UNIX FILTERS

- Many Unix files have lines containing fields –string of characters represented by a meaningful entity.Some commands expect these fields to be separated by a suitable delimiter that's not used by the data.
- Typically this delimeter is a : (as in /etc/passwd and $PATH), but we have used the |(pipe) as delimeter for some of the sample files.
- Filters Reviewed
- Filters were introduced in earlier chapters as a category of commands that take advantage of the shell's redirecton features.
- A filter has the capacity of reading standard input when used without a filename as argument and from the file otherwise.

UNIX FILTERS

- The Piping mechanism of Shell lets the standard output of one filter serve as standard input to another.
- The pr command prepares a file for printing by adding suitable headers,footers and formatted text.
- A simple invocation of the command is to use it with a filename as argument.
- $pr dept.lst
- May 06 10:38 1997 dept.lst page 1
- pr adds five lines of margin at the top and five at the bottom.The lower portion of the page has not been shown in the examples.
- The header shows the date and time of last modfcn of the file along with the filename and pagenumber.

UNIX FILTERS

- pr options
- pr –k option (Where K is an integer) prints in k columns .If a program outputs a series of 20 numbers one in each line the this option can make good use of the screen's empty spaces.
- pr is a filter ,It can obtain it's input from the standard output of another program.
- $a.out | pr –t -5
- 0 4 8 12 16
- 1 5 9 13 17
- 2 6 10 14 18
- 3 7 11 15 19

UNIX FILTERS

- If you are not using the –t option then you can have a header of your choice with the –h option. This option is followed by the header string.
- -d DoubleSpace's input,Reduces clutter.
- -n Number Lines,helps in debugging code.
- -o n offsets Lines by n spaces, increases Left margin of page.
- Combine these various options to produce just the format you need.
- $pr –t –n -d –o 10 dept.lst
- There’s an option that uses a number prefixed by a + to print a specific page number.
- pr +10 chap01 starts printing from page 10
- pr -1 54 chap01 page Length set to 54 Lines.

head Displaying the Beginning of a File

- The head command as the name implies displays the top of the file.When used without an option ,it displays the first ten lines of the file.
- $head emp.lst
- You can use the –n option to specify the line count and display ,so the first three lines of the file.
- $ head -n 3 emp.lst
- Head can be used in imaginative ways.
- Consider that you are resuming an editing session the next day and find out that you are unable to recall the name of the file you last edited.

Head command

- Since ls –t displays filenames in order of their modification time the job is easily done by picking up the first filename from the list and using it as an argument to vi.
- This requires command substitution
- vi `ls -t | head –n 1` Opens last Modification of File.
-

tail command

- tail: Displaying the end of a File:
- The tail command displays the end of a file .It provides an additional method of addressing lines and like head it displays the last ten lines by default.
- The last three lines are displayed in this way.
- $tail -n 3 emp.lst
- 3564|Sudhir Agarwal|Execution |Personnel |6/7/1987|7500
- 2355|J.b.saxena |G.M.|marketing |12/3/45|8000
- 1110|v.k.Agarwal|G.M.|marketing|31/12/1980|9000.
- It can also address lines from the beginning of the file instead of the end.
- The +count option allow you to do that ,where count represents the line number from where the selection should begin.

tail command

- Since the File contains 15 Lines ,selecting the last five implies using.
- $tail +11 emp.lst 11^{th} Line onwards,possible with + symbol.
- tail options
- Monitoring File Growth(-f)
- Many Unix programs constantly write to the System's Logfiles as long as they are running.
- System adminstrators need to monitor the growth of these filesto view the latest messages.
- tail offers the –f (follow) option for this purpose.
- tail –f /oracle/app/oracle/product/8.1/Orainst/install.log.

Extracting Bytes Rather than Lines(-c)

- Tail will support the –c option followed by a positive or Negative Integers depending on whether the extraction is performed relative to the beginning or end of a file.
- $tail -c -512 foo Copies last 512 bytes from foo.
- $tail -c +512 foo Copies everything after skipping 511 bytes.

Grep Command Pattern Matching

- You often need to search for a file for a pattern ,either to see the Lines containing (or not containing) it or have it replaced with something else.
- Grep takes care of all search requirements you may have .
- Grep command: Searching for a Pattern
- Unix has a special family of commands for handling search requirement and the principal member of this family is the grep command.
- grep scans its Input for a pattern and displays the Lines containing the pattern the Line Numbers or Filenames where the Pattern occurs.
- grep searches for pattern in one or more filenames or the standard input if no filename is specified.

Grep Command

- The First argument (barring the options) is the pattern and the remaining arguments are filenames.
- $ grep “Sales” emp.lst
- 2233|a.k.Shukla|g.m.|Sales|12/12/1952|6000
- Like that the pattern is matched for a File.
- Because grep is also a filter it can search its standard Input for the pattern,and also save the standard output in a File.
- who | grep Kumar >foo
- Though we have used the pattern both with (“sales”) and without quotes

It is generally safe to quote the pattern .In fact quoting is essential if the search string consists of more than one word or uses any of the shell’s character like *,$ etc.

Grep command

- When grep is used with multiple filenames ,It displays the filenames along with the Output.
- In the next example grep searches two files .Don’t bother about what they contain just observe each line is preceded by the filename.
- $grep “director” emp1.lst emp2.lst
- Emp1.lst:1006
- Emp2.lst:9876
- ignoring case(-i) When you look for a name ,but are not sure of the case ,use the –i (ignore) option.This option ignores case for pattern matching.
- $grep -i `Agarwal’ emp.lst
- 9876| jai Agarwal|director |production|12/8/50|7000

Grep Command

- Deleting Lines(-v) grep can play a Inverse role too;The –v option selects all lines except those containing the pattern.
- Thus,you can create a file otherlist containing all but directors.
- $grep -v ‘director’ emp.lst >otherlist
- $wc –l otherlist
- Displaying Line Numbers
- The –n (number) option displays the Line Numbers containing the pattern,along with the Lines.
- $grep –n ‘marketing’ emp.lst
- The line numbers are shown at the Beginning of each line ,separated from the actual Line by a:

Counting Lines Containing Pattern(-c)

- The –c option counts the number of Lines containing the pattern(Which is not the same as number of occurrences.
- $ grep –c “director” emp.lst
- If you use this command with Multiple Files ,The filename is prefixed to the line count:
- $grep -c director emp*.lst
- emp.lst:4 emp2.lst:2 emp1.lst:2 empold.lst:4
- $grep -l “manager” *.lst
- Desig.lst
- Emp.lst
- emp1.lst

Matching Multiple patterns (-e)

- $grep –e ‘Agarwal’ –e “Aggarwal” –e “Agrawal” emp.lst
- You could Question the wisdom of entering such a long command line when the pattern don’t differ much from one another.
- Yes grep supports sophisticated pattern matching techniques that can display the same lines but with a single expression.
- Taking Patterns from a File(-f)
- You can place all the three patterns in a separate file,One pattern per Line. Grep uses the –f option to take patterns from a file.
- $ grep -f pattern.lst emp.lst

Grep Patterns

Option	Significance
- i	Ignores case for matching
-v	Does’nt display lines matching expressions.
-n	Display’s Line Numbers along with lines.
-c	Display’s Count of number of occurences.
-l	Display’s List of Filenames only.
-e exp	Specifies expression with this option can use Multiple times.Also used for matching expression beginning with hypen.
-f file	Takes pattern from file ,one per Line.

Basic Regular Expressions

- To view the file emp.lst once again and locate all the agarwals.On close examination you'll see three Lines containing similar ,but not identical Agarwal,Aggarwal and Agrawal.
- It's tedious to specify each pattern separately with the –e option.
- Like the shell's wildcard which match similar filename with a single expression,grep uses an expression of a different type to match a group of similar patterns.
- Unlike wild cards however this expression is a feature of the command that uses it and has nothing to do with shell.
- It uses an elaborate metacharacter set ,overshadowing the shell's wild-card and can perform amazing matches.
-

Basic Regular Expression

- Regular Expression take care of some common query and substitution requirements.You may want to replace multiple spaces with a single space or display lines that begin with a #.
- Grep supports basic Regular expression (BRE) by default and Extended Regular expression (ERE) with the –E option.
- Regular Expressions are interpreted by the command and not by the shell.

Basic Regular Expression

Symbols Or Expression	Matches
*	Zero or More Occurrences of Previous character
g*	Nothing or g,gg,ggg, etc
.	A single character
.*	Nothing or any number of characters
[pqr]	A single character p,q or r
[c1-c2]	A single character within the Ascii range represented by C1 and C2
[1-3]	A digit between 1 and 3
[^pqr]	A single character Which is not a p,q or r
[^a-z A-Z]	A Non-alphabetic character
^pat	Pattern pat at the beginning of Line
Pat$	Pattern pat at end of the Line
Bash$	Bash at the end of Line
^$	Lines Containing Nothing

Character Class

- A regular expression let's you specify a group of characters enclosed within a pair of rectangular brackets [ ], in which case the match is performed for a single character in the group.
- Thus the expression [ra] matches either an r or an a. The metacharacters [and] can now be used to match Agarwal and Agrawal
- [aA]g[ar][ar]wal matches the two names. The character class [Aa]

Matches the letter a in both lowercase and uppercase.The model [ar][ar] matches any of the four characters.

aa ar ra rr of which second and third are relevant to present problem.

Character Class

- grep “[Aa]g[ar][ar]wal” emp.lst
- 3564|Sudhir Agarwal|executive |personnel |7/6/47|7500
- 1867|v.k.Agrawal |g.m.|marketing|12/11/50|9000
- A single pattern has matched two similar Strings,that’s what regular expression is all about.
- The pattern [a-zA-Z0-9] matches a single alphanumeric character.when you use a range ,make sure that the character on the Left of the hypen has a lower asci value than the one on the right.

The * Operator

- The *(asterisk) refers to the immediately preceding character. However it's Interpretation is the trickiest of the lot as it bears absolutely no resemblance whatsoever with the * used by wild cards or Dos.
- Eg g* matches the single character g,or any number of g's.Thus apart from this null string it also matches the following strings g,gg,ggg.
- Mark the keywords "Zero or more occurrence of the previous character" that are used to describe the significance of the *.

The * Operator

- Observe carefully that the Regular expression [Aa]gg*[ar][ar]wal

matches all three patterns.

.) grep "[Aa]gg*[ar][ar]wal" emp.lst

2476|anil Aggarwal|manager

3564 |Sudhir Agarwal|executive

0110|v.k.Agrawal |G.m.

A simple regular expression matches all three names! You don't have to use the –e opton three times to get the result.

The Regular Expression .*

- The dot along with the * (.*) constitutes a very useful regular expression .It signifies any number of characters or none.
- Consider that you want to lookup the name j.saxena but are not sure,whether it actually exists in the file as j.b.saxena or as joginder saxena.
- $ grep “j.*saxena” emp.lst
- 2345|j.b.saxena |g.m. | marketing |3/12/45 |8000
- Note that if you look for the name j.b.saxena,the expression should be

j\.b\.saxena .The dots need to be escaped here with the \, the same character you used in the shell for despecializing the next character.

Specifying Pattern Locations

- ^ (caret) For matching at the Beginning of a line.
- $ -For Matching at the end of Line.
- Eg grep “^2” emp.lst
- It matches 2233 2265 2476
- $grep “7…$” emp.lst
- 9876|Jai sharma|directr|prodcn|3/12/50|7000
- Search employee Id that don’t begin with 2.
- grep “^[^2]” emp.lst
- ls –l|grep “^d”

The ^ caret in Regular Expression

- The caret (^) has a triple role to play in regular expressions when placed at the beginning of a character class
- Eg [^a-z] it negates every character of the class.
- When placed outside it, and at the beginning of the expression e,g

^2… It is matched at the beginning of the line.

. At at any other location (e.g. a^b) it matches Literally.

WILD-CARD Characters

- The meta characters that are used to construct the generalized pattern for matching filenames belong to a category called wild-cards.

Wild-Card	Matches
*	Any Number of characters Including None
?	A single character
[ijk]	A single character either an I,j or k
[x-z]	A single character that is within the ASCII range of charcaters X and Z.
[!ijk]	A single Character that is not an I,j or k [not in C shell]
[!x-z]	A single character that is not within the ASCII range of characters x and z
{pat1,pat2}	Pat1.pat2 etc.

Shell's Offerings

- The Bourne family comprising the Bourne shell (/bin/sh) and its derivatives –the korn shell(/bin/ksh) and bash [/bin/bash]
- Pattern Matching-The Wild cards
- Try listing all filenames beginning with chap.The most obvious solution is to specify all the filenames separately.
- $ls chap chap01 chap02 chap03
- If filenames are similar (as above) we can use the facility offered by the shell of representing them by a single pattern or Model.
- For instance chap* represents all filenames beginning with chap.
- The pattern is framed with ordinary characters (like chap) and a metacharacter(like *) using well-defined rules.

The * and ?

- In the First chapter ,you used the command ls chap* to list all filenames

beginning with chap.The metacharacter * is one of the characters of the shell's wild card set.

$ls chap*
chap chap01 chap02 chap03 chap04 chap15

- When the shell encounters this command Line ,It identifies the * immediately as a wild-card.
- Caution: Be careful when you use the * with rm to remove files.

If instead of typing rm *.o which removes all the object files .

You inadvertently introduce a space b/wn * and .o $rm * .o

$ls chap? Or ls chap??

Matching the Dot

- The * does'nt match all Files beginning with a .(dot).
- If you want to list all hidden filenames in your directory having at least three characters after the dot must be matched explicitly.

$ ls .???*

.bash_profile .exrc .netscape .profile

However if the filename contains a dot anywhere but not at the beginning it need not be matched explicitly.

$ls emp*lst

Emp.lst emp1.lst emp2.lst

Negating the Character Class(!)

- How about framing a pattern that reverses the above matching criteria?
- *.[!co] Matches all filenames with a single character extension but not the .c or .o files.
- [!a-zA-Z] Matches all filenames that don’t begi with an alphabetic character.
- Even though the character class is meant to be used with a group of characters it’s the only way you can negate a match for a single character.
- * .[!0] Not the .o files.

Character Class

- The pattern [abcd] is a character class, and it matches a single character a,b,c or d. This can be combined with any string or another wild-card expression ,so selecting chap01,chap02 and chap04 now becomes a simple matter.
- $ls chap0[124]
- chap01 chap02 chap04 Range specification is also possible inside the class with a –(hyphen),the two characters on either side of it from the range of characters to be matched.
- $ls chap0[1-4]
- $ls chap[x-z]

Rounding Up

- Some of the Wild-card characters have different meanings , depending on where they are placed in the pattern.
- The * and ? Lose their meaning when used inside the class and are matched literally.
- Similarly – and ! also lose their significance when placed outside the class.Whether or not you are able to devise a suitable pattern that matches a group of filenames depends.
- Ls *.c List all files with extension .c
- mv *../bin Moves all Files to bin subdirectory of parent directory
- cp foo foo* copies foo to foo* (* loses meaning here).
- Cp ????? Progs copies to progs directory all files with six character names

Rounding Up

- It’s the shell that interprets these expressions .
- In the Next topic we discuss two techniques of removing the special meaning of metacharacters.

Escaping and Quoting

- You might well think that if the shell uses some special characters to match filenames then the filename themselves must not contain any of these characters.
- One of the sample lines featured in the previous section shows how easy it is to actually create a file named foo*.Because the pattern foo*

  also matches the filename foo* the shell will include.

- $ls chap*
- chap chap* chap01 chap02 chap03 chap04
- Trying rm chap* would be dangerous ,it would remove other filenames beginning with chap also.

Escaping and Quoting

- Escaping: Providing a \(backslash) before the wild card to remove

(escape) its special meaning.

Quoting: Enclosing the wild-card or even the entire pattern within quotes

(like ‘chap*’) .Anything within these quotes (barring few exceptions) are left alone by the shell and not interpreted.

Placing a \ immediately before a metacharacter turns off its special meaning.

For instance in the pattern * the \ tells the shell that the asterisk has to be matched Literally instead of being interpreted as a metacharacter.

This means that we can remove the file chap* without affecting the other filenames that also begin with chap by using .

```
$rm  chap\*
```

Escaping and Quoting

- \ suppresses the wild-card nature of the *,thus preventing the shell from performing filename expansion on it.
- This Feature is known as escaping.
- If you have files chap01, chap02 and chap03 in your current directory and still dare to create a file chap0\[1-3\]
- Then you should escape the two rectangular brackets when accessing the file. $ls chap0\[1-3\]
- $rm chap0\[1-3\] .
- Escaping the Sequence: Apart from metacharacters there are other characters that are special –like the space character.The shell uses it to delimit command Line arguments.So to remove the file My Document.doc which has a space embedded. $ rm My \ Document.doc