Unit 4
Filters and Regular Expressions

Dr.S.Thenmozhi

Filters and Pipes

- Filter is any command that gets its input from the standard input stream, manipulates the input, and then sends the result to the output stream
- Filters can be used in the left of the pipe, right of the pipe or between two pipes.
- Because filter can send output to the monitor, it can be used in the left of the pipe.
- Because filter can receive input from the keyboard, it can be used in the right of the pipe.

Concatenating Files

- The cat command writes the file contents to the standard output
- It can take multiple files as input
- When multiple files are given, it takes one after the another
- The result becomes as one output and it can be saved in one file also
- The cat command is to catenate multiple file. But when given with one file, it catenates with null file as second.
- It does not give automatic pause after the end of the screen
- It does not check for the filetype before catenating. It just catenates

- Cat is used to create a file. There is only one input which comes from the keyboard.
- As we want to save the contents to the file, we redirect to the file
- End of file is identified with ^d
- Four categories: visual characters, buffered output, missing files, numbered lines
- -v allows us to see control characters with the exception of tab, newline, form feed characters
- -vt – the tabs appear as ^I
- To supress blank lines -s
- Numbered lines - -n

Filtering Beginning of File - head

- Displaying the beginning of a file to the std. output.
- If no files is specified, it receives from the std. input
- It can work with multiple files also
- When used without any options displays first ten lines of a file
- -n to specify the no. of lines to be displayed [counts from beginning]
- head –n 3 stud.dat or head -3 stud.dat
- What does the following command will do?
  - gedit `ls –t | head -1`
- [opens up the last edited file]

Filtering End of File - tail

- Displaying the end of a file
- When used without any options displays first ten lines of a file
- -n to specify the no. of lines to be displayed [counts from end]
  - tail –n 3 stud.dat or tail -3 stud.dat
- +n counts from the beginning
  - tail -n +11 stud.dat [displays from line number 11, skips first 10 lines]
- -c extract bytes rather than lines

Filter Columns - cut

- Split a file vertically or column wise
- cut -c => To cut/extract specific columns
- Ranges can also be specified
- Mutiple columns separated by comma
- cut –c -3,6-22,28-34,55- stud.dat
- -f => to cut fields -d =>specify delimiter
- cut –d "|" -f 2,3 stud.dat or cut –d \| -f 2,3 stud.dat
- What does this command will do?
  - who | cut –d " " –f 1
- [cuts the first field from the result of who command]

Combines columns - paste

- To merge/paste contents/combines lines together
- It is done vertically
- paste stud1.dat stud2.dat
- pastes both files vertically
- -d – to paste with delimiter
- paste -d "\t |" stud1.dat stud2.dat
- To specify the input coming from std. input put – instead of a filename

Ordering the file - sort

- Orders a file
- It identifies fields and sort on specified fields
- Sort reorders lines in ASCII collating sequence – numerals, uppercase letters, finally lowercase letters
- sort stud.dat
- The sorting sequence can be altered by appropriate options
- Sorting can be done one more than one fields

Sort Options

- -t *char* – uses delimiter char to identify fields
- -k n – sorts on the nth field
- -k m,n – sorts on the mth field and then on nth field
- -k m.n – starts on the nth col of mth field
- -u – removes repeated lines
- -n – sorts numerically
- -r – reverse sort order
- -f – case insensitive sort
- -c – checks if file is sorted
- -o *filename* – places output in a file *filename*
- -m *list*- merges sorted files in list

- sort –t "|" –k 2 stud.dat // sorting on primary key
- sort –t "|" –k 3,3 –k 2,2 stud.dat // sorting on secondary key – start and end should be specified
- sort –t "|" –k 5.7,5.8 stud.dat // specifies the column position of the field specifies 7^{th} ,8^{th} column of the 5^{th} field

Uniq

- Locate repeated and non-repeated lines
- Special tool instead of sort command with –u option
- uniq stud.dat //fetches unique lines in the file to std. Output
- Uniq requires the input as sorted input
- So, uniq can be piped after sort command
- sort stud.dat | uniq
- sort stud.dat | uniq – stud1.dat // two filename one the source and other the destination i.e output will be written into stud1.dat

- uniq Options
- -u – selecting non- repeated lines
- -d – selects the duplicated lines
- -c – counts the frequency of occurrence of all lines

Counts words lines and characters

- Wc command counts the number of characters, words, lines in one or more documents
- Character count includes \n character also
- -c counts only characters
- -l counts only lines
- -w counts only words

Comparing Files

- Cmp – compare, Diff – difference, Comm-common
- cmp – examines files byte by byte, it stops at the first byte that is different
- The byte number of the first difference is reported
- -l lists all difference in the files, byte by byte
- -s to supress output. No output is displayed, the results can be determined by exit status of command. If exit status is 0, then both files are identical. If 1 then there is difference.

- diff – Diff shows the line-by-line difference between two files or directories
- -b ignore tailing blanks
- -w ignore white spaces
- -I ignore case
- Each difference is displayed
- comm – Displays the lines that are identical. It compares line by line.
- The results are displayed in three column, first represent the unique lines of file1, second represents the unique line of file2 and the last column represents the lines that are common in both

Translating Characters - tr

- tr filter manipulates on individual characters in a line
- Format: tr options exp1 exp2 std input
- tr takes input only from the std input.
- It does not take filename as argument
- By default it translates each character in expression1 to its mapped counterpart in expression2.
- The first character in first is replaced with the first in the second and so on
- The length of the two expressions should be equal

- Changing case of text
  - head –n 3 stud.dat | tr A-Z a-z
- Deleting characters -d
  - tr –d ‘|’ < stud.dat | head –n 3 ?????
- Compressing multiple consecutive characters –s (squeezes)
  - tr –s “ “ < stud.dat |head –n 3
- Complementing the action –c
  - Tr –c –d [:digits:] <stud.dat |head –n 3

Regular Expressions

- Regular expressions is a pattern consisting of a sequence of characters that is matched against text.
- grep and sed are unix utilities that use regular expressions
- Regular expression is like a mathematical expression. A mathematical expression is made of operands and operators. Like wise, regular expressions is made of Atoms and operators.
- Atoms is what text we are looking for and where it is to be found
- Operator is not required in all expressions

Atoms

- 4 types
  - Single character
  - Dot
  - Class
  - Anchor
- Single character – Simplest atom is single character. It is oen single character that has to be matched in the text. If it is found in the text the match is successful else it is unsuccessful.

- Dot matches any single character except the newline character (\n).
- It can work with other atoms
- Eg: a. is a single character atom and dot atom combined. It matches any pair of character with first letter a and followed by any other character
- Class defines set of ASCII characters. The character set is enclosed in brackets.
- Eg: [ABC] can math either A or B or C
- Range of characters can be given with - . ^ can be used for exclusion.
- Escape character – It is used when the matching character is one of the other two tokens.
- Eg: [aeiou\-] means aeiou or hyphen

- Anchors – used to line up the pattern with a particular part of a string.
- Defines where the next character in the pattern must be located in the text.
- Four types – beginning of line ^, End of line $, Beginning of word \<, end of word \>
- Eg: ^Q checks for line which starts with Q
- Eg: g\> checks for word which ends with g

Operators

- To make the regular expressions more powerful, we can combine atoms with operators.
- Regular exp. Operators plays the same role as mathematical operators
- 4 different categories
  - Sequence operators
  - Alternation operators
  - Repetition operators
  - Group operators

- Sequence operator – series of atoms, an invisible sequence operator is applied between them
- Alternation - | - define one or more alternatives
- Eg: A|B
- Repetition – (\{m,n\}) – set of escaped braces that contains two numbers separated by comma.
- It specifies that the atoms or expression immediately before the repetition may be repeated. Ie. It matches previous character m to n times
- Eg: A (\{3,5\}) - matches A 3 to 5 times
- Eg:A(\{3 \}) – matches A 3 times
- Eg: A(\{3, \}) – minimum 3 times, max n times
- Eg: A(\{ , 3 \}) – minimum 0 to 3 times

Short Form operators

- * - repeat an atom 0 or more times
  - It is equivalent to (\{ 0, \})
- + - atom should appear 1 or more times
  - It is equivalent to (\{ 1, \})
- ? – pattern 0 or one time
  - It is equivalent to (\{ 0, 1\})

- Group operators –is a pair of opening and closing paranthesis
- Eg: A(BC)(\{ 3 \}) – Starts with A and followed by 3 times of BC ie., it matches for ABCBCBC

Grep

- Global regular expression print
- It is used to search the input file that match the regular expression and write them to the standard output (monitor)
  - Copies the input line to the pattern space. pattern space is a buffer that can hold only one text line.
  - Applies the regular expression to the pattern space.
  - If there is a pattern match copies line to the std.output.
- Repeats the above three steps for all the lines.
- Grep is a search utility.
- If no pattern is matched it prints nothing
- Grep is a filter. Hence it can be used right or left side of the pipe

- Grep cannot be used to add, delete or change a line
- Grep cannot print part of the line
- Grep cannot read part of a file
- Grep cannot select a line based on contents of previous or next line
- Grep Family – grep, egrep, fgrep

Atoms	grep	fgrep	egrep	Operators	grep	fgrep	egrep
Character	√	√	√	Sequence	√	√	√
Dot	√		√	Repetition	All but ?		* ? +
Class	√		√	Alternation			√
Anchors	√		^ $	Group			√

- grep Options
  - -i – ignore the case
  - -v –inverse option. To display all lines expect the line where pattern is found
  - -n – to display line numbers containing the pattern
  - -c – counting lines containing the pattern
  - -l – display only the filename that contains the pattern
  - -e – to provide multiple patterns
  - -f – taking patterns from a file