab>  2783726<tab>  3005072<tab>  1434029
Houston<tab>      TX<tab>  1630553<tab>  1595138<tab>  1049300
Los Angeles<tab>  CA<tab>  3485398<tab>  2968528<tab>  1791011
New York<tab>     NY<tab>  7322564<tab>  7071639<tab>  3314000
Philadelphia<tab> PA<tab>  1585577<tab>  1688210<tab>  736895
```

- When the data are separated by tabs, it is easier to use fields to extract the data from the file.
- Fields are separated from each other by a terminating character known as a delimiter.
- To specify a field, we use the field option (-f).
- $ cut –f1 censusTab

SESSION 6.15 *Extract Field 1*

```
$ cut -f1 censusTab
Chicago
Houston
Los Angeles
New York
Philadelphia
```

- $ cut –f1,3-5 censusTab

- $ cut –f1,3,5 –d”/” censusSlash

Chicago/2783726/1434029

- There is one more option of interest, the suppress option (-s). This option tells cut not to display any line that does not have a delimiter.

TABLE 6.5 **cut** *Command Options*

Option	Code	Results
Character	`-c`	Extracts fixed columns specified by column number.
Field	`-f`	Extracts delimited columns.
Delimiter	`-d`	Specifies delimiter if not tab (default).
Suppress	`-s`	Suppresses output if no delimiter in line.

paste command

- The paste command combines line together. It gets its input from two or more files.
- The paste command combines the first line of the first file with the first line of the second file and writes the combined line to the standard output stream.
- Between the columns it writes tab and at the end of the column it writes a new line character.
- If the first file is longer than the second file, paste writes the extra data from first file a separation delimiter, such as tab and the new line until all data have been output.

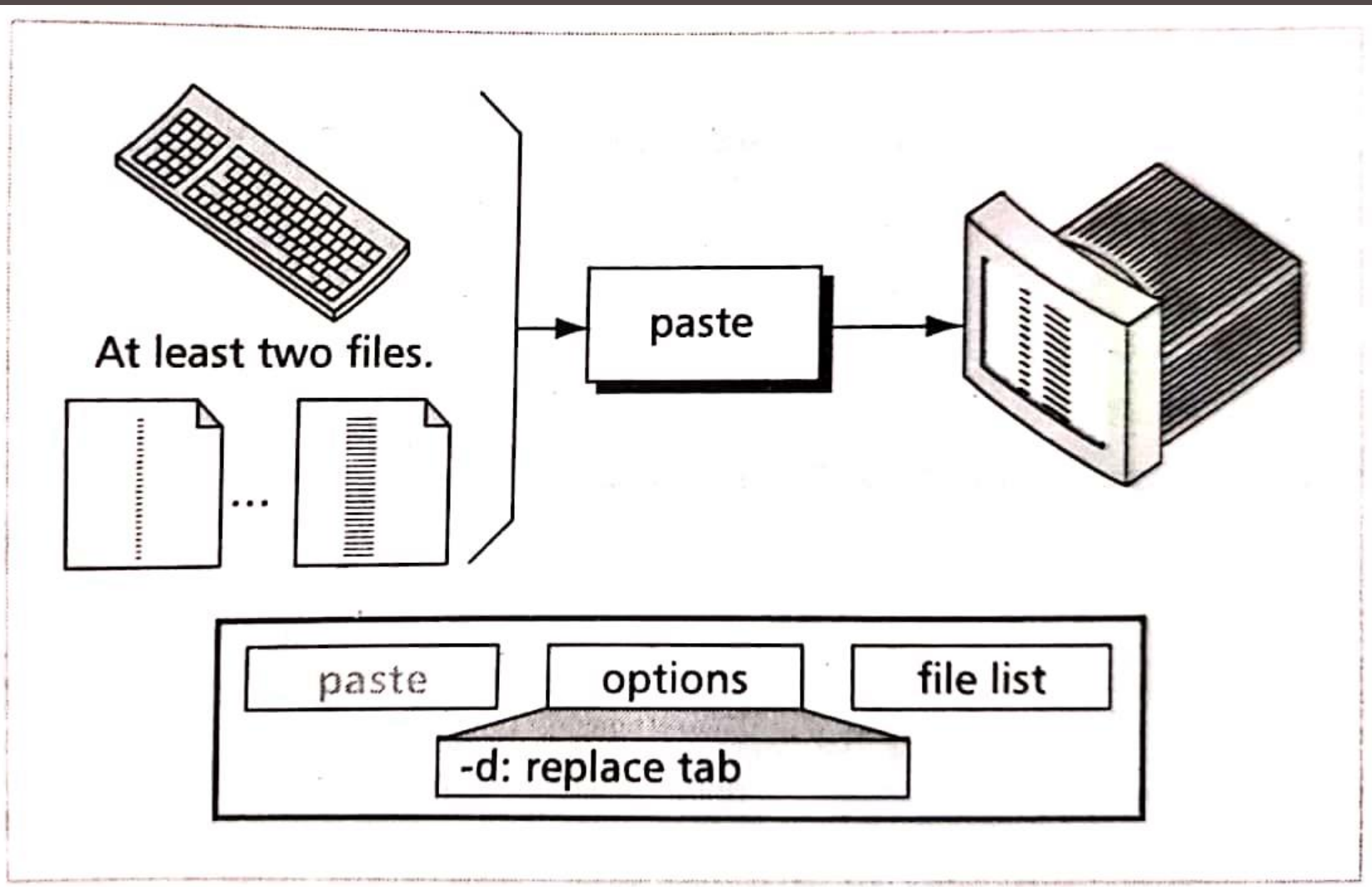


FIGURE 6.10 *The* **paste** *Command*

- You can specify one or more delimiter (-d) be used to separate the data.
- If there are two files, you can specify one delimiter, if there are three you can specify two delimiter.

- $ paste –d"\t#" fileOddEven fileOdd fileEven

```
1       1#2
2       3#4
3       5#6
4       7#
5       9#
```

SORTING

Sort command

- The sort utility options, field specifiers, and input files. The field specifiers tell it which fields to use for the sort.

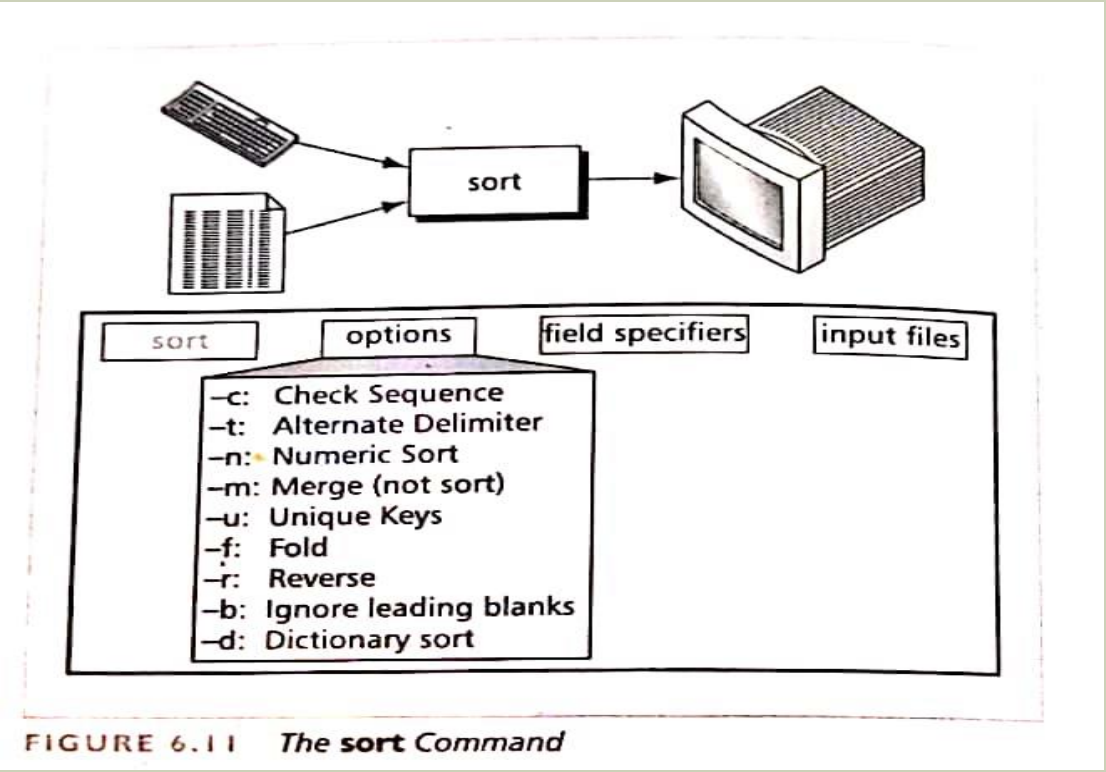


FIGURE 6.11 *The* **sort** *Command*

- **Sort by Lines**
- The easiest sort arranges data by lines. It compares the first character in one line with the first character in another line.
- In comparing characters, sort uses the ASCII value of each character.

- **Sort by Fields**
- Sorting by lines is easy but it works only when the data at the beginning of the line control the sort sequence.
- When the data that control the data sequence are not at the beginning of the line, we need to use a field sort.

TABLE 6.6 *Line Sort Demonstration*

Unsorted Data	Sorted Data
forouzan	gilberg
gilberg	!
8	27
27	8
!	Paula
~	Paulo
Paulo	forouzan
Paula	~

- **Fields**
- The UNIX sort defines a field as a set of character delimited by a single blank or a tab.
- The first field of a line is delimited by the space or tab after it.
- The second field is delimited by a space or tab before it and one after it.

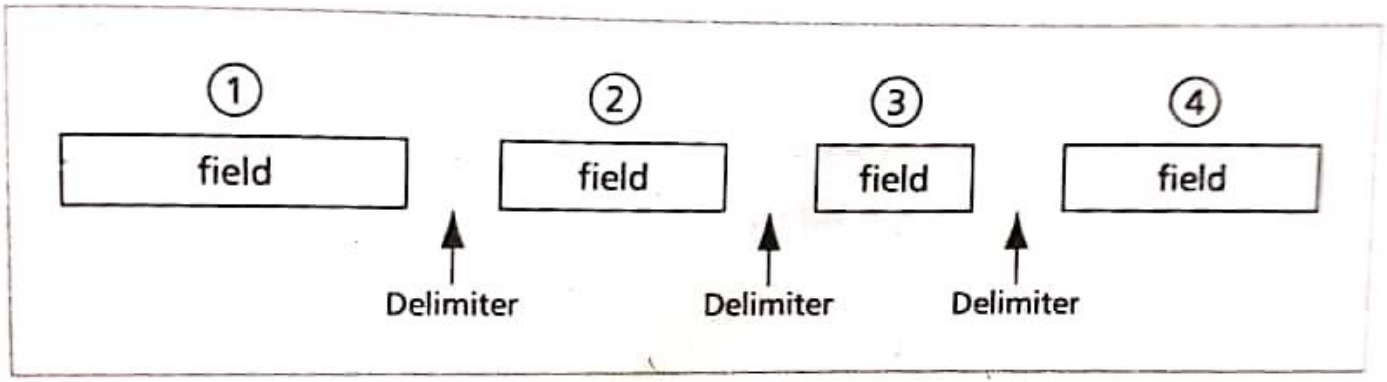


FIGURE 6.12 *Fields within a Line*

- **Field Specifiers**
- Field specifiers are a set of two numbers that together identify the first and last field in a sort key.
- Number1 specifies the number of fields to be skipped to get the beginning of the sort field, whereas number2 specifies the number of fields to be skipped relative to the beginning of the line to get to the end of the sort key (**+number1 –number2**).

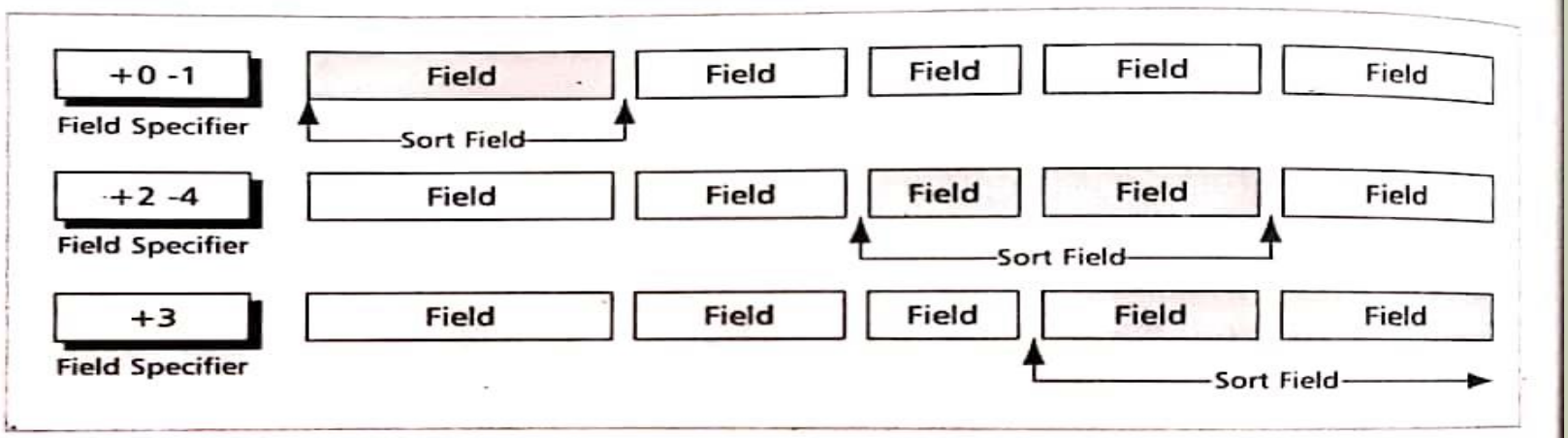


FIGURE 6.13 *Field Specifier Examples*

- If an invalid field specification is specified , such as start number greater than the no. of fields on a line, the field specification is ignored and sort uses the beginning of the line for the sort.

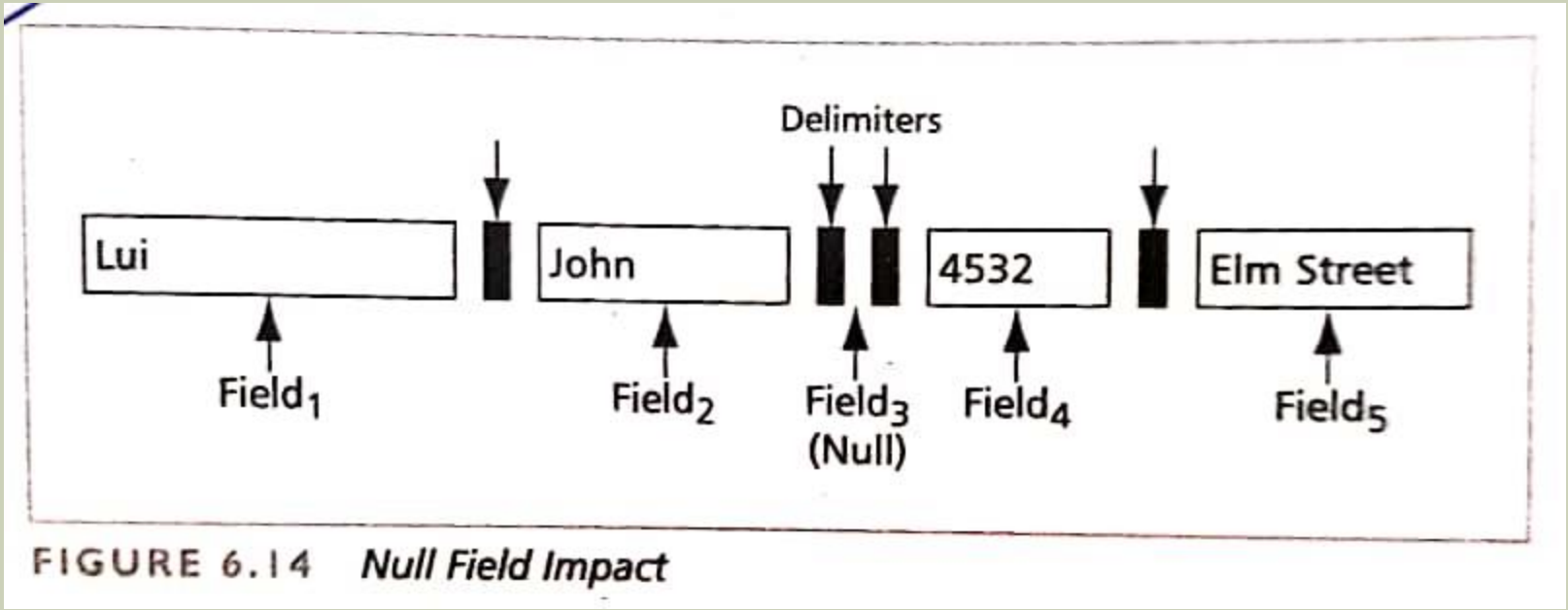


FIGURE 6.14 *Null Field Impact*

- **Options**
- Options can be used globally, in which case they apply to all field specifier, or locally, in which case they apply to only the current field specifier.

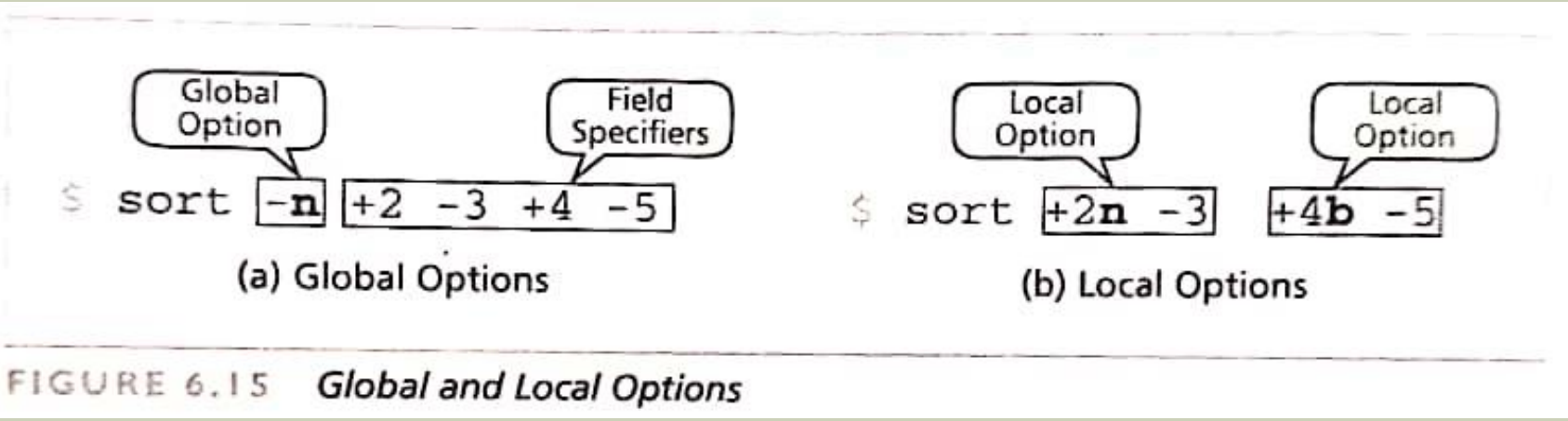


FIGURE 6.15 *Global and Local Options*

- **Check sort sequence**
- The check sort option (-c) verifies that the file is sorted. If it is not sorted, the first out-of-sequence line is displayed.

- **Delimiter**
- The delimiter option (-t) specifies an alternate delimiter. Like delimiter specifications in cut, it can and should be placed in quotes.

- **Numeric Sort**
- Sort assumes an ASCII value to the data to be sorted. It sorts data as though they are strings of characters.

Merge Files

- A merge combines multiple ordered files into one file that is ordered.
- The files are already ordered, you can save time by using the merge option (-m).
- That if the files are not ordered, sort will not give you an error message. Rather, it will do its best to sort the data, and the output will not be ordered.

Unique Sort Fields

- The unique option (-u) eliminates all but one line when the sort fields are identical.

- **Reverse (Descending) order**
- To order the data from largest to smallest, we specify reverse order (-r).

- **Ignore Leading Blanks**
- The blanks option (-b) is mandatory in fixed formatted fields. If we do not ignore leading blanks, then each blank is considered a separate [null] field.

- **Dictionary Sorting**
- To sort the index correctly, we use the dictionary option(-d).

- **Fold Lowercase**
- To make our sort ignore the difference between upper and lower case, we use the fold option (-f). In the fold option, the uppercase character are folded into lowercase so that they all sort the same.

TABLE 6.7 *Sort Command Options*

Option	Code	Results
Check sequence	`-c`	Verifies that data are correctly sorted.
Alternate delimiter	`-t`	Specifies alternate delimiter. Default: tab and space.
Numeric sort	`-n`	Data are numeric, not string. Default: string data.
Merge	`-m`	Input files are sorted. Merges them.
Unique sort fields	`-u`	Deletes duplicate sort fields, keeping last one.
Reverse order	`-r`	Sorts data in descending sequence.
Ignore leading blanks	`-b`	Considers leading blanks as one blank for field separation.
Dictionary sort	`-d`	Sorts special characters first.
Fold lowercase	`-f`	Sorts upper- and lowercase together.

TRANSLATING CHARACTERS

- **tr Command**
- The tr command replaces each character in a user-specified set of characters with a corresponding character in a second specified set.

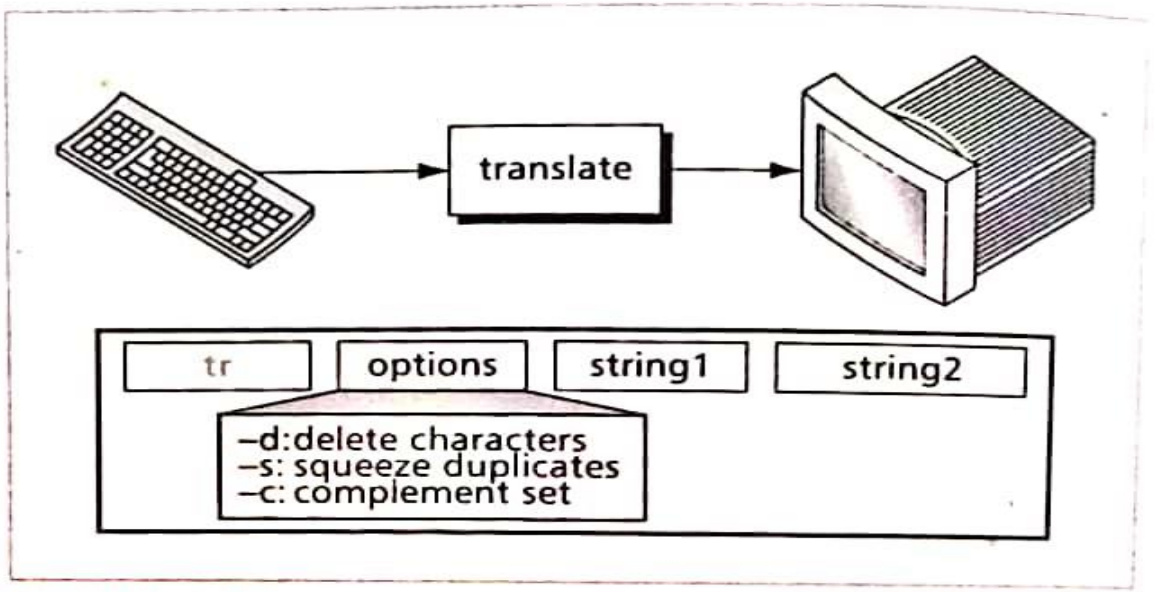


FIGURE 6.17 *The* **tr** *Command*

SESSION 6.44 *Squeeze Output with Translate*

```
$ tr -s "ie" "dd"
The fiend did dastardly deeds
Thd fdnd d dastardly ds
```

SESSION 6.45 *Using Translate's Complement Option*

```
$ tr -c "aeiou" "*"
It is very easy to use TRANSLATE.
**i***e***ea****o*u*e***********
```

TABLE 6.8 *Translate Options*

Option	Code	Results
Delete characters	-d	Deletes matching characters.
Squeeze duplicates	-s	After translation, deletes consecutive duplicate characters.
Complement set	-c	Uses complement of matching set.

- **Simple Translate**
- Translate receives its input from standard input and writes its output to standard output.
- **$ tr “aeiou” “AEIOU” –>This is classroom**
- o/p :- ThIs Is classrOOm

- **Nonmatching Translate Strings**

SESSION 6.42 *Translate Strings Don't Match*

```
$ tr "aeiou" "AE?"                    # Case 1: string2 is shorter than string1
It is very easy to use translate.
It ?s vEry EAsy t? ?sE trAnslAtE.

$ tr "aei" "AEIOU"                    # Case 2: string1 is shorter than string2
It is very easy to use translate.
It Is vEry EAsy to usE trAnslAtE.
```

FILES WITH DUPLICATE LINES

- **uniq command**
- The uniq command deletes duplicate lines, keeping the first and deleting the others.
- To delete the nonadjacent lines, the file must be sorted.

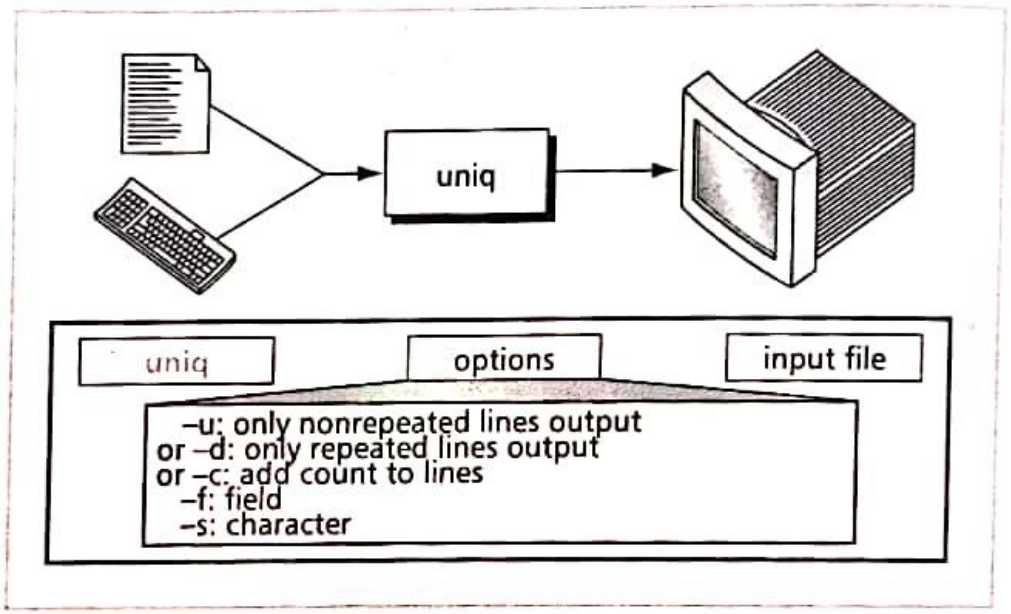


FIGURE 6.18 *The* **uniq** *Command*

TABLE 6.10 *Unique Options*

Option[a]	Code	Results
Unique	`-u`	Only unique lines are output.
Duplicate	`-d`	Only duplicate lines are output.
Count	`-c`	Outputs all lines with duplicate count.
Skip field	`-f`	Skips leading fields before duplicate test.
Skip characters	`-s`	Skips leading characters before duplicate test.

[a]Unique, duplicate, and count cannot be combined.

COUNT CHARACTERS, WORDS, OR LINES

- **wc command**
- The wc command counts the number of characters, words, and lines in one or more documents.
- The character count includes newlines (/n).
- Options can be used to limit the output to only one or two of the counts.

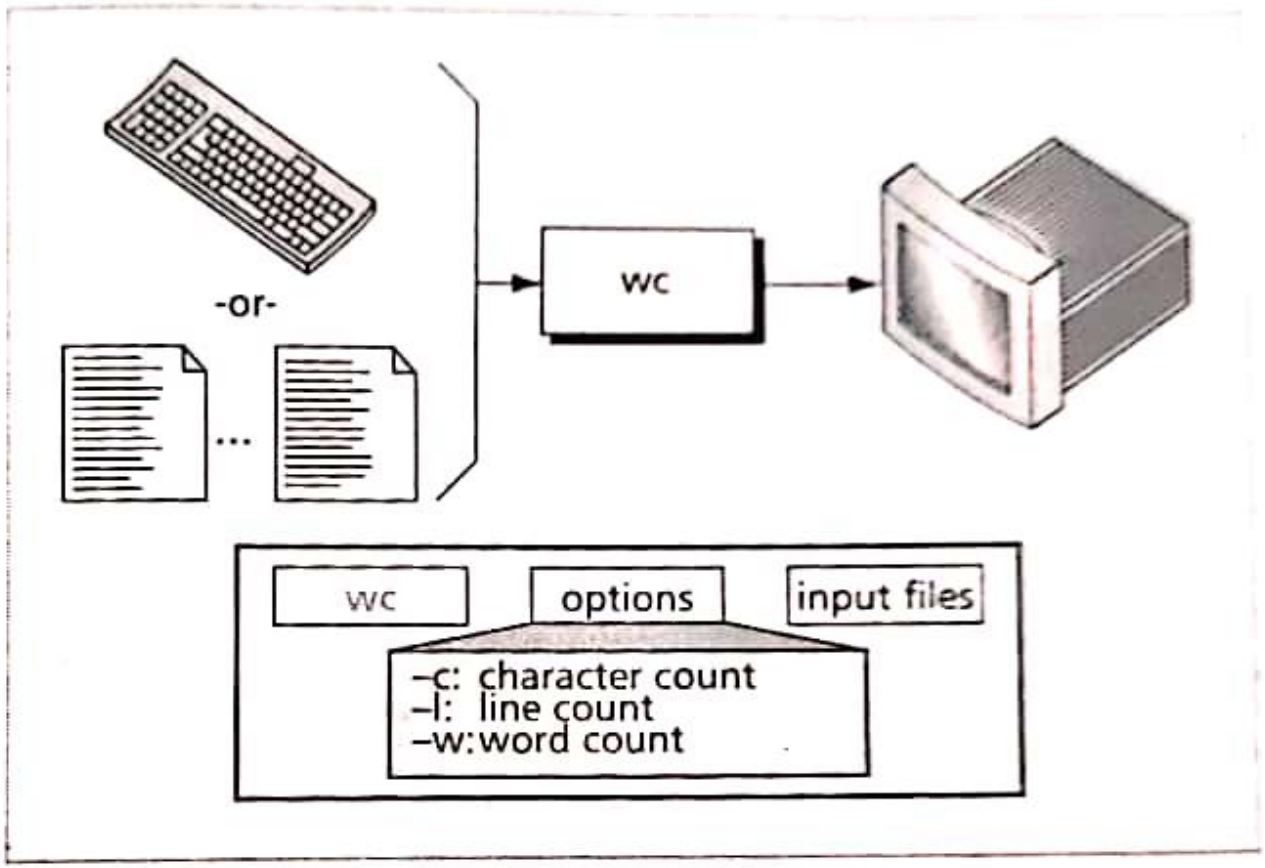


FIGURE 6.19 *The* **wc** *Command*

TABLE 6.11 *Word Count Options*

Option	Code	Results
Character count	`-c`	Counts characters in each file.
Line count	`-l`	Counts number of lines in each file.
Word count	`-w`	Counts number of words in each file.