UNIT-4

Filters

Contents

Filters

- 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.

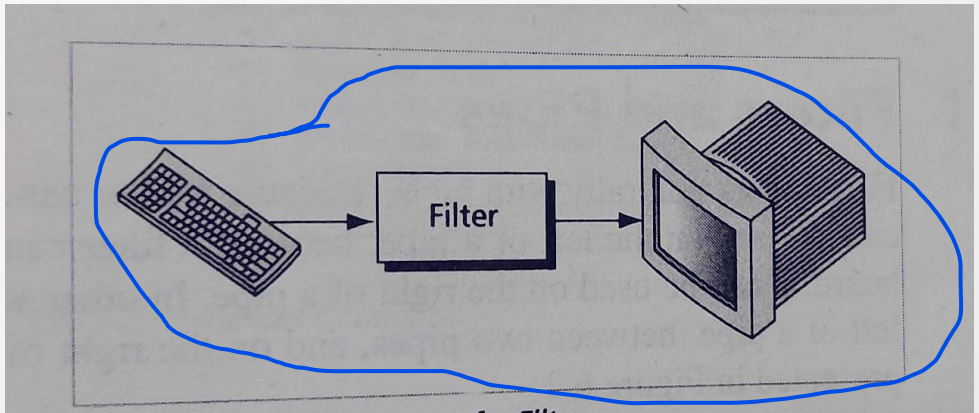


FIGURE 6.1 *Concept of a Filter*

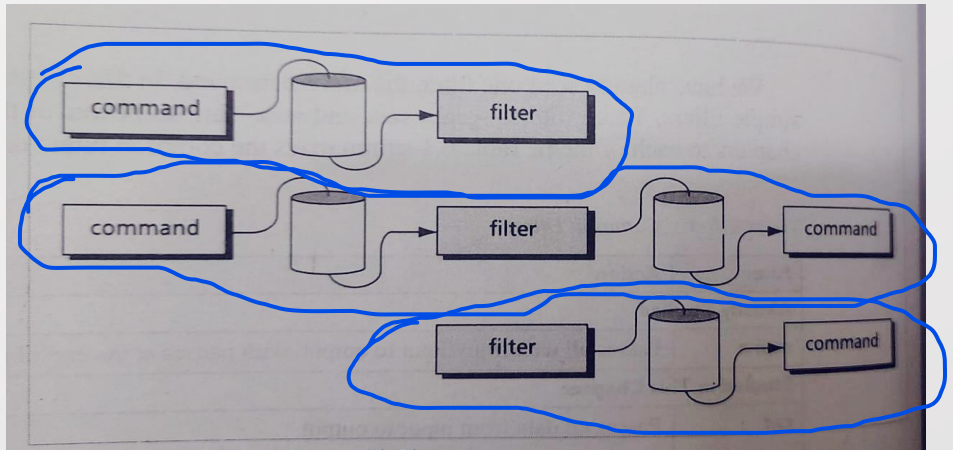


FIGURE 6.2 *Using Filters with Pipes*

Common Filters:

cat- passes all data from input to output.

cmp- Compares two files.

comm-Identifies common lines in two files.

diff- Identifies differences between two files.

head- Passes number of specified lines at the beginning of the data

tail- Passes number of specified lines at the end of the data

Common Filters:

- **cut**- passes only specified columns.
- **paste-** Combines columns
- **sort-** Arranges the data in sequence
- **tr-** translates one or more characters as specified.
- **Uniq** – deletes duplicate lines.
- **Wc** – counts characters, words, or lines.

Three special filters:

- **grep** – passes only specified lines.
- **sed** – passes edited lines.
- **awk** – passes edited lines and parses lines.

1.Filters and Pipes.

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

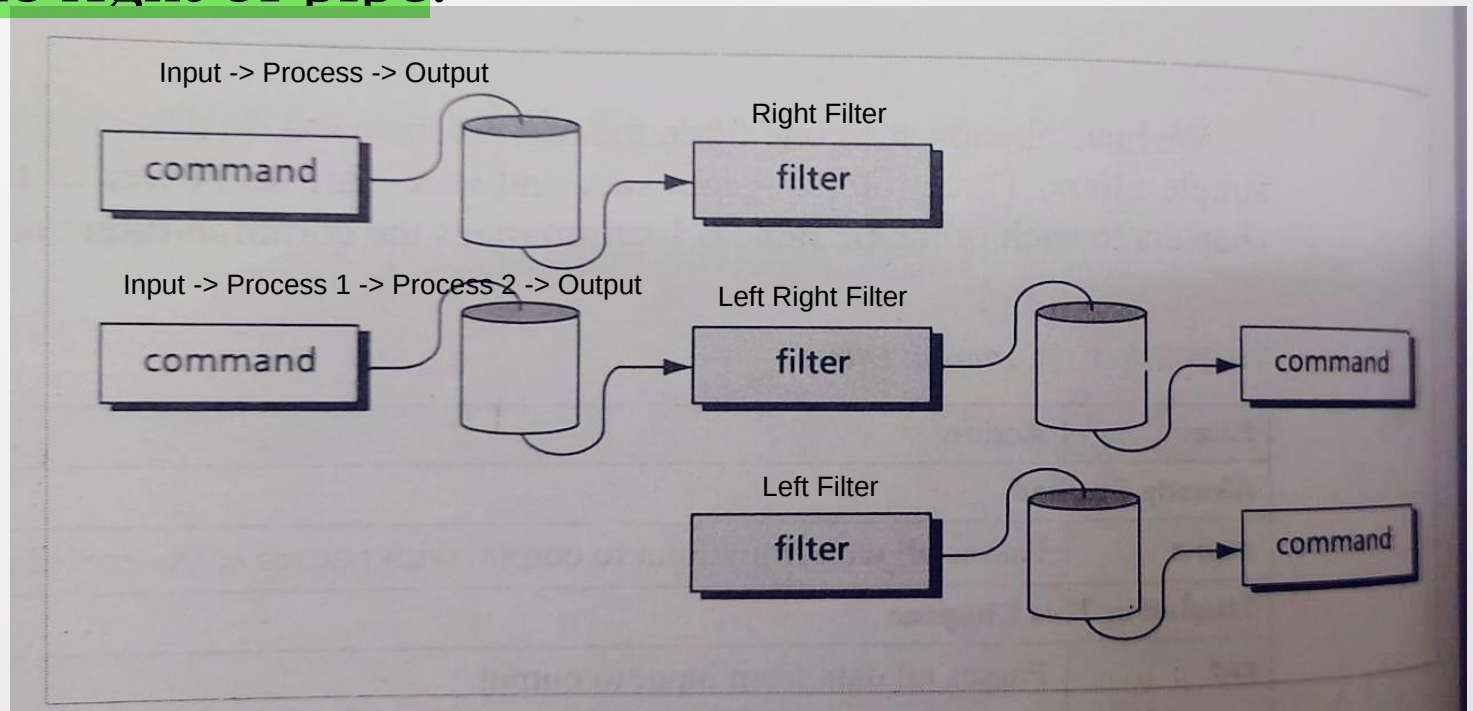


FIGURE 6.2 *Using Filters with Pipes*

2. Concatenating files (cat)

- Unix Provides a powerful utility to concatenate commands.
- It is known as the **catenate** command, or **cat** for short.
- It combines one or more files by appending them in the order they are listed in the command.
- The input can come from the keyboard; the output goes to the monitor.

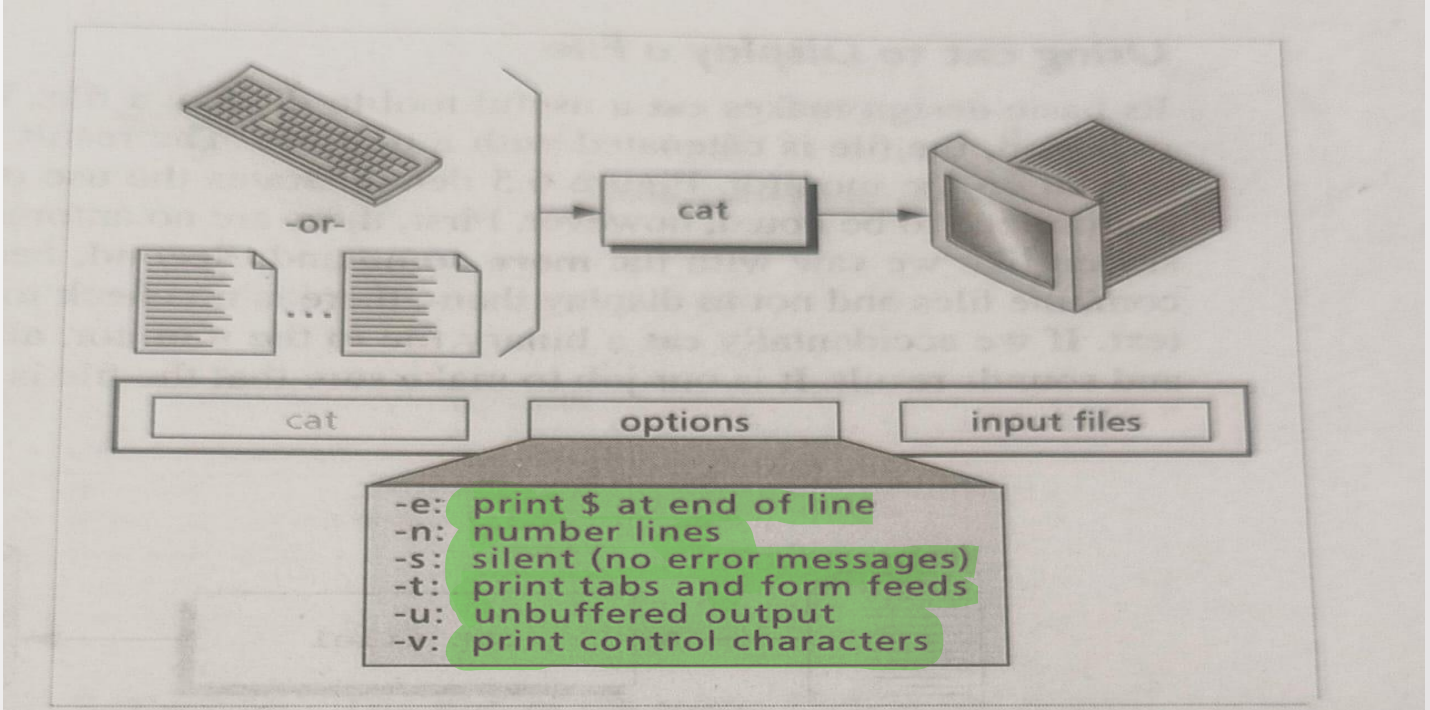


FIGURE 6.3 *The **cat** Command*

cat vtunes

2. Concatenating files (cat)

- Concatenate files:
  cat f1,f2
- Using cat to display a file:
  cat filename
- Using cat to create a file:
  cat > filename
- End of file ctrl+d (abbreviated as ^d)
- cat options: There are 6 options, grouped in to 4 categories.

➢ Visual characters
➢ Buffered output
➢ Missing files
➢ Numbered lines

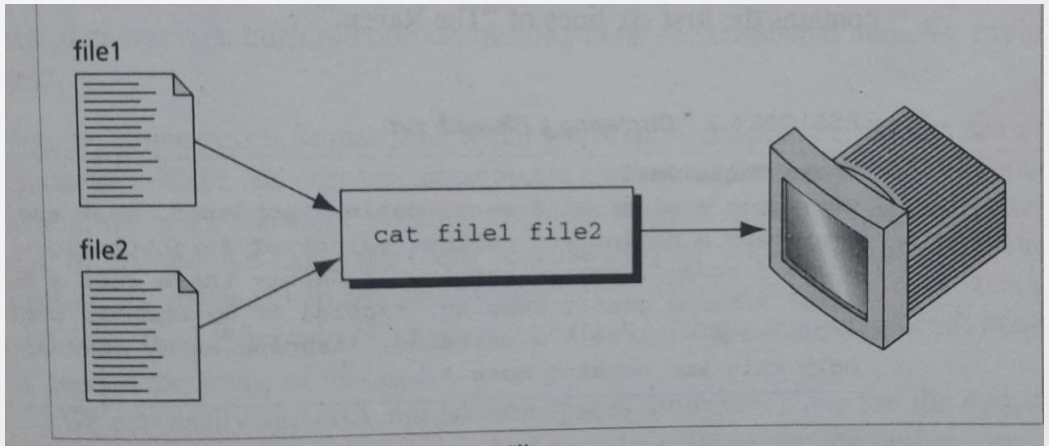


FIGURE 6.4 *Using* **cat** *to Combine Files*

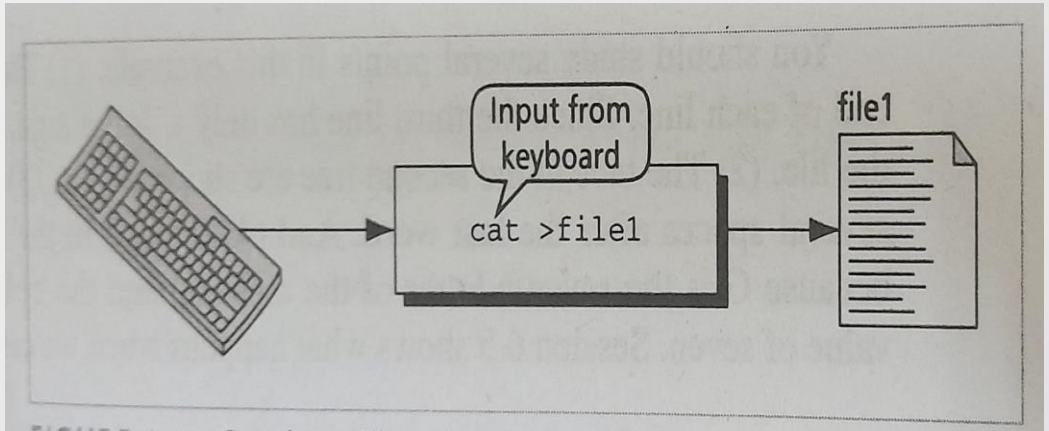


FIGURE 6.6 *Creating a File with* **cat**

Concatenating files (cat)

- Visual characters:

cat –v ------print control characters

cat –vet filename ----- (-ve) $ sign printed at the end, (-vt) tabs appear as ^I.

- Buffered output

cat –u ----- written to the file immediately

- Missing files

cat –s-------To avoid error message on output screen when file is missing during concatenation make silent option is used.

- Numbered lines

- cat –n ------numbered line output

Display beginning and end of files:

head command

- head command specified number of lines from the beginning of one or more files to the standard output.
- head –n------outputs first N lines (default is 10 lines)
- When multiple files are included

head –n file1 file2 ------it displays filename before its output.

tail command

Displays the data from the end of the line

Tail command

- +N --- Skips N-1 lines, copies rest to end of file.
- -N ----- Last N lines
- -l ------- counts by line
- -c ------- counts by character
- -b ------- counts by disk block
- -r ------- outputs in reverse order

- If we want to extract lines from 8 to 13
- head -13 filename | tail +8

head -13 takes the first 13 lines, tail +8 skips the first 7 lines and prints the rest, so 8-13

Cut and paste

- Cut and paste commands perform similar operations on files.
- Cut removes columns of data from a file
- Paste combines columns of data.

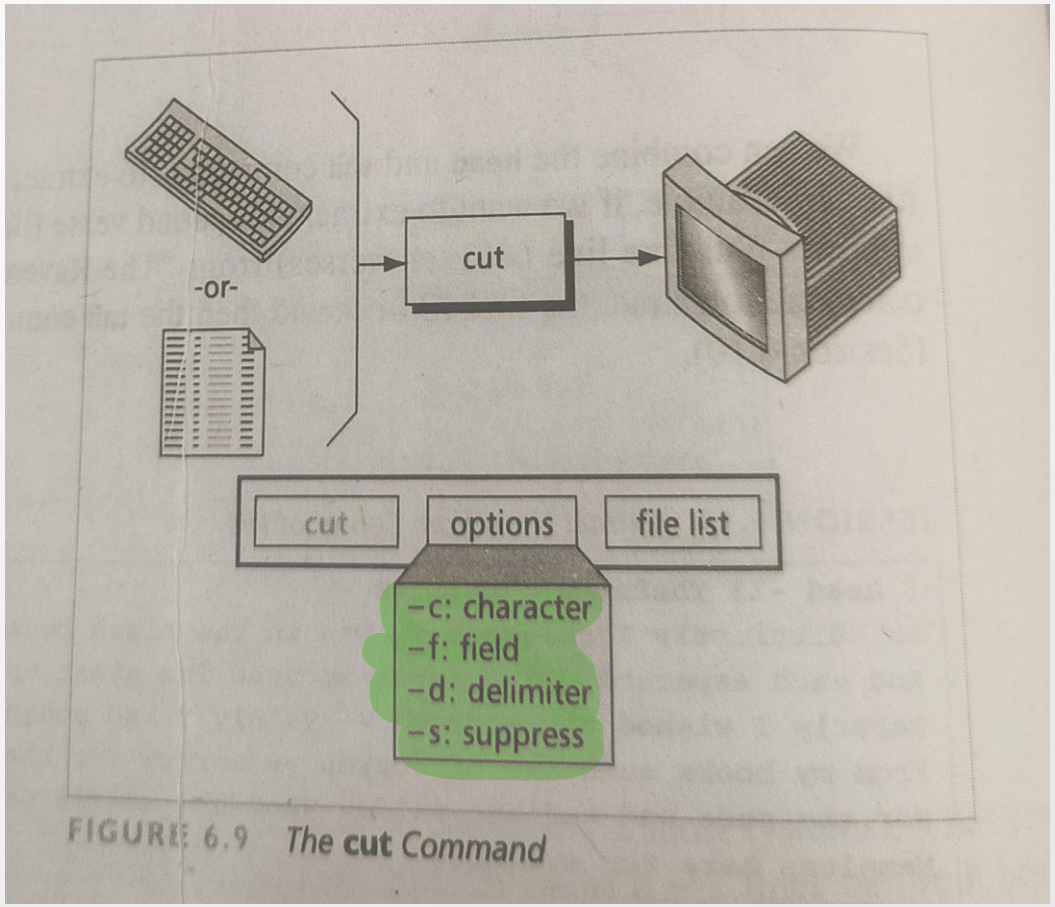


FIGURE 6.9 The **cut** Command

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

Cut

- Specifying character positions
  - –c----character option

    cut –c1-14, 19-25 filename

    this command will output the characters in positions 1-14 and 19-25 from each line of that file

- Field specification
  - f ------fields

    cut –f1,3-5 filename

    command will output fields 1, 3, 4, and 5 from each line of the file named filename

  - -d------specifies delimiter if no tab

    cut –f1,3-5 –d”/” filename

    command will output fields 1, 3, 4, and 5 from each line of the file named filename. Fields are determined by the delimiter specified by -d, which is /

  - -s ------not to display any line that does not have delimiter (suppress the output if no delimiter in line)

- Paste – combines two files

  paste f1 f2

  doesnt modify original files, but just displays in the terminal

- Paste option –d (specifies delimiter between file contents)

  paste –d”\t#” file1 file2 file3

Paste

It combines lines together.

Cat and paste are similar ---but cat combines files vertically (by lines). The paste combines files horizontally (by columns)

- Paste – combines two files

  paste f1 f2 Tab delimiter (default)

- Paste option –d (specifies delimiter between file contents)

  paste –d”\t#” file1 file2 file3

apple\tfruit#red
banana\tfruit#yellow
cherry\tfruit#red

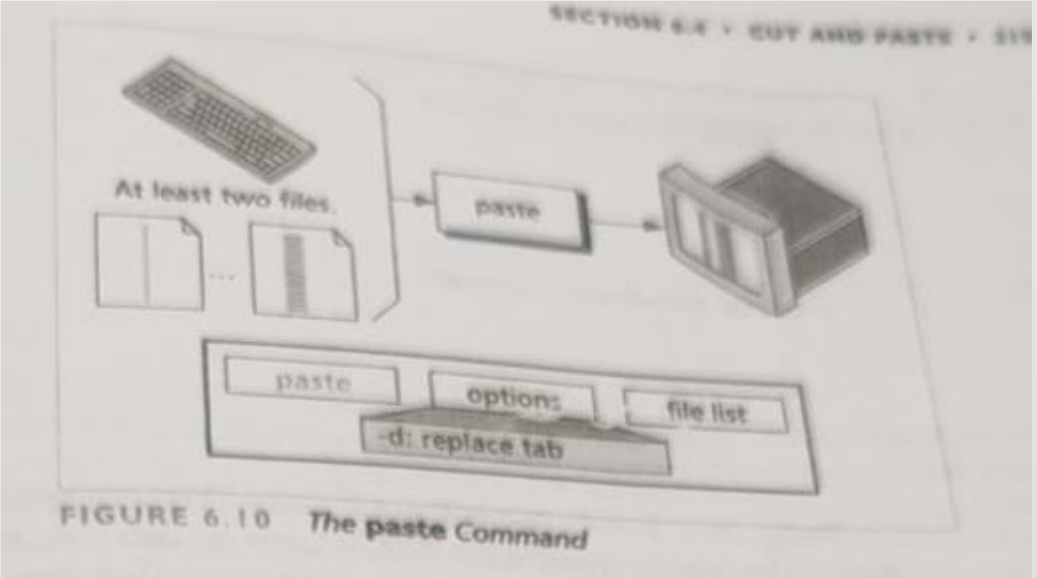


FIGURE 6.10 *The* **paste** *Command*

Sorting

- Lot of data—We need to organize them for analysis and efficient processing.
- Most powerful organizing techniques is sorting.
- We arrange them in sequences---sort
- We use ascending sequence—arrangement in which each piece of data is larger than its precedings
- Sort in descending sequence

sort mn u r fd cnt

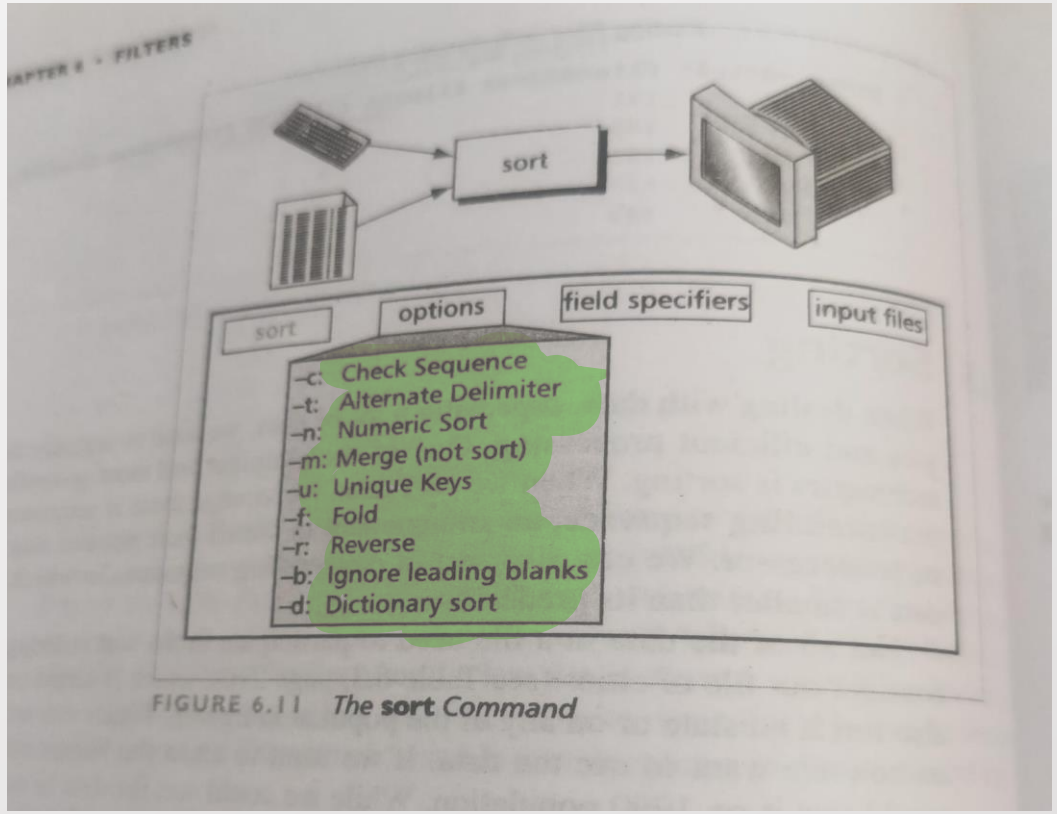


FIGURE 6.11 *The* **sort** *Command*

Sorting

- It is a simple and most powerful organizing technique.
- Sort by lines: Arrange data by lines.

  Uses ASCII value of each character (A—65, a – 96)
- Sort by fields: Sort defines a field as a set of characters delimited by a single blank or tab

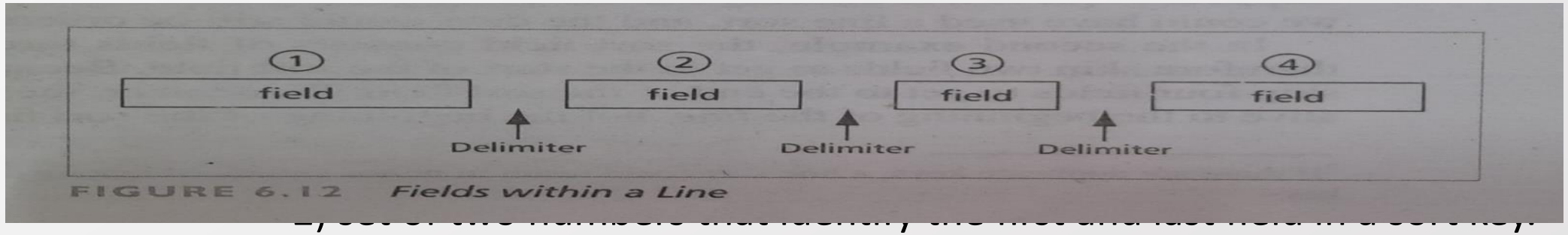
- 

FIGURE 6.12 *Fields within a Line*

+number1 -number2 (number1--- specifies the no of fields to be skipped to the beginning of the sort field.

number2---specifies the no of fields to be skipped, relative to the beginning of the line, to get to the end of the sort key.)

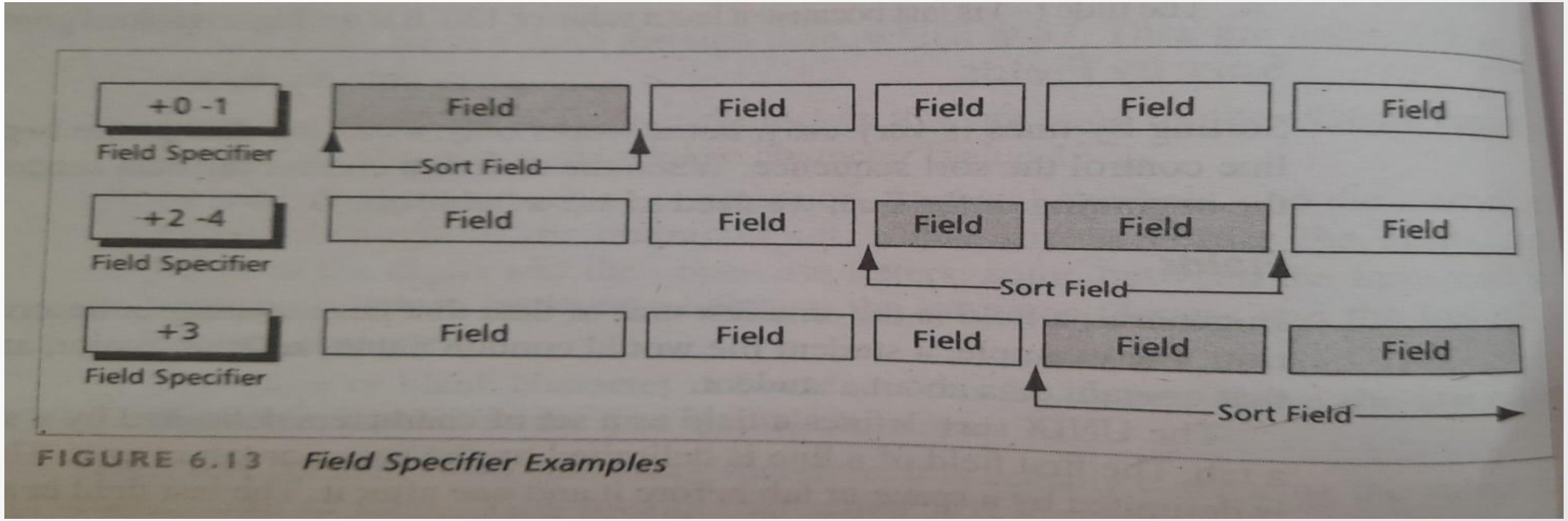


FIGURE 6.13 *Field Specifier Examples*

- $ sort +0 -1 filename
- $ sort +2 -4 filename
- Check sort sequence (-c): verifies that the file is sorted or not, if not sorted, the first out-of-sequence line is displayed.

  $ sort –c +0 -1 filename

Field 0 to 1

Field 2 to Field 4

sort +0, -1 filename

John Math 85	Amy Math 95
Amy Science 90	Amy Science 90
Mark History 78	John Math 85
Amy Math 95	John Science 80
John Science 80	Mark History 78
Mark Math 88	Mark Math 88

Delimiter (-t):

- Specifies alternate delimiter
- The -t Option As mentioned, if you skip over fields, sort assumes that the fields being skipped are delimited by space or tab characters.
- The -t option says otherwise. In this case, the character that follows the -t is taken as the delimiter character.
- $ sort –t'&' +1 -2 /etc/passwd ------Sort by user id

Root &3 &1 &The Super User
Cron &1 &2 &Cron Daemon for periodic tasks
Bin &0 &3 &The owner of system files

Bin &0 &3 &The owner of system files
Cron &1 &2 &Cron Daemon for periodic tasks
Root &3 &1 &The Super User

- Numeric sort (-n):

- $ sort -n data ----------Sort numerically

- Skipping Field: The +1 says to skip the first field. Similarly, +5n would mean to skip the first five fields on each line and then sort the data numerically.

- Fields are delimited by space or tab characters by default.

- If a different delimiter is to be used, the -t option must be used.

- $ sort +1 -2n data ----------Skip the first field in the sort

John 30
Alice 20
Bob 25

Alice 20
Bob 25
John 30

- Merge files (-m): Combines multiple ordered files into one file that is ordered.
- $ sort –m file1 file2 merge sort already sorted files.
- Unique sort fields (-u): $ sort -u names ------The -u option tells sort to eliminate duplicate lines from the output. unique lines are only kept

$ sort –t'/' –u +1 -2 filename

$ sort –u –t'/' +1 -2 filename

$ sort –ut'/' +1 -2 filename

- Ignore Leading blanks (-b):

➢ If we do not ignore leading blanks then each blank is considered as separate null field.

➢ If this option is used fields can have no embedded spaces.

$sort –b +1 -2 filename

Reverse (-r): To order data from largest to smallest

- $ sort names

Charlie
Emanuel
Fred
Lucy
Ralph
Tony
Tony

$ sort -nr +2 -3 filename

- $ sort -r names --------Reverse sort

Tony
Tony
Ralph
Lucy
Fred
Emanuel
Charlie

Multiple pass sort: $ sort –t'/' +1 -2 +2n -3 filename
$ sort –t'/' +1 -2 +2nr -3 filename

Translating characters: tr command

- The tr filter is used to translate characters from standard input. The general form of the command is

tr from-chars to-chars

Ex: $ tr e x < file.txt e replaced with x

$ tr '[a-z]' '[A-Z]‘< file.txt

lower to uppercase

tr dsc

- $tr “aeiou” “AEIOU”
- It is easy to use translate
- It Is EAsy tO UsE trAnslAtE

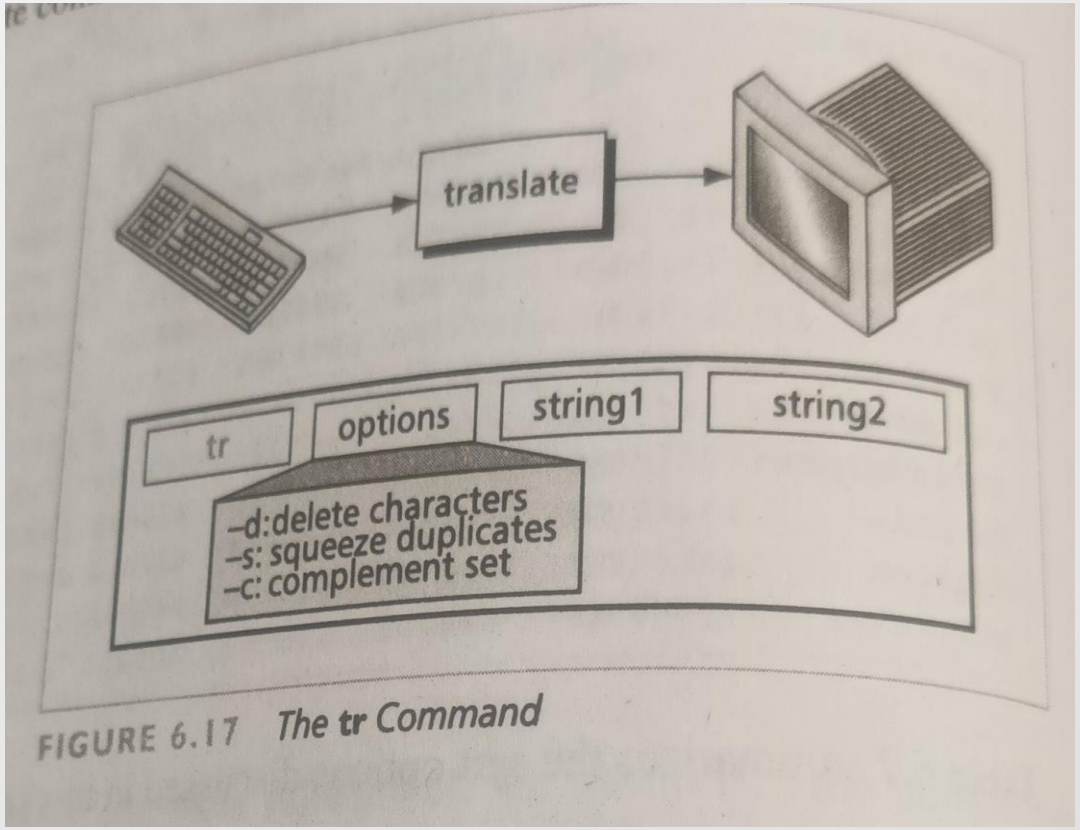


FIGURE 6.17 The tr Command

Translating characters: tr command

- -d -------delete character
- $tr –d “aeiou”
- It is easy
- It s sy

- tr –s “ie” “dd”----------squeeze output (deletes consecutive occurrences of same character)

  The fiend did dastardly deeds

  Thd fdnd d dastardly ds

The "ie" and "dd" arguments tell tr to replace every occurrence of i with d and every occurrence of e with d

1st. Thd fddnd ddd dastardly ddds

2nd. Thd fdnd d dastardly ds

- tr -d '[0-9]' ----------Delete all digits

Files with duplicate lines: Uniq command

- Uniq f1
- Uniq –u f1----------displays with the message
- Uniq –d f1----------only duplicate lines
- Uniq –c f1----------count duplicate lines

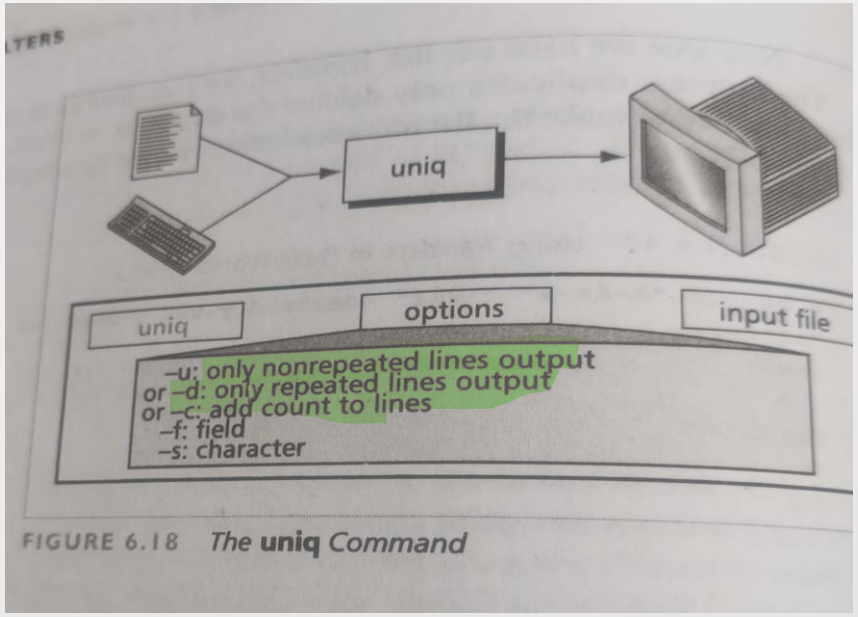


FIGURE 6.18 *The* **uniq** *Command*

- $ sort names | uniq
- $ sort names | uniq -d ----------List duplicate lines
- $ sort names | uniq –c -------Count line occurrences

uniq fu dsc

1 Charlie

1 Emanuel

2 Tony

Count Characters, words, or lines: wc command

- wc –c filename ------counts the number of characters
- wc –w filename ----- counts the number of words
- wc –l filename ------- counts the number of lines

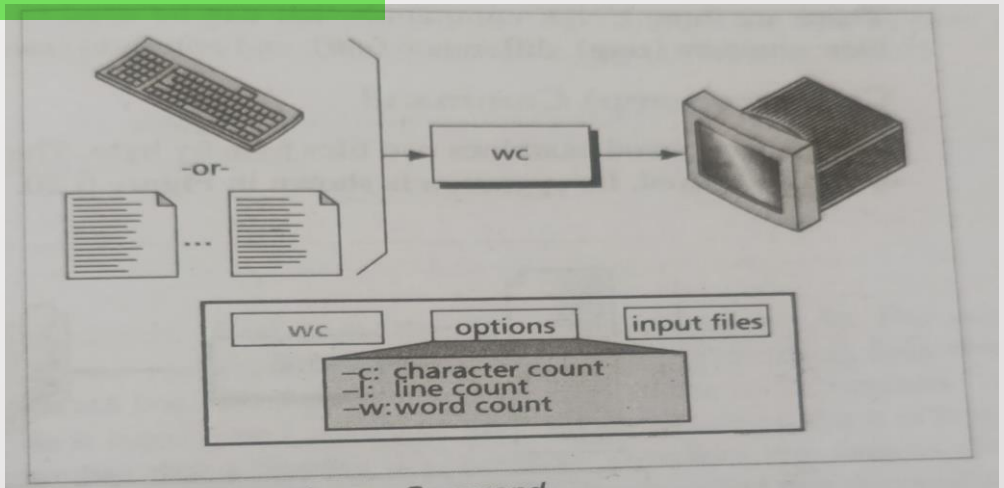


FIGURE 6.19 *The* **wc** *Command*

Comparing files:

- Compare (cmp) command

  cmp f1 f2----------compares two files byte by byte

- Difference (diff) command

  diff f1 f2

- Common (comm) command

  comm f1 f2

Compare (cmp) command

- cmp (compare) command needs two filenames as arguments.
- The two files are compared byte by byte,and the location of the first mismatch is echoed to the screen.
- If two files are identical, cmp displays no message, but simply returns the prompt.

```
user@ubuntu:~$ cat abc
This is some text
user@ubuntu:~$ cat xyz
This is another file
user@ubuntu:~$ cmp abc xyz
abc xyz differ: byte 9, line 1
user@ubuntu:~$
```

Common (comm) command

- It requires two sorted files, and lists the differing entries in different columns.
- When you run comm, it displays a three-columnar output.
- The **first** column contains lines unique to the first file, and the **second** column shows lines unique to the second file.The **third** column displays lines common to both files.

```
user@ubuntu:~$ cat file1
Ankur
Charul
Ishaan
user@ubuntu:~$ cat file2
Anubhav
Charul
Himanshu
Vishal
user@ubuntu:~$ comm file1 file2
Ankur
        Anubhav
                Charul
        Himanshu
Ishaan
        Vishal
user@ubuntu:~$
```

Common (comm) command

- These commands require single-column output from `comm`, and `comm` can produce it using the options -1, -2 or -3
- To drop a particular column, simply use its column number as an option prefix.

```
user@ubuntu:~$ comm -12 file1 file2
Charul
user@ubuntu:~$
```

Difference (diff) command

- Unlike its fellow members, cmp and comm, it also tells you which lines in one file have to be changed to make the two files identical.

```
user@ubuntu:~$ cat file1
Ankur
Charul
Ishaan
user@ubuntu:~$ cat file2
Anubhav
Charul
Himanshu
Vishal
user@ubuntu:~$ diff file1 file2
1c1
< Ankur
---
> Anubhav
3c3,4
< Ishaan
---
> Himanshu
> Vishal
user@ubuntu:~$
```