# UNIX

## Filters

# Lesson Objectives

- **In this lesson, you will learn:**
  - Filter commands in UNIX:
    - Simple Filters
    - Advance Filters

# What is a Filter?

- **Filters are central tools of the UNIX tool kit.**
- **Commands work as follows:**
  - Accept some data as input.
  - Perform some manipulation on the inputted data.
  - Produce some output.
- **Most of them work on set of records, with each field of a record delimited by a suitable delimiter.**
- **When used in combination, they can perform complex tasks too.**

# head Command

- **The** head **command, by default, will display the first 10 lines of a file.**
  - **Example 1:** To display fist 10 lines from file employee:

```
$head employee
```

  - **Example 2:** To display first 5 lines from file employee:

```
$head –5 employee
```

- **Single command can be used to display lines from more than one file.**

```
$ head -1 PuneEmp PKPEmp
```

# tail Command

- **The** tail **command is useful to display last few lines or characters of the file.**
  - **Example 1:** To display last ten lines from employee:

```
$tail employee
```

  - **Example 2:** To display last seven lines:

```
$tail –7 employee
```

  - **Example 3:** To display lines from the 10th line till end of the file:

```
$tail +10 employee
```

  - **Example 4:** To display last 5 characters of the file:

```
$tail -5c employee
```

# cut Command

- **The** cut **command retrieves selected fields from a file.**

```
$ cut [options] <filename>
```

- Options :
    - -c : selects columns specified by list
    - -f : selects fields specified by list
    - -d : field delimiter (default is tab)

# cut Command

- **Example 1:** To display 2nd and 3rd field from file bookDetails.lst:

```
$ cut -d"|" -f2,3 bookDetails.lst
```

- **Example 2:** To display characters from 1st to 4th and 31st to 35th from file bookDetails.lst :

```
$ cut -c1-4,31-35 bookDetails.lst
```

# paste Command

- **The paste command is used for horizontal merging of files.**

```
$paste <file1><file2><Enter>
```

  - Options : -d (Field delimiter)
  - **Example 1:** To paste enum.lst and ename.lst files:

```
$ paste enum.lst ename.lst
```

  - **Example 2:** To paste enum.lst and ename.lst files with ‘|’ character as delimiter:

```
$ paste –d’l’ enum.lst ename.lst
```

# sort Command

- **The** sort **command is useful to sort file in ascending order.**

```
$sort <filename>
```

- Options are:
  - -r : Reverse order
  - -n : Numeric sort
  - -f : Omit the difference between Upper and lower case alphabets
  - -t : Specify delimiter
  - -k : to specify fields as primary or secondary key
- Example:

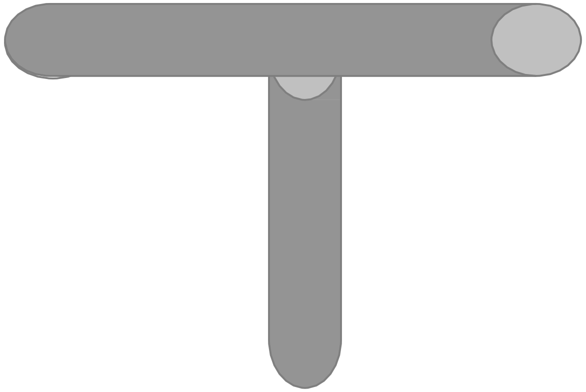
```
$ sort -t"|" +1 bookDetails.lst
$sort –k3,3  -k2,2 employee
```

# uniq Command

- **The** uniq **command fetches only one copy of redundant records and writes the same to standard output.**
  - –u option: It selects only non-repeated lines.
  - –d option: It selects only one copy of repeated line.
  - -c option: It gives a count of occurrences.
- **To find unique values, the file has to be sorted on that field.**
  - **Example:** To find unique values from file duplist.lst

```
$ uniq duplist.lst
```

# tee Command



Standard Input

Standard Output

Output file

- To display contents of file employee on screen as well as save it in the file:

```
$ tee user.txt < employee
```

# find Command

- **The** find **command locates files.**

```
find <path list>  <selection criteria> <action>
```

- **Example 1:** To locate the file named **.profile** starting at the root directory in the system **-print** specify the action:

```
$ find / -name .profile –print
```

- **Example 2:** To locate the file named myfile starting at the root directory in the system

```
find / -type f -name "myfile" -print
```

# grep Command

- **The syntax for** grep **command is as follows:**

```
grep <options> <pattern> <filename(s)>
```

  - **Example:** The following example will search for the string Unix in the file **books.lst**. The lines which match the pattern will be displayed.

```
grep 'Unix' books.lst
```

# grep Command

- **Options of grep:**
  - c : It displays count of lines which match the pattern.
  - n : It displays lines with the number of the line in the text file which match the pattern.
  - v : It displays all lines which do not match pattern.
  - i : It ignores case while matching pattern.
  - -w : It forces grep to select only those lines containing matches that form whole words

# grep Command

- **Example 1:** To print all lines containing “rose” regardless of case:

```
$grep -i rose flower.txt
```

- **Example 2:** To print all lines containing “rose” as a word:

```
$grep -w rose flower.txt
```

- **Example 3:** To print all lines not  containing “rose”:

```
$grep -v rose flower.txt
```

# grep Command

- **Regular Expression:**

| Expression | Description |
|---|---|
| **^ (Caret)** | **match expression at the start of a line, as in ^A.** |
| **$ (Question)** | **match expression at the end of a line, as in A$.** |
| **\ (Back Slash)** | **turn off the special meaning of the next character, as in \^.** |
| **[ ] (Brackets)** | **match any** one **of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9].** |
| **[^ ]** | **match any one character** except **those enclosed in [ ], as in [^0-9].** |
| **. (Period)** | **match a single character of any value, except end of line.** |
| *** (Asterisk)** | **match zero or more of the preceding character or expression.** |
| **\{x,y\}** | **match** x **to** y **occurrences of the preceding.** |
| **\{x\}** | **match exactly** x **occurrences of the preceding.** |
| **\{x,\}** | **match** x **or more occurrences of the preceding.** |

# grep Command

- **Examples of Regular Expression:**

| Example | Description |
|---|---|
| grep “smile“ files | search *files* for lines with ‘smile’ |
| grep '^smile' files | 'smile' at the start of a line |
| grep 'smile$' files | 'smile' at the end of a line |
| grep '^smile$' files | lines containing only 'smile' |
| grep '\^s' files | lines starting with '^s', "\" escapes the ^ |
| grep '[Ss]mile' files | search for ‘Smile' or 'smile' |
| grep 'B[oO][bB]' files | search for BOB, Bob, BOb or BoB |
| grep '^$' files | search for blank lines |
| grep '[0-9][0-9]' file | search for pairs of numeric digits |

# fgrep Command

- **The fgrep command is similar to grep command.**
- **Syntax:**

```
$fgrep  [ -e pattern_list] [-f pattern-file] [pattern] [Search file]
```

- **The** fgrep **command is useful to search files for one or more patterns, which cannot be combined together.**
- **It does not use regular expressions. Instead, it does direct string comparison to find matching lines of text in the input.**

# fgrep Command

- **Options of fgrep command:**
  - -e pattern_list :
    - It searches for a string in pattern-list.
  - -f pattern-file :
    - It takes the list of patterns from pattern-file.
  - pattern
    - It specifies a pattern to be used during the search for input.
    - It is same as grep command.
  - E.g To search employee file for all patterns stored in mypattern file

```
$ fgrep –f mypattern employee.lst
```

# Summary

- **In this lesson, you have learnt:**
  - The head and tail filter commands filter the file horizontally.
  - The cut and paste commands filter the file vertically.
  - -m option of sort command is used to merge two sorted files.
  - The tee command helps us to send o/p to standard o/p as well as to file.
  - grep, fgrep, and egrep commands use to search files for some pattern.

# Review Questions

- **Question 1: ____command to display directory listing on screen as well as store it in dirlist.lst.**
- **Question 2: ____filter commands filter file vertically?**
- **Question 3: ____filter commands filter file horizontally?**