Lab manual for
Subject: System Programming

Class : sixth semester(CSE)

List of Practical.

1) Write a Program to create ,read , and write into a file having record of students.
2) a) Write a program to generate Machine Op-Code Table , Symbol Table and Pseudo Op- Code table during First Pass Assembler.
   b) Write a program to generate Machine Op- code table using Two pass Assembler.
3) a) Write a program to Generate Macro Name Table , Macro definition Table and Argument List Array during Pass One of Two Pass Macro.
   b) Write a program to generate Expanded Source in Pass Two of Two Pass Macro.
4) Study of Lexical Analyzer.
5) Study of Device Drivers.
6) Study and implement general purpose commands in UNIX . (Date , Who , Who am I, Cal, Echo , Clear, Mesg, Mail, and Login Command)
7) To Study and implement all the directory oriented Commands of UNIX(Cd , MKdir , rmdir, And Pwd Command)
8) To Study and implement all the File oriented Commands of UNIX(ls- list files, Cat, cp, rm commands)
9) To study implement HEAD, TAIL , CUT and PASTE commands.
10) To Study Common Object File Format(COFF)

Practical I :

Aim: Write a program to create , read and write into a file having record of Students.

Theory:

Opening a File

To open a file we use fopen()

```
$filename= ‘c:\file.txt’;
$fp = fopen($filename,”r”);
```

The fopen() function normally uses two parameters. The first is the file which we want to open. This can be a file on your system. In this example we are opening a text file on a windows machine . The second parameter is the “mode”. We can open a file for only reading. We can open a file for only writing . We can open a file for reading and writing . In the above example ”r” means reading.

Reading Info from a File:

After we have opened the file , we will probably want to do something with it. To read the contents of a file into a variable we use fread().

```
$contents= fread( $fp, filesize($filename));
fclose($fp);
```

Now fread() also uses two parameters. The first parameter is the variable to which we assigned the fopen() function, the second is the number of bytes we want to read up to in the file. In this case we want to read the entire file, so we use the filesize() function to get the size of the file specified in the $filename variable. Thus, it reads the entire file in.

Let us assume C: file.txt contains:

Line1
Line2

Line3
Line4
Line5

So now $contents would contain:
$contents= “line1\n line2\ n line3\n line4\n line5”;
If we printed the variable out the output would be:

Line1
Line2
Line3
Line4
Line5

The contents of the file are read into a string , complete with newline characters(\n. We can then process this string however we like.

Write To a File:

We can also write data into a file once we have opened it. To do this we use the fputs() function.

```
$filename = ‘C: \file .txt ‘;
$fp = fopen( $filename, “a”);
$string=”\nline5”;
$write=fputs($fp,$string);
fclose($fp);
```

Firstly we open the file. Notice the “a” parameter ? That means “open the file for writing only, and place the file pointer at the end of the file”.

We then use fputs() to write to the file. We then close the file. So now what will the file contain/

Line1
Line2
Line3
Line4
Line5

Modes:

There are various modes you can use to open a file, they are listed on the php.net fopen() function page, but I'll stick them up on here too.

- 'r' – Open for reading only, place the file pointer at the beginning of the file.
- 'r+'- Open for reading and Writing ; place the file pointer at the beginning of the file.
- 'w'- Open for writing only, place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist , attempt to create it.
- 'a'- Open for writing only, place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+'- Open for reading and writing ; place the file pointer at the end of the file. If the file does not exist , attempt to create it.

Viva Voice:

1) What is the use of fseek() function?
2) What is the difference between getw() and getc() function.
3) What is the use of ftell() function?
4) How we are creating file?
5) What is the difference between fscanf() and fread()?
6) What is the difference between fprintf() and fwrite()?
7) What are the different techniques of file handing?

Practical II:

Aim:

a) Write a Program to Generate Machine OP-Code table , Symbol table and Pseudo- Op-code table during First Pass of Two Pass Assembler.
b) Write a program to generate Machine Code during Pass Two of Two Pass Assembler.

Theory:

Assembler translates a program written in assembly language and generates a sequence of machine instructions. In an assembly language program, generally there is one statement per line, which can be one of two types.

1) Mnemonic machine instructions – each one is translated to a single executable instruction.
2) Assembly directives, which control assembly processing . There are many types of directives;
3) Location counter directives, which modify location of next instruction.

Assemblers produce object code modules, symbol tables , and assembly listing , giving translation of each statement . A typical listing may contain.

Line#	Label	Mnemonic	Operands	Comment	Location	Object Code
12	A	ADD	#2, Reg1	ADD2	10002E	1200CF0F

Object code denotes the code to be loaded at given memory location . It can denote either an executable machine instruction or some data to be stored at that location.

Logic of Two pass Assemblers:

Pass-1(Gather Symbols)

Assign address to each statement as per the size of each instruction and addressing modes.
Save values of labels to be used in Pass-II
Process assembler directives , Which affects address assignment .

Write output to an intermediate file.

Pass –II (Assemble instruction)

Assemble instructions as per opcode table and symbol values
Generate data value for storage directives
Perform assembly directives not done during pass –I
Write object code and Symbol tables to a file and assembly listing to output device.

Assembler skips all comments and detects errors in pass-I

Viva Voice:

1) What is the purpose of pass1 of assembler?
2) What is the purpose of pass2 of assembler?
3) What are the functions of POTGET, MOTGET,LTSTO, BTSTO,LITASS?
4) What is the difference between MOT of pass1 &pass2?
5) What are the examples of fixed tables? Why are they called fixed tables?
6) What are the examples of variable tables? What is the purpose of variable tables?
7) What are various operations before transferring control to pass2?

Practical III

Aim:

a) Write a program to generate Macro Name Table, Macro Definition table and argument list array in Pass One of Two Pass Macros.
b) Write a program to generate expanded Source in Pass Two of Two Pass Macros.

Theory:

The assembly language programmer often finds it necessary to repeat some blocks f code many times during programming. The block may consist of code to save or exchange sets of registers or code to set up linkages or perform a series of arithmetic operations. Thus Macro facility is used here, Macro instruction (macros) are single line abbreviations for groups of instructions.

In employing a macro , the programmer defines a single “instruction “ to represent a block of code . For every occurrence of this one line macro instruction in his program , the macro processing assembler will substitute the entire block.

Implementation :

Statement of Problem:

1. Recognize macro definitions.
2. Save the definitions
3. Recognize calls
4. Expand calls and substitute arguments.

Specification of Databases

Pass I data bases

1. The input macro source deck.
2. The output macro source desk copy for use by pass2
3. The Macro definition Table , used to store the body of macro definitions
4. The Macro Name Table, used to store the names of defined macro

5. The Macro Definition Table Counter , used to indicate the next available entry in the MDT
6. The Macro Name Table counter , used to indicate the next available entry in MNT
7. The argument List array , used to substitute index markers for dummy arguments before storing a macro definition.

Pass II database:

1. The copy of the input macro source deck.
2. The output expanded source deck to be used as input to the assembler.
3. The MDT created by pass I
4. The MNT created by pass II
5. The MDTP used to indicate the next line of text to be used during macro expansion.
6. The ALA , used to substitute macro call arguments for the index markers in the stored macro definition.

Viva Voice:

1) What are the tasks performed by macro processor?
2) What is the purpose of pass1 macro processor?
3) What is the purpose of pass2 macro processor?
4) What is the purpose of MDT table?
5) What is the purpose of MNT table?
6) What is the use of MDI & MDLC?

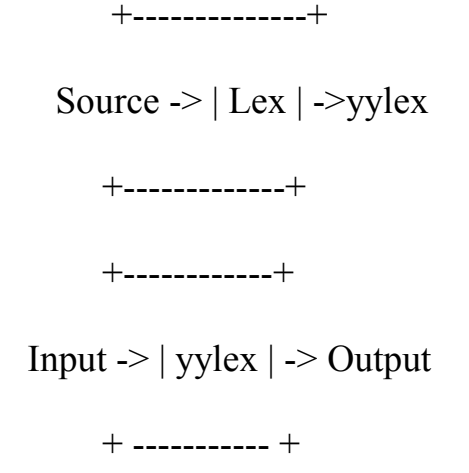
Practical IV:

Aim: Study of Lex- A Lexical Analyzer Generator.

Theory:

Lex is a program generator designed for lexical processing of character input streams. It accepts a high level, problem oriented specification for character string matching and produces a program in a general purpose language which recognizes regular expressions. The user in the source specifications given to lex specifies the regular expressions. The Lex written code recognizes these expressions in an input stream and partitions the input stream into strings matching the expressions. At the boundaries between strings program sections provided by the user are executed. The Lex source file associates the regular expressions and the program fragments . As each expression appears in the input to the program written by lex , the corresponding fragment is excuted.

Lex turns the user's expressions and actions (called source in this memo) into the host general purpose language; the generated program is named yylex. The yylex program will recognize expressions in a stream (called input in this memo) and perform the specified actions for each expression as it is detected.



An Overview of Lex

Figure 1

Working of LEX :

Summary of Source Format.

The general form of a Lex source file is:

{ definitions}
%%
{ rules}
%%
{ user subroutines}

The definitions section contains a combination of

1) Definitions, in the form " name space translation "
2) Included code, in the form " space code"
3) Include code , in the form

```
%{
    code

%}
```

4) Start conditions, given in the form
   %S name1 name2……..
5) Character set tables , in the form
   %T
   number space character-string
   ……
   %T
6) Changes to internal array sizes, in the form
   %x nnn

Where nnn is a decimal integer representing an array size and x selects the parameter as follows.

Letter	Parameter
P	Positions
n	States
e	tree nodes
a	transitions
k	packed character classes
o	output array size

Lines in the rules section have the form " expression action " where the action may be continued on succeeding lines by using braces to delimit it.

Regular expression in Lex use the following operators.

x the character "x"
"x" an "x" , even if x is an operator
\x an "x", even if x is an operator.
[xy] the character x or y
[x-z] the characters x, y or z
[^x] any character but x.
. any character but newline
^x an x at the beginning of a line.
<y>x an x when lex is in start condition y.
x$ an x at the end of a line.
x? an optional x.
x* 0,1,2,……..instances of x.
x+ 1,2,3,…….. instances of x.
x | y an x or a y.
(x) an x
x / y an x but only if followed by y
{ xx } the translation of xx from the definitions section.
x{ m ,n } m through n occurrences of x.

Viva Voice:

1) What is the role of lexical analyzer?
2) Explain lexical analysis.
3) What are the tokens, patterns , lexemes?
4) How can you recognize a token?
5) What are the various issues in lexical analysis?
6) Explain 3 parts of the lex program.
7) What are the Lex specifications?
8) What are the rules used to define the lexical analysis function?
9) What is the regular expression?

Practical V:

Aim: Study of Device drivers.

Theory:

Introduction:

The purpose of a device driver is to handle requests made by the kernel with regard to a particular type of device. There is a well defined and consistent interface for the kernel to make these requests. By isolating device specific code in device drivers and by having a consistent interface to the kernel, adding a new device is easier.

A device driver is a software module that resides within the Digital Unix kernel and is the software interface to a hardware device or devices. A hardware device is a peripheral, such as a disk controller , tape controller , or network controller device. In general , there is a one device driver for each type of hardware device. Device drivers can be classified as:

1) Block Device Drivers
2) Character Device Drivers(including terminal drivers)
3) Network Device Drivers
4) Pseudodevice drivers.

1. Block Device Driver

A block device driver is a driver that performs I/O by using file system block sized buffers from a buffer cache supplied by the kernel. The kernel also provides for the device driver support interfaces that copy data between the buffer cache and the address space of a process.

Block device drivers are particularly well- suited for disk drives, the most common block devices. For block devices , all I/O occurs through the buffer cache.

2. Character Device Driver:

A character device driver does not handle I/O through the buffer cache, so it is not tied to a single approach for handling I/O. You can use a character

device driver for a device such as a line printer that handles one character at a time. However , character drivers are not limited to performing I/O one character at a time. (despite the name " character" driver) . For example , tape drivers frequently perform I/O in 10 K chunks. You can also use a character device driver when it is necessary to copy data directly to or from a user process. Because of their flexibility in handling I/O , many drivers are character drivers . Line printers, interactive terminals, and graphics displays are examples of devices that require character device drivers.

A terminal device driver is actually a character device driver that handles I/O character processing for a variety of terminal devices. Like any character device, a terminal device can accept or supply a stream of data based on a request from a user process. It cannot be mounted as a file system and therefore does not use data caching.

3. Network Device Driver

A network Device driver attaches a network subsystem to a network interface , prepares the network interface for operation, and governs the transmission and reception of network frames over the network interface.

4. Psedodevice Driver

Not all device drivers control physical hardware . Such device drivers are called "peudodevice" drivers. Like block and character device drivers , pseudodevice drivers make use of the device driver interfaces. Unlike block and character device drivers, pseudodevice drivers do not operate on a bus. One example of a pseudodevice driver is the pseudoterminal or pty terminal driver , which simulates a terminal device . The pty terminal driver is a character device driver typically used for remote logins.

Viva Voice:

1) What is a device driver?
2) What is the difference between character driver and block driver?
3) What are the header files included for line printer driver?
4) What are the major design issues for designing a RAM derive?
5) How do device drivers handle interrupts?

Practical VI:

Aim : To study and implement the general purpose commands in UNIX . (Date , Who , Who am I, Cal , Echo , Clear , Mesg , Mail and LOGIN command)

Introduction:

Commands:

Login

Syntax: $login[username]
No Options
Purpose: Sign on to another account

Passwd

Syntax: $passwd [username]
No Options
Purpose: Change Login password

Man

Syntax: $man[options][section]entry…..
Options:- e option gives you a one line introduction to the command . Entry| more gives the pagewise introduction to the command
Purpose: Display On – Line UNIX user's manual.

Date

Syntax: $date[username]
No Options
Purpose: Sign on to another account

Mesg

Syntax: $mesg[option]
Options: none- Report your terminal write permission

N Remove write permission for your terminal
Y Grant write permission for your terminal

Purpose: Permit or deny messages.

Mail

Syntax: $mail[option]
$mail username…….
Options
-p Display entire contents of mailfile without prompting for instructions, then mail exits.
-q Cause mail to exit after receiving an interrupt signal without changing the mailfile
-r Reverse order in which mail displays messages.
$mail username…
Type your text here….

$mail
From username fri sep 7 12:40 PDT 1984
Your text display here……..

??
q quit
x exit without changing mail.
P print
S[file] save (default mbox)
d delete
Purpose: Send or Receive electronic mail among users

Echo

Syntax: $echo[options][arg…….]
Options:- n Do not terminate output with a new line character
Purpose: Display the contents of command

tty

Syntax : $tty
No Options
Purpose: Get your terminal name

Date

Syntax: $date
$date[+format]

Format
The formatting directives for the date include

%D The date as MM/DD/YY
%a Abbreviated weekday(sun to sat)
%h Abbreviated month (jan to dec)
%j day of the year (001 to 365, or 366 on leap years)
%w the day if the week(Sunday=0 , Monday=1 , and so on)
%m The month of the year(01 to 12)
%d The day of the month (01 to 31)
%y Last two digits of the year(00 to 99)

The formatting directives for the time include
%T The time as HH:MM:SS
%r The time as HH:MM:SS(A.M./P.M)
%H The hour (00 to 23)
%M The minute (00 to 59)
%S The second(00 to 59)

Purpose: display the current date and time

Cal

Syntax: $cal[month-number]year
No options

Purpose: Print calendar

Who

Syntax: $who

Purpose: Determine who is on the system

Who am I

Syntax: $who am I
Purpose: Display the name under which you are logged in.

Viva Voice:

1) What do you mean by kernel?
2) How does the file represent kernel?
3) List features of UNIX.
4) What is the significance of mail and merge command?
5) What does Multi-user technology mean?

Practical VII:

Aim: To study and implement all the directory oriented Commands of UNIX(Cd, Mkdir , rmdir , and Pwd Command)

Introduction:

Commands:

Cd

Syntax: $cd
$cd pathname
No options

Purpose: Change working directory

Mkdir

Syntax: $mkdir pathname…….
No options

Purpose: Make a directory.

Rmdir

Syntax: $rmdir pathname…
No options

Purpose: Remove a directory

Pwd

Syntax: %pwd
No options

Purpose: Print working directory.

Viva Voice:

1) Differentiate between Ordinary files , directory files and device files.
2) What does directory file contain?
3) How to rename a directory?
4) What is a regular file?
5) How is the password of a particular Id Changed?

Practical VIII:

Aim: To study and implement all the file oriented Commands of UNIX (ls- list files, Cat, cp, rm commands)

Introduction:

Commands:

ls

Syntax: $ls [option]
Option
-x : output in multiple columns
-Fx Identifying directories and executable files.
-axF: Showing hidden files
-r: Reversing the sort order
-R Recursive listing
Purpose: Listing files

cat

Syntax: $cat filename(with extention)
$cat >filename
Type the contents of file here…….
This is used for creating a file.

$cat filename
This is used for displaying the contents of a file.

Purpose: Displaying & creating files.

cp

Syntax: $cp[option] source file destination file
Option:
-I It warns the user before overwriting
-r It copies all files and sub directories from source to destination file.

Purpose: Copying a file

rm

Syntax: $rm [option] source file destination file
Option:
-I It asks the user for confirmation before removing a file
-r It removes all files & subdirectories.

Purpose: Deleting files

mv

Syntax: $mv source file destination file
No options
Purpose: Renaming a file (or directory) and moving a group of files to a different directory.

Result:

Viva Voice:

1) Explain the following options to ls
   -X,-F,-r,-a,-R,-l,-d,-t,-u,-I

2) What are the general commands used with files?
3) How is a duplicate file created?
4) How are the files under directories and subdirectories accessed?
5) What will cat foo foo foo display?
6) What will rm –f1* do?
7) How will you copy a directory structure bar1 to bar2?
8) What is the command used to terminate a current operation?

Practical IX:

Aim: To study and implement HEAD , TAIL , CUT, and PASTE commands

Introduction:

Commands:

Head

Syntax: $head [-linecount]filename
No options
Purpose: displaying the beginning of a file.

tail

Syntax: $tail [-linecount] filename
No options
Purpose: Displaying the end of a file.

cat

Synatx: $cat [option] filename
Option:
-c: It is used to exact specific columns from file .
Purpose: Slitting a file vertically

paste

Syntax: $paste filename1 filename2
No options
Purpose: Pasting the files.

Viva Voice:

1) How are the records in a file displayed on the screen?
2) Give the functions of the cut and paste command?
3) Explain how the contents of the file are edited.?
4) How are the columns adjusted using cut command?

Practical X:

Aim: Study of Common Object File Format.

Theory:

Common Object File Format(COFF). COFF is the format of the output file produced by the assembler and the link editor.

The following are some key features of COFF:

- Applications can add system dependent information to the object file without causing access utilities to become obsolete.
- Space is provided for symbolic information used by debuggers and other applications.
- Programmers can modify the way the object file is constructed by providing directives at compile time.

The object file supports user –defined sections and contains extensive information for symbolic software testing. An object file contains:

- A File header
- Optional header information
- A table of section headers
- Data corresponding to the section headers.
- Relocation information
- Line numbers
- A symbol table
- A String table

Explanation of the various fields in object File:

Object File Format

FILE HEADER
Optional Information
Section 1 Header
…
Section n Header
Raw Data for Section 1
…..
Raw Data for Section n
Relocation Info for Sect. 1
…
Relocation Info for Sect .n
Line Numbers for Sect.1
…
Line Numbers for Sect. n
SYMBOL TABLE
STRING TABLE

Viva Voice:

1) What are the header files included in File header section?
2) Distinguish between Symbol table and String table entries.
3) Give difference between Static and Dynamic relocation
4) What are the contents of section headers?