UNIX PROGRAMMING (JNTUK-R16)
UNIT-4: FILTERS

SYLLABUS:
- The Grep Family
- Other Filters
- The Stream Editor Sed
- The AWK Pattern Scanning and processing Language
- Good Files and Good Filters.

Text Books:
1. The Unix programming Environment by Brain W. Kernighan & Rob Pike, Pearson.
2. Introduction to Unix Shell Programming by M.G.Venkateshmurthy, Pearson.

- Filter commands accept input data from **stdin** (standard input) and produce output on **stdout** (standard output).
- Filters are the methods to find what we required as output.

1. The Grep Family:
- The grep family consists of three commands
  - a) The grep command
  - b) The egrep command
  - c) The fgrep command
- These are called as Seaching Filters Family.

a) The grep command:
- This command is used to search, select and print specified records or lines from an input line.
- grep means "globally search a regular expression and print it"
- grep command filters the content of a file which makes our search easy
- The word global specifies that the entire input file or standard input is searched for a specified pattern or patterns.
- By default, grep prints matching lines.
- The 'grep' command is also used with pipe (|)

Syntax:

$grep [options] pattern [file1] [file2]

Options used in grep command:

OPTIONS	PURPOSE
-v	Prints all lines that do not match pattern
-i	Matches either upper or lowercase i.e, in case insensitive way
-n	Prints the matched line and its line number
-l	Prints only the names of files with matching lines (letter "l")
-c	Prints only the count of matching lines

Example:

$cat courses

```
Oracle is a DataBase
oracle is a database
Java is an Oop Language
java is an oop language
unix is an operating system
Unix is an Operating System
$
```

```
i) $grep  Unix  courses
Unix is an Operating System
  $
ii)$grep  -i  Unix  courses
unix is an operating system
Unix is an Operating System
  $
iii) $grep  -n  Unix  courses
   6: Unix is an Operating System
$
iv)$grep  -ni  Unix  courses
   5: unix is an operating system
   6: Unix is an Operating System
$
v)$grep  -ci  Unix  courses
2
   $
vi) $grep  "^u"  courses
unix is an operating system
   $
vii) $grep  "m$"  courses
unix is an operating system
Unix is an Operating System
   $
```

b) The egrep command:

- egrep stands for "extended grep"
- grep means "globally search a regular expression and print it"
- It has two additional meta characters, + and ?
- This command is the most powerful member of the grep command family.
- The **advantage** of this command is that **multiple search patterns** can be handled very easily.
- The pipe (|) character is used to mention alternate patterns.

Example:

```
$cat  courses
Oracle is a DataBase
```

```
oracle is a database
Java is an Oop Language
java is an oop language
unix is an operating system
Unix is an Operating System
$
```

```
i) $egrep  (Unix|Java)  courses
     Unix is an Operating System
     Java is an Oop Language
  $
```

egrep's additional characters:

- egrep command has two additional meta characters.
- They are,
  - plus (+)
  - question mark (?)
- The + character matches with one or more instances of the previous character.
- For example, a+ matches any of the patterns like a, aa, aaa, aaaa, …, so on.
- The ? matches with zero or one occurrence of the previous character.
- For example, a? matches with either no character or a single a.

c) The fgrep command:

- fgrep stands for fixed grep or fixed character grep.
- grep means "globally search a regular expression and print it".
- This command uses only fixed characters patterns.
- In other words, it does not allow the use of regular expressions.
- Because this command works with only fixed patterns and does not involve itself in the interpretation of any regular expression.
- It is the fastest among the entire pattern-searching programs
- It is used for searching large files.
- This command also accepts multiple search patterns.
- Whenever multiple search patterns are used, they are separated by a new line character.

Example:

```
$cat  courses
Oracle is a DataBase
oracle is a database
Java is an Oop Language
java is an oop language
unix is an operating system
Unix is an Operating System
$
i) $egrep  Unix
```

```
>|Java  courses
  Unix is an Operating System
  Java is an Oop Language
$
```

2. Other Filters:

- The different types of filters are
  - comm
  - diff
  - head
  - tail
  - nl
  - cut
  - paste
  - sort
  - tr
  - tee

i) The comm command:

- As the name implies, this command prints the lines that are similar or common in two files as output.
- The comparison is done line-by-line and the result is displayed in three columns where the first column consists of lines that are unique in file1, second column consist of lines that are unique in file2 and third column consist of lines that are common in both files.
- It is mandatory that comm performs its comparison between two input files.

Syntax:

```
$comm [-options] file1 file2
```

ii) The diff command:

- This command prints the differences between two files by performing the comparison line-by-line and by comparing first file with the second file.
- When a difference is marked, the first file is altered in order to match it with the second file.
- This command takes its input either from a file or directory and writes the output on the standard output stream.

Syntax:

```
$diff [options] file1/dir1 file2/dir2
```

iii) The head command: (displaying first few lines)

- It is a Line Filter.
- The head command is used to display few lines at the beginning of one or more files.
- This is useful in verifying the contents of a file.
- By default, it displays the first 10 lines (records) of a file.
- It works on multiple files

Syntax:

head [options] [files]

Example:

```
$cat languages
C
C++
Java
Javascript
Python
Ruby
$

$head -3 languages
1 C
2 C++
3 Java
$
```

iv) The tail command: (displaying last few lines)

- It is a Line Filter.
- The tail command is used to view or display last few lines at the end of a file.
- By default, this command displays last 10 lines (records) of a file.
- It will not work on multiple files.

Syntax:

tail [options] [files]

Example:

```
$cat languages
C
C++
Java
Javascript
Python
Ruby
$

$tail -3 languages
4 Javascript
5 Python
6 Ruby
$
```

v) The nl command:

- It is a Numbered Filter.
- It is used to give number to the lines of files

Syntax:

nl [options] [files]

Example:

```
$cat fruits
Apple
Banana
Orange
$

$nl fruits
1 Apple
2 Banana
3 Orange
$
```

vi) The cut command: (splitting files vertically)

- Using this command, required fields or columns can be extracted from a file
- It is a Split Filter.

Syntax:

cut [options] [files]

Example:

```
$cat emp
EmpID  EmpName   Sal
111      aaa        10000
222      bbb        12340
333      ccc        15400
$

$cut -f 3 emp
Sal
10000
12340
15400
$
```

vii) The paste command: (appending files vertically)

- This command is used to merge lines of files.
- It is a Merging Filter.
- This command is used to create new tables or files by gluing together fields or columns from two or more files.

Syntax:

paste [options] [files]

Example:

```
$cat file1
Hi Hello
$

$cat file2
Welcome
$

$paste file1 file2
Hi Hello
Welcome
```

viii) The sort command:

- This command is used to sort lines of text files.
- This command is one of the powerful and a general-purpose tool that is used for sorting information stored in a file.
- It is a Sorting Filter
- In addition to sorting, this command can be used for merging sorted files.
- It takes zero, one or more number of file names as its arguments.

Syntax:

sort [options] [files]

Example:

```
$cat Names
Virat
Dravid
ABD
Dhoni
$

$sort Names
ABD
Dhoni
Dravid
Virat
$
```

ix) The tr command;

- It is used to translate or delete characters.
- It is a Translation Filter.
- Translation includes both substitution as well as deletion of characters or strings.

Syntax:

tr [options] [Sets]

Example:

```
$cat Names2
aaab
bbba
$

$tr 'a' 'A' < Names2
AAAb
bbbA
$
```

x) The tee command:

- It is a bi-redirection Unix Filter.
- It is used to read from standard input and write to standard output and files.
- Bi-redirection means one copy is displayed on the output screen and another copy is on a separate file.

Syntax:

```
tee [options] [files]
```

Example:

```
$cat file3
aaa
bbb
ccc
$

$cat file3 | wc -l | tee file4
3
$

$cat file4
3
$
```

3. The Stream Editor Sed:

- sed is a stream editor for filtering and transforming text.
- sed is one of the most powerful filters.
- It stands for stream editor.
- Despite its name editing, it does not modify the original file.
- It reads the standard input (a keyboard or a file), processes it using a separate file called sed script and writes the result to the standard output.
- A sed script is a file containing a list of instructions to be applied to each line in the input file.
- If the script contains only one line it can be provided as in-line at the command line.

Options in sed command:

- There are three useful options available with sed utility.

OPTION	FUNCTION
-c	It is a default option. It indicates that the script is on the command line
-f	It indicates that the script is in a file which immediately follows this option
-n	It suppresses the automatic output. That is, it will not display the contents of the pattern space

Opeartion of sed:

- sed gives a line number to each line in the input file to address lines in the text.
- It does the following operations for each input line.
  - It copies an input line to a buffer called the pattern space. A pattern space can hold one or more lines at the same time.
  - To each line in the pattern space that matches the specified addresses in the sed instruction, it applies all the instructions in the sed script one by one.
  - If –n option is not specified on the command line, then it copies the pattern space to the standard output (a monitor or a file).
  - It repeats this process for each input line starting with step1.
  - The sed utility also uses a temporary buffer called hold space to hold one or more lines as directed by the sed instructions.

Example:

```
$cat  Sample
1  Java
2  Dotnet
3 Python
4  Swift
5  Angular JS
$

$sed  2q  Sample
1  Java
2  Dotnet
$

$sed  -n  4p  Sample
4  Swift
$
```

4. The AWK Pattern Scanning and processing Language:

- awk is a filter program that was originally developed in 1977 by Aho, Weinberger, and Kernighan as a pattern-scanning language.
- The name awk is derived from the first letters of its developer's surnames.
- It is a programming language with C-like control structures, functions and variables.
- It was designed to work with structured files and text patterns
- One of the very important features of this filter is that it operates at the field level.
- In Linux, awk is available as gawk (GNU awk)

Features of awk:

- Field-oriented file processing
- Regular Expressions
- Predefined Variables
- Numeric Operations
- Comparison Operators
- Arrays
- Control Statements
- Report Generation

Syntax of an awk program statement:

$awk [options] 'program' filelist

- Where the
  - Use of options is optional
  - filelist will have zero or more input filenames
- Program will have one or more statements having the following syntax

Syntax:

pattern {action}

- The pattern component of a program statement indicates the basis for a line or record selection and manipulation

Fields and Records in awk:

- A file is viewed as a collection of Fields and records by the awk utility.
- A field is a data unit that gives some data.
- Each line in a file is a record.
- That is a record is a collection of several fields.
- However, the record contains related data.
- A file that is organized into records is called a data file.

Example:

- Consider the output of ls -l (long list) command in Unix.
- It lists the information about all files in a directory.

```
$ls -l
-rw-r--r--  own1  file1
-rw-r--r--  own1  file2
-rw-r--r--  own1  file3
$
```

- Here, it contain three fields and three records
- The fields are file permissions, owner and filenames
- The records gives information about file1, file2, file3

Output Statements:

- In awk, the output can be displayed using three statements,

print statement;

```
$awk '{print}' info.dat
```

printf statement:

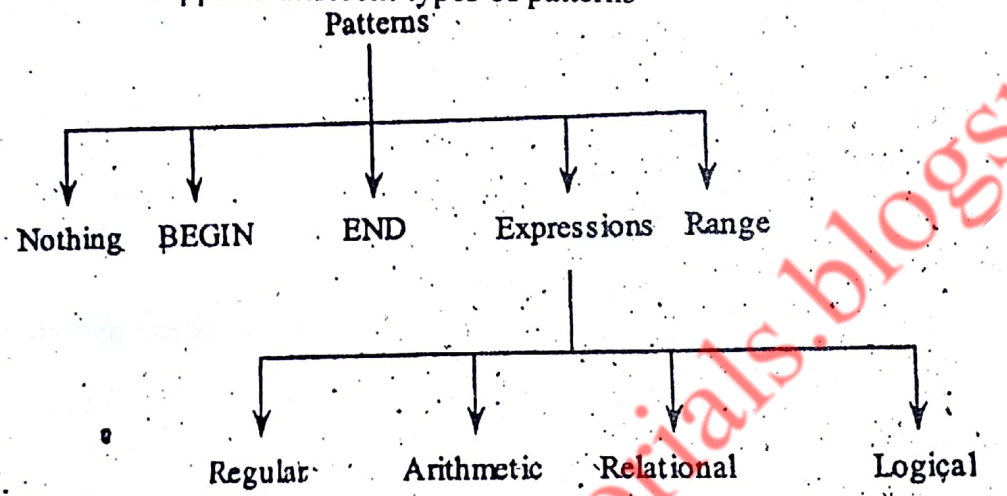
```
$awk '{printf("%S%4d%5.2f\n", $1, $2, $3)}' info.dat
```

sprint statement:

```
value=sprint("%S%4d%5.2f\n", $1, $2, $3)
```

Different Patterns available in awk:

- The pattern identifies on which records in a file, the action is to be taken.
- If the pattern matches the record, its associated action is taken.
- If the pattern does not match the record, its associated action is skipped.
- If an action is specified without a pattern then it is always taken.
- The awk supports different types of patterns



Variables in awk:

- Like all programming languages, awk also permits the use of variables
- In awk, two types of variables are available
  - i) User-defined variables
  - ii) Built-in variables
- The Built-in variables of awk are,

Variable	*Meaning*
FILENAME	Name of the current input file.
FS	Input field separator (default: blank and tab).
NF	Number of fields in input record
NR	Number of current record
OFS	Output field separator (default: blank or tab).
ORS	Output record separator (default: new line).
RS	Input record separator (default: new line).
ARGC	Number of command line arguments.
ARGV	Command line arguments array.

Operators in awk:

i) Arithmetic operators + - * / %
ii) Logical operators || && !
iii) Relational operators > >= < <= == !=
iv) Assignment operators = += -= *= /= %=
v) Match operators ~ !~
vi) Increment and Decrement operators ++ --
vii) Conditional operator ?:

Expressions in awk:

- Regular Expressions
- Arithmetic Expressions
- Relational Expressions
- Logical Expressions

awk decision-making control structures:

i) if-else structure
ii) Nested if structure
iii) case structure

awk loop statements:

i) while
ii) do-while
iii) for

awk unconditional control statements:

i) break
ii) continue

Associative Arrays in awk:

- An array is a collection of variables of same type.
- The awk supports only one dimensional arrays called associative arrays.
- These arrays are called associative arrays because, each array index is associated with the array element.

Creating an Array:

- An associative array is created, when a variable is used with the array operator, brackets ([ ]), for the first time.
- It also initializes the array elements to zero.
- There is no need to declare them explicitly.
- The array elements are referred through their index values.
- The index can be anything, an integer or even the string.
- Infact, the integer indexes are stored as strings.

Example:

employee[$2]

- Initialization of an array is given as,

Example:

employee[$2]=$4

- Delete an array element is given as,

Example:

delete employee[$2]

string functions in awk:

- The string functions in awk are,
  - length( )
  - index( )
  - substr( )
  - split( )
  - sub( )

length( ):

- The length function counts the number of characters in a string and returns the count value.

Syntax:

length(str)

index( ):

- The index function determines the first position of a sub-string with in a string. If the string not found it returns 0.

Syntax:

index(str, substr)

substr( ):

- The substr( ) returns the substring from a given string.

Syntax:

substr(str, starting_index)

split( ):

- The split function divides a string into two substrings using a field separator.

Syntax:

split(str, array)

sub( ):

- The sub function substitutes one string for another string that matches a regular expression.
- It returns 1 (true) if the string was substituted.
- It returns 0 (false) if the matching string was not found and the substitution failed.

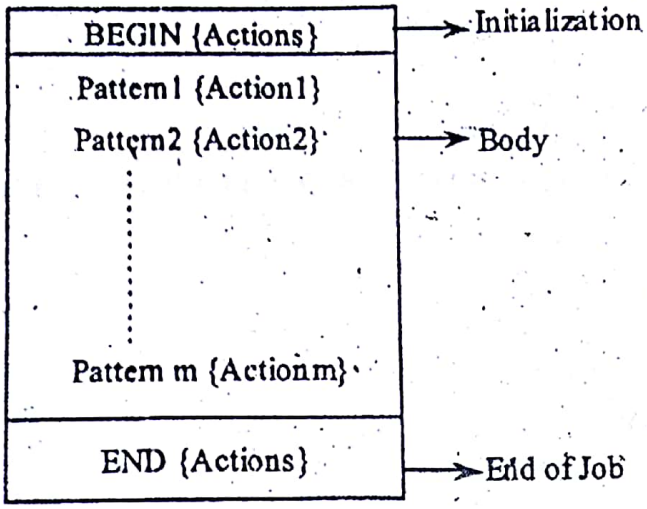
Syntax:

sub(regular_expr, withstring, inputstring)

awk script:

- An awk script is divided into three parts. They are,
  - i) Initialization
  - ii) Body
  - iii) End of a job

- An awk script is depicted as



i) Initialization:
- The initialization part defines the instructions for initializing variables, creating report headings, set system variables etc.
- This is identified with the token BEGIN and all the instructions are enclosed in curly braces.
- This part is processed only once before awk reads the first line from the input line.

ii) Body:
- The body consists of one or more instructions for processing the data in a file.
- Each instruction consists of a pattern and an action that will be taken when the pattern is matched.
- The awk applies each instruction in the body to each record in the file one after the other.

iii) End of a job:
- The end part is also executed only once after the last line from the input file has been read.
- It includes the instructions to analyze, print or perform other activities at the end of data processing.

5. Good Files and Good Filters:

- The awk programs of one or two lines length can perform filtering contributing to a larger pipeline.
- This is one of the application of awk.
- A single filter can solve certain problems sometimes, but the filters which are joined together as pipeline can solve the problem divided into sub-problems.
- Such usage of tools is assumed as heart of unix programming.
- The files that are filterable contain lines of text that does not contain decorative headers, trailers or blank lines.
- Every line is an object of interest such as filename, word, running process description.
- Therefore the wc and grep programs can search the items by name and count them.
- Even though the file contains more data about the objects, it will be in line_by_line and column format.
- Such files can be easily processed and rearranged by awk.

- All the filters have same design and write output of argument files and standard input to standard output.
- Since arguments specify only input, the command output is fed to pipeline.
- The optional arguments are provided with filenames.
- At last error messages are forwarded to standard error.
- But such conventions do effect individual commands.
- If name of the output or input file is asked then communication with parameters, headers and tailors, pipeline will not work properly.
- Pipes must be written if Unix does not provide them.