PIPES

- Used to combine two or more commands
- The commands are separated by a vertical bar (|)
- The standard output of one command is sent to the standard input of another command
- No intermediate outputs can be viewed
- If error occurs in the 1st command, then the error is sent to the error stream instead of sending it to the next command

Pipes

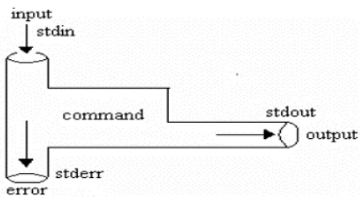
syntax:

```
command1 | command2
```

example:

```
$ ls -l | wc -l
```

The output of ls -l is sent as input to the wc command and the number of lines is displayed in the output stream



tee

- In pipe the intermediate output can't be viewed
- tee is a command which saves the output in a file as well as writes the output to the standard output

example:

$ who | tee userlist.txt
The tee command displays the output of who in the monitor and also saves the output in userlist.txt

$ who | tee /dev/tty | wc -l
This command will display the output of the who command in the screen and counts the number of lines with the wc command and displays the count also
the tee command should have the argument which is the file name.
/dev/tty – it is a special file which indicates the user's terminal

Filters

A command is referred to as a filter if it can read the input, alter it in some way, and write its output to standard output stream

When a program performs operations on input and writes the result to the standard output, it is called a filter. One of the most common uses of filters is to restructure output

UNIX has a large number of filters. Some useful ones are the commands

- awk
- grep
- sed

Simple Filters

grep:

- used to match patterns in a file
- used for searching file
- scans its input for a pattern and displays lines containing the pattern
- grep acts as filter
- grep can be used along with pipe command
- It does not change the content of the file

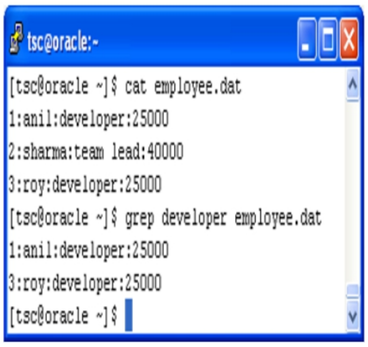
Filters using grep

syntax:

- grep [options] pattern [filename[s]]
  - searches for pattern in one or more filename(s), or the standard input if no filename is specified

in the example, “developer” is the pattern to be searched and “employee.dat” is the filename where the search should happen

When there is more than 1 word in the pattern, enclose the pattern within double quotes



```
[tsc@oracle ~]$ cat employee.dat
1:anil:developer:25000
2:sharma:team lead:40000
3:roy:developer:25000
[tsc@oracle ~]$ grep developer employee.dat
1:anil:developer:25000
3:roy:developer:25000
[tsc@oracle ~]$
```

Filters using grep

- Options in grep:
  - -i → ignoring the case
  - -v → selects all lines except those containing the pattern
  - -n → displays the line number along with the matching content
  - -c → displays the count
  - -l → displays the filenames containing the pattern
- options to explore:
  - -x, -f

Filters using grep continued...

Basic regular expression

- * zero or more occurrence of the previous character

  example : g* → g gg ggg ggg.....
- . single character
- [ ] single character within the group
- [^] single character which is not part of the group
- ^$ lines containing nothing
- ^pat pattern pat as the beginning of the line
- pat$ pattern pat at the end of the line

Filter using cut

- splitting a file vertically
- remove sections from each line of files
- cut can be used along with pipe

syntax:

- cut option[s] file[s]

options in cut:

- -f → fields to be cut
- -d → delimiter

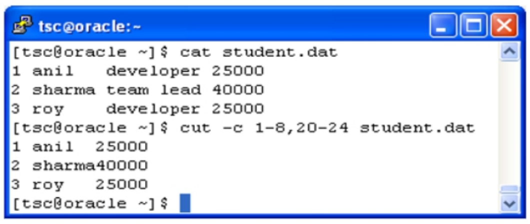
(both -f and -d are used when there is a special character that separates the column.)

- -c → characters. Used when the number of columns are not equal in each line

Filter using cut

example:

-c is used when the file contains fixed length record

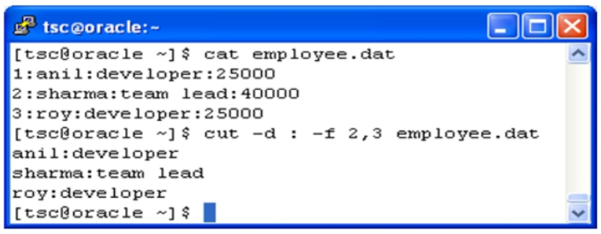


```
[tsc@oracle ~]$ cat student.dat
1 anil   developer 25000
2 sharma team lead 40000
3 roy    developer 25000
[tsc@oracle ~]$ cut -c 1-8,20-24 student.dat
1 anil  25000
2 sharma40000
3 roy   25000
[tsc@oracle ~]$
```

Filter using cut

Example:

- -d and -f are used when the columns are separated by a delimiter
- In the given example ":" is the delimiter



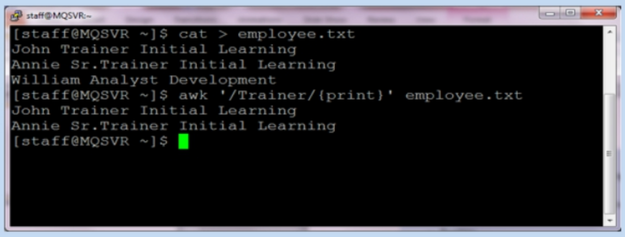
```
[tsc@oracle ~]$ cat employee.dat
1:anil:developer:25000
2:sharma:team lead:40000
3:roy:developer:25000
[tsc@oracle ~]$ cut -d : -f 2,3 employee.dat
anil:developer
sharma:team lead
roy:developer
[tsc@oracle ~]$
```

Advanced Filters

awk

- It is an interpreted programming language, which focuses on processing text
- It is used to execute complex pattern-matching operations on streams of textual data
- **Syntax**

  awk [-F *fs*] [-v var=*value*] ['*prog*' | -f *progfile*] [*file* ...]



```
[staff@MQSVR ~]$ cat > employee.txt
John Trainer Initial Learning
Annie Sr.Trainer Initial Learning
William Analyst Development
[staff@MQSVR ~]$ awk '/Trainer/{print}' employee.txt
John Trainer Initial Learning
Annie Sr.Trainer Initial Learning
[staff@MQSVR ~]$
```