Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

B.E.

Fifth Semester Examination, May-2008

System Programming & System Administration (IT-303-E)

Note : Attempt only five questions. All questions carry equal marks.

Q. 1. (a) Discuss the role and functions of a system manager. **20**

Ans. The role and functions of a system manager are following :

(i) User Access Permissions : The system manager provide users the username and passwords. Manager also decides their access permissions.

(ii) Schema Definition : System manager is responsible for schema definition or formation of database structure in the system.

(iii) Software Updation : System manager is responsible for installation and maintenance and updation of softwares used in system.

(iv) Fault Diagnosis : It is the responsibility of the system manager to find fault in the system, if there exists a fault he should diagnose the fault.

(v) Backup : The system manager should take the backup of the system to avoid any data loss.

(vi) File Access Permission : System manager assigns various access file permission like read only, read-write, etc.

Q. 1. (b) What do you understand by control instructions ? Discuss various types of control instructions in shell. **20**

Ans. Control Instructions : Control instructions are the instructions the controls the sequence of execution of the instruction in a program. Various control instructions are used in programs to make a program execute according to the requirement of users. Following are some of the control instructions :

(i) Switch-Case : Switch-case is used to execute a specific block of code depending on the condition given in case statement.

Example :

```
int i,
switch(i)
{
      Case 1 :
           printf "This is case 1";
           break;
      Case 2 :
           printf "This is case 2";
           break;
     default :
           printf "This is default case";
}
Or char a;
switch (a)
```

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

```
{
    Case 'A' :
        printf "Alphabet A";
        break;
    Case 'B' :
        printf "Alphabet B";
        break;
    default :
        printf "Wrong Choice";
}
```

(ii) Goto : Goto is used to transfer the control of program to a specific block of instructions in a program.

Example :

```
C :
{
        printf "Hello India"
}
        printf "India is a Great Country";
        goto C;
```

(iii) If-Else : If-else statements or instructions are used to execute either of the instructions block.

Example :

```
int i = =5;
if (i > 5)
{
        printf "i is greater than 5";
}
else
        printf "i is less than or equal to 5";
```

Q. 2. How would you perform the following task under unix environment : **20**

(a) Sorting file

(b) Searching for pattern

(c) Printing files

(d) Comparing files

Ans. (a) Sorting File : In unix sorting of file is performed using sort command.

Syntax :

```
$ sort filename
```

by default the sorting is done in ascending order.

By default, sort recorder lines in ASCII collating sequence-whitespace first, then numerals, uppercase letters and finally lowercase letters. This default sorting sequence can be altered by using certain options.

The various sort options are following :

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

–tchar → Uses delimeters char to identify fields.

–k n → Sorts on nth field.

–k m, n → Sorts starts on mth field and ends on nth field.

–k m, n → Starts sort on nth column of mth field.

–u → Removes repeated lines.

–n → Sorts numerically.

–r → Reverses sort order.

–f → Folds lowercase to equivalent uppercase {case-insensitive}.

–m list → Merges sorted files in list.

–c → Checks if file is sorted.

–o filename → Places output in file filename.

(b) Searching for Pattern : In unix grep command is used for searching patterns. Grep stands for Global Regular Expression Printer.

Syntax :

$ grep 'pattern' filename.

Various grep Options :

–i → Ignores case for matching.

–v → Doesn't display lines matching expression.

–n → Displays line numbers along with lines.

–c → Displays count of number of occurrences.

–l → Displays list of filenames only.

–e exp → Specifies expression exp with this option. Can use multiple times. Also used for matching expression beginning with a hypen.

–x → Matches pattern with entire line.

–f file → Takes patterns from file, one per line.

–E → Treats pattern as an extended regular expression.

–F → Matches multiple fixed strings.

(c) Printing Files : pr command is used in unix for printing files. The pr command prepares a file for printing by adding suitable headers, footers and formatted text. pr adds five lines of margin at the top and five at the bottom. The lower portion of the page does not contain anything whereas header shows date & time of last modification of file, along with the filename and page number.

Syntax :

$ pr filename

Various Pr Options :

–k → Where k is an integer, prints file in k columns.

–d → Doublespaces input, reduces clutter.

–n → Numbers lines, which help in debugging code.

–on → Offsets lines by n spaces, increases left margin of page.

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

(d) Comparing Files : In unix three different commands are used in comparing files. Following are those commands :

(i) cmp : cmp is used to compare two files byte by byte, and the location of the first mismatch is echoed on the screen. By default, cmp does not bother about possible subsequent mismatches.

Syntax : $ cmp file1 file2

The -l (list) option gives a detailed list of the byte number, and the differing bytes in octal for each character that differs in both files.

(ii) comm : comm is the command used to find out the records available in one file and not in the other, or even those common to both comm command requires two sorted files.

Syntax :

$ comm file1 file2

The output of comm command shows three columns :

First column contains unique records of file1.

Second column contains unique records of file2.

Third column contains common records of both files.

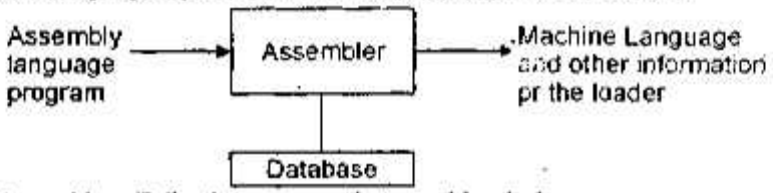
(iii) diff : diff command is also used to display file differences, it also tells the user which lines in one file have to be changed to make the two files identical.

Syntax :

$ diff file1 file2

Q. 3. What do you understand by an assembler ? Describe how assemblers are designed & explain single phase and two phase assembler. 20

Ans. Assembler : An assembler is a program that accepts an assembly language program and produces its machine language equivalent along with infomation for the loader.



Design of Assembler : Following are steps in assembler design :

(i) Statement of Problem : The first pass of assembler has only to define the symbols, the second pass can then generate the instructions and addresses specifically an assembler must do the following :

(i) Generate instructions :

(a) Evaluate the mnemonic in the operation field to produce its machine code.

(b) Evaluate the subfields-find the value of each symbol, process literals and assign addresses.

(ii) Process pseudo ops :

Pass 1 : Purpose-define symbols and literals.

Pass 2 : Purpose-generate object program.

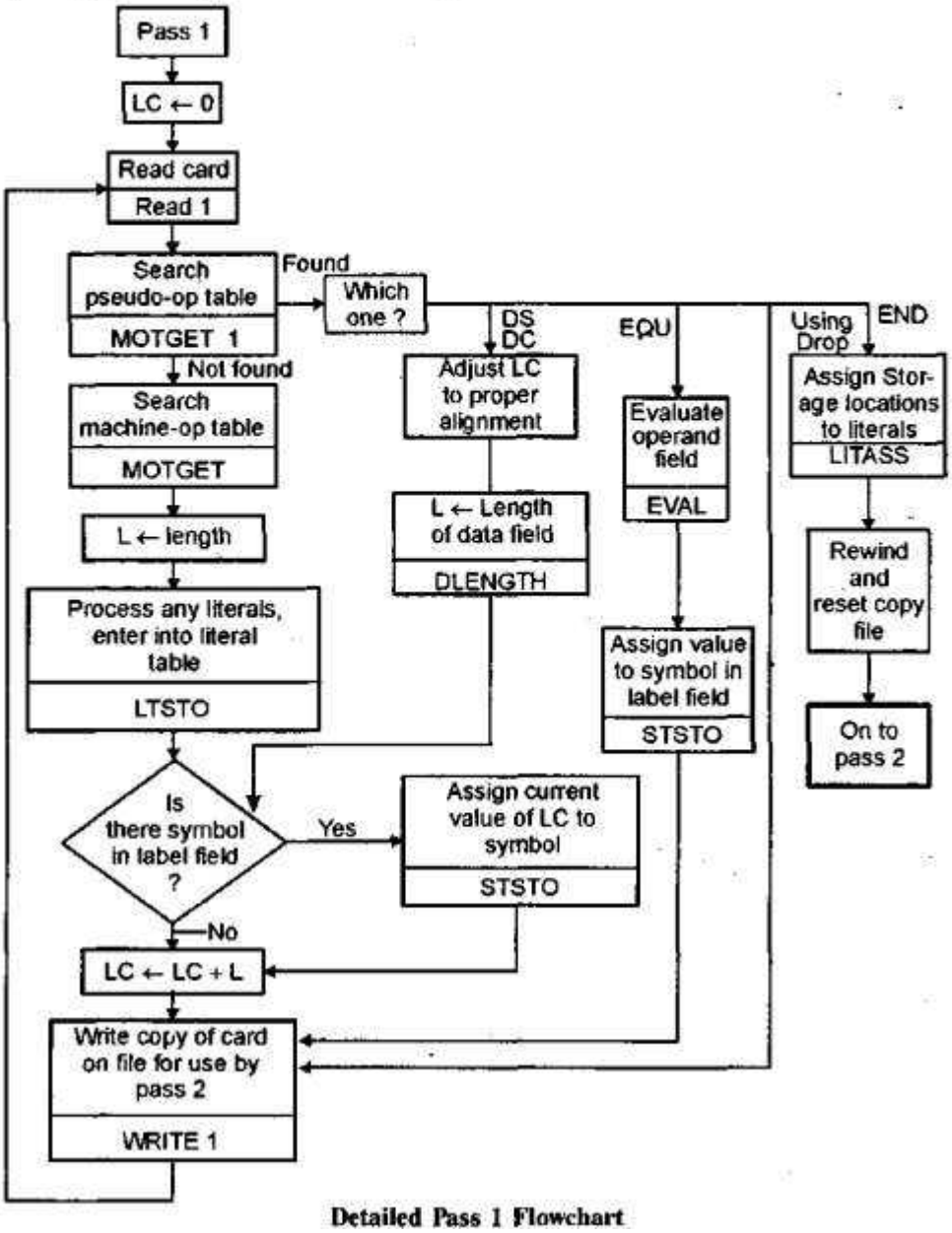
(ii) Data Structure :

Pass 1 : Data Bases :

(i) Input source program.

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

(ii) A Location Counter (LC)
(iii) A table, the Machine-Operation Table (MOT)
(iv) A table, the Pseudo-Operation Table (POT)
(v) A table, the Symbol Table (ST)
(vi) A table, the Literal Table (LT)
(vii) A copy of the input to be used later by pass 2.



Detailed Pass 1 Flowchart

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

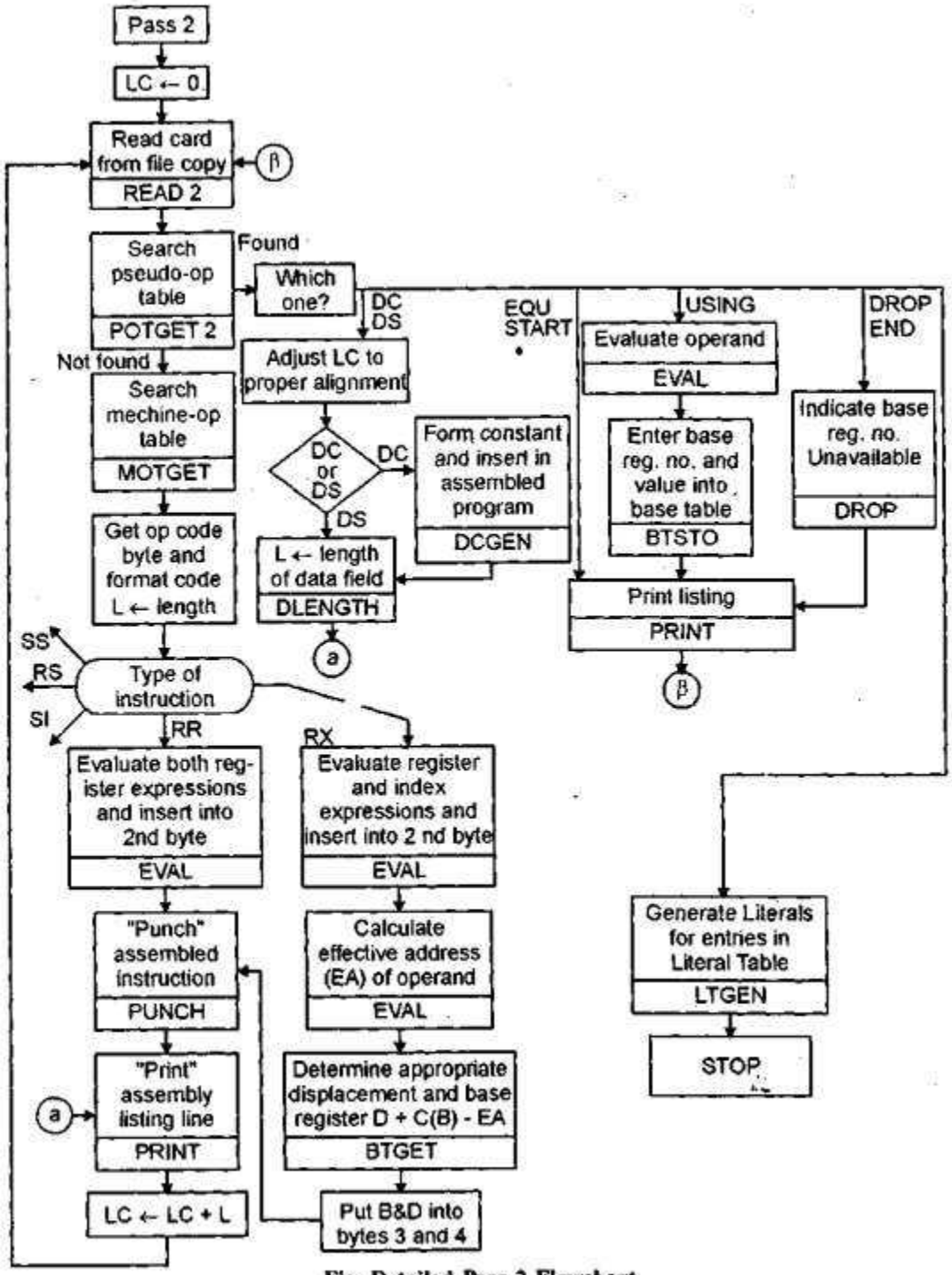


Fig. Detailed Pass 2 Flowchart

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

Pass 2 : Data Bases :

(i) Copy of source program input to pass 1

(ii) Location Counter (LC)

(iii) A table, the Machine Operation Table (MOT)

(iv) A table, Pseudo-Operation Table (POT)

(v) The Symbol Table (ST)

(vi) A table, the Base Table (BT)

(vii) A workspace, INST

(vii) A workspace, PRINT LINE.

(viii) A workspace, PUNCH CARD

(iii) Format of Databases : The third step in our design procedure is to specify the format and content of each of the database–a task that must be undertaken even before describing the specific algorithm underlying the assembler design.

(iv) Algorithm : The following flowcharts describe in detail an algorithm for an assembler. These diagrams represents a simplification of the operations performed in a complex assembler but they illustrate most of the logical processes involved.

(v) Look for Modularity : We now review our design, looking for functions that can be isolated. Typically, such functions fall into two categories :

(i) Mutli-use

(ii) Unique.

Q. 4. Explain the evolution of major components of a programming system. 20

Ans. The following are evolution of the components of a programming system :

(i) Assembler : An assembler is a program that takes as an input an assembly language program and produces an equivalent machine language program.

(ii) Loaders : A loader is a program that places programs into memory and prepares them for execution. It is the purpose of the loader to assure that object programs are placed in memory in an executable form.

(iii) Macros : If in a program a specific block of code is to be repeated again and again we use macro for that block. Macro is an abbreviation used for the specific block of code. We can call the same block by giving the macro name.

(iv) Compilers & Interpreters : Compiler is a program that accepts a program written in a high level language and produces an object program. An interpreter is a program that appears to execute a source program as if it were machine language.

(v) Formal Systems : A formal system is an uninterpreted calculus. It consists of an alphabet, a set of word called axioms and a finite set of relations called rules of inference.

Q. 5. What do you understand by macro instruction, explain the following : 20

(a) Macro instruction arguments

(b) Conditional macro expansion

Ans. Macro instructions often called macros are single-lined abbreviations for groups of instructions. In employing a macro, the programmer essentially defines a single "instruction" to represent

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

a block of code. For every occurrence of this one-line macro instruction in program, the macro processing assembler will substitute the entire block. Macro instructions are usually considered as an extension of the basic assembler language.

(a) Macro Instruction Arguments : The macro facility is capable of inserting blocks of instruction in place of macro calls. An important extension of this facility consists of providing for arguments, or parameters, in macro calls. Corresponding macro dummy arguments will appear in macro definition.

Example :

```
A          1, DATA 1
A          2, DATA 1
A          3, DATA 1
⋮             ⋮
A          1, DATA 2
A          2, DATA 2
A          3, DATA 2
                ⋮
DATA 1  DC    F' 5'
DATA 2  DC    F' 10'
```

In the above example DATA 1 & DATA 2 are called a macro instruction argument or dunny argument.

(b) Conditional Macro Expansion : Two important macro processor pseudo-ops, AIF and AGO, permit conditional reordering of the sequence of macro expansion. This allows conditional selection of the machine instructions that appear in expansions of a macro call.

AIF is a conditional branch pseudo-op, it performs an arithmetic test and branches only if the tested condition is true.

AGO is an unconditional branch pseudo-op or goto statement. It specifies a label appearing on some other statement in the macro instruction definition.

Example :

```
          MACRO
& ARG0    VARY    &COUNT, &ARG1, &ARG2, &ARG3
& ARG0      A     1, &ARG1
            AIF   (& COUNT EQ 1). FINI
            A     2, &ARG2
            AIF   (& COUNT EQ2). FINI
            A     3, & ARG3
FINI      MED D
             ⋮
LOOP 1    VAR Y   3, DATA 1, DATA 2, DATA 3
             ⋮
LOOP 2    VAR Y   2, DATA 3, DATA 2
```

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

```
LOOP 1      VAR Y     1, DATA 1
            :
DATA 1      DC        F' 5'
DATA 2      DC        F' 10'
DATA 3      DC        F' 15'
```

Q. 6. What is shell programming ? Explain. **20**

(a) Shell variables

(b) Wild cards

(c) Advanced features of shell programming

Ans. (a) Shell Variables : The shell supports variables that are useful both in the command line and ll scripts.

A variable assignment is of the form variable = value

i.e., $ count = 5

But its evaluation requires the $ as prefix to the variable name.

i.e., $ echo $ count.

A variable can also be assigned the value of another variable.

i.e., total = $ count.

Variable names begin with a letter but can contain numerals and the – as the other characters. Name case sensitive. Unlike in programming languages, shall variables are not typed; or don't need to use a ', int or long prefix when we define them. Infact, we don't even have to declare them before we can use n. All shell variables are of the string type, which means even a number like 231 is stored as a string er than in binary.

(b) Wild Cards : The metacharacters that are used to construct the generalized pattern for ching filenames belongs to a category called wild cards.

Wild Card	Matches
*	Any number of characters including none
?	A single character.
[i j k]	A single character-either an i, j or k.
[x - z]	A single character that is within the ASCII range of characters x and z.
[! i j k]	A single character that is not an i, j or k.
[! x-z]	A single character that is not within the ASCII range of characters x and z.
{pat 1, pat 2.....}	pat 1, pat 2, etc..

(c) Advanced Features of Shell Programming : Following are the advanced features of shell ramming :

(i) Interactive Scrpits : The read statement is used to make scripts interactive.

(ii) Command Line Arguments : The user can also use command line arguments in scripts.

(iii) Conditional Execution : The logical operators && and // are used for conditional execution.

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

(iv) Expression Evaluation : The user can evaluate expressions using test statement.

Q. 7. In unix o/s how the following events are handled : 2

(a) Logging in

(b) Changing user password

(c) Looking at the file contents

(d) Basic operators on files

Ans. (a) Logging in : The login prompt indicates that the terminal is available for someone to log ir This message also indicates that the previous user has logged out. Enter your username at the logi prompt and press Enter key. Now system prompts for password, enter your secrete password and you wi be logged in.

login : kumar [Enter]
Password : * * * * * [Enter]

If by mistake we enter wrong username or password the message flashes as

login incorrect
login :

(b) Changing User Password : The passwd command is used for changing user password.

Syntax : $ passwd

Example : passwd ↵

Enter login password : * * * * *
New Password : * * * * * *
Re-enter New password : * * * * * *
passwd (SYSTEM) : passwd successfully changed.

passwd expects us to respond three times. First, it prompts for the old password. Next, it chec whether we have entered a valid password, and if we have, it then prompts for new password. Enter tl new password using the password naming rules. Finally, passwd asks us to reenter the new password.

(c) Looking at File Contents : The cat command is one of the most well-known commands of tl unix system for looking' looking at file contents

Syntax : $ cat filename

cat options :

(i) Displaying Non-printing Characters (-V) : Cat is normally used for displaying text files onl Executables, when seen with cat, simply display junk. If we have non-printing ASCII characters in o input, we can use cat with -V option to display these characters.

(ii) Numbering Lines (-n) : The -n option numbers lines.

(d) Basic Operations on Files : Following are the commands and operations performed by them (files :

(i) cmp : cmp is used to compare two files.

Syntax : $ cmp file1 file2

(ii) cp : cp is used to copy a file.

Syntax : $ cp file1 file2

The file1 is copied into file2.

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

(iii) rm : rm is used to delete file.

Syntax : $ rm filename

(iv) **mv** : mv is used to rename files.

Syntax : $ mv file1 file2

Q. 8. Describe the following terms : 20

(a) Block and fragments

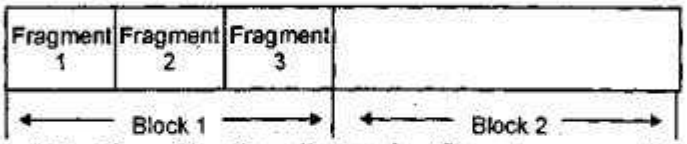
(b) Inode table

(c) AWK utility

(d) Overlays

Ans. (a) Block and Fragments : When we issue an instruction to save a file, the write operation takes place in chunks or blocks. Each block here represents an integral number of disk sectors of 512 bytes each. The data is first transferred to a buffer cache which the kernel later writes to disk.

When a block of memory is divided into small parts for specific purpose, these parts of block is known as fragments i.e., fragments are the smaller parts of a block which when combined together can result in a big block of memory.



(b) Inode Table : At the time of booting, all secondary file systems mount i.e., attach themselves to the main file system creating the illusion of a single file system to the user. Every file is associated with a table that contains all that we could possibly need to know about a file-except its name and contents. This table is called the inode (index node) and is accessed by the inode number. The inode contains the following attributes of a file :

(i) File type

(ii) File permissions

(iii) Number of links

(iv) The UID of the owner

(v) The GID of the group owner

(vi) File size in bytes

(vii) Date and time of last modification

(viii) Date and time of last access

(ix) Date and time of last change of the inode

(x) An array of pointers to keep track of all disk blocks used by the file.

(c) AWK Utility : The awk command made a late entry into the UNIX system in 1977 to augment the tool kit with suitable report formatting capabilities. Named after its authors, Aho, Weinberger and Kernighan, awk, until the advent of perl, was the most powerful utility for text manipulation. It combines features of several filters, though its report writing capability is the most useful. awk can do several things and some of them quite well. Unlike other filters, it operates at the field level and can easily access,

Downloaded from http://studentsuvidha.in and http://studentsuvidha.in/forum

transform and format individual fields in a line. It also accepts extended regular expressions (EREs) for pattern matching.

(d) Overlays : Overlays help in dynamic loading of subroutines into the main memory for execution. All the subroutines of a program are not required at the same time and loading all of them at a same time results in inefficiency. Moreover, if we have a subroutine that is larger than the size of main memory we cannot load the complete subroutine into memory and unable to execute, but overlays solves this problem by loading only the part of the subroutine that is required recently and keeping the rest of subroutine on the disk, this helps in increasing the efficiency and effective memory utilisation.