



fsmk.org/labmanual

VI
SEMESTER
FOR CSE & ISE
UNIX SYSTEMS PROGRAMMING AND
COMPILER DESIGN LABORATORY
[10CSL68]

LAB MANUAL

VTU SYLLABUS 2010

FREE SOFTWARE MOVEMENT
KARNATAKA





Content

- 1. Introduction**
- 2. Contributors**
- 3. Foreword**
- 4. Introduction to gcc**
- 5. Syllabus**
- 6. Problems and Solutions**



About Free Software Movement Karnataka (FSMK)



Free Software Movement Karnataka (FSMK) is a nonprofit organization formed in March 2009 to spread the ideals of free software. We try to increase the understanding of the philosophy behind free software and encourage its use in educational institutions, our very own community center and other organisations.

The phrase "free and open source software" can be used to collectively describe a set of operating systems and standalone applications that are free from the clutches of large corporate organisations which produce software only for commercial purposes. Free software is often freely available and is created by a thriving community of programmers, designers and writers. The source code (i.e. the computer program) that underlies an application or an operating system is also open to the perusal of the common individual, which allows every curious tinkerer, student or otherwise, to play around with the source code. Most free software organisations welcome the general public to be part of their community and to contribute in any of the three spheres mentioned above. In the event of you not being a programmer, designer or writer, you can also become a member of the community by actively using free software applications, thereby reporting any issues that you notice, and also by spreading awareness about free software among your friends and family. At FSMK, we believe that it is unfortunate that schools, colleges and other small organisations that can use Linux and related free software for no cost, instead invest in using proprietary and commercial software, thereby spending large amounts annually on licensing and other fees, which can instead be used for the betterment of education or related services, and thereby, the betterment of society.

I want to be involved Website: <http://fsmk.org/>

Mailing list: <http://www.fsmk.org/?q=mailinglistssubscribe>

About Spoken Tutorial Project:



The Spoken Tutorial project is an initiative of National Mission on Education through ICT, Government of India, to promote IT literacy through Free and Open Source Software. The project is being developed and coordinated by IIT-Bombay and led by Dr. Kannan M. Moudgalya. The project aims at building a repository of self learning courses through video tutorials of various open source softwares. These courses are then used to organize 2 hour workshops in government organizations, NGOs, SMEs and School and Colleges in India completely free of cost for the participants. These tutorials are not only available in English but also in various regional languages for the learner to be able to learn in the language he/she is comfortable in. Currently, Spoken Tutorials are available for free software tools like Blender, GIMP, Latex, Scilab, LibreOffice Suite, Ubuntu Linux, Mozilla Firefox, Thunderbird, MySQL and also programming languages and scripts like C, C++, Python, Ruby, Perl, PHP, Java. All the spoken tutorials which are released under Creative Commons License are available for download free of cost at their website <http://spoken-tutorial.org>.

About Jnana Vikas Institute of Technology, Bidadi



Jnana Vikas Institute of Technology was established in the year 2001 by JNANA VIKAS VIDYA SANGHA with a mission to not just provide a solid educational foundation to students but to build their careers, to make them eminent personalities in the society and to make the industry doors open to them. It is approved by AICTE, New Delhi and affiliated to VTU, Belgaum. It has a residential campus with nearly 73 faculties, 28 technical and non-technical supporting staff, 27 administrative and supporting staff and 590 students and is a self-contained campus located in a beautiful green land of about 25 acres. The institute has four academic departments in various disciplines of engineering and three departments in general science with nearly 19 laboratories all together, organized in a unique pattern. There is a separate department for management discipline. The campus is located at Bidadi, in southern part of city of Bengaluru. More information about the college is provided at their website, <http://www.jvitedu.in/>



Contributors

Following is the list of all the volunteers who contributed to the making of the Lab Manual.

- Ananda Kumar H N, Faculty, A.T.M.E College of Engineering, Mysore
- Arun Chavan L, Faculty, The Oxford College of Engineering, Bangalore
- Bhavya D, Faculty, P.E.S. College of Engineering, Mandya
- Bindu Madavi K P, Faculty, The Oxford College of Engineering, Bangalore
- Byregowda B K, Faculty, Sir M. Visvesvaraya Institute of Technology, Bangalore
- Deepika, Faculty, P.E.S. College of Engineering, Mandya
- Kiran B, Faculty, A.T.M.E College of Engineering, Mysore
- Manjunatha H C, Faculty, Sir M. Visvesvaraya Institute of Technology, Bangalore
- Neeta Ann Jacob, Faculty, The Oxford College of Engineering, Bangalore
- Rajesh N, Faculty, Sir M. Visvesvaraya Institute of Technology, Bangalore
- Shashidhar S, Faculty, A.T.M.E College of Engineering, Mysore
- Shwetha M K, Faculty, P.E.S. College of Engineering, Mandya
- Abdul Nooran, Student, Vivekananda College of Engineering and Technology, Puttur
- Abhiram R., Student, P.E.S Institute of Technology, Bangalore South Campus
- Ajith K.S., Student, The Oxford College of Engineering, Bangalore
- Akshay Gudi, Student, P.E.S. College of Engineering, Mandya
- Arjun M.Y., Student, P.E.S. College of Engineering, Mandya
- Chaitra Kulkarani, Student, Government Engineering College, Hassan
- John Paul S., Student, Jnana Vikas Institute of Technology, Bidadi
- Karan Jain, Student, P.E.S Institute of Technology, Bangalore South Campus
- Kuna Sharathchandra, Student, P.E.S. College of Engineering, Mandya
- Manas J.K., Student, Dr. Ambedkar Institute of Technology, Bangalore
- Manu H., Student, P.E.S. College of Engineering, Mandya
- Meghana S., Student, Government Engineering College, Hassan
- Nagaraj, Student, Vivekananda College of Engineering and Technology, Puttur
- Nandan Hegde, Student, Vivekananda College of Engineering and Technology, Puttur
- Narmada B., Student, P.E.S Institute of Technology, Bangalore South Campus
- Nawaf Abdul, Student, P.A. College of Engineering, Mangalore
- Nitesh A. Jain, Student, B.M.S. Institute of Technology, Bangalore
- Nitin R., Student, The Oxford College of Engineering, Bangalore
- Padmavathi K., Student, B.M.S. Institute of Technology, Bangalore
- Poojitha Koneti, Student, B.M.S. Institute of Technology, Bangalore
- Rahul Kondi, Student, S.J.B. Institute of Technology, Bangalore
- Rohit G.S., Student, Dr. Ambedkar Institute of Technology, Bangalore
- Samruda, Student, Student, Government Engineering College, Hassan
- Samyama H.M., Student, Government Engineering College, Hassan
- Santosh Kumar, Student, Dr. Ambedkar Institute of Technology, Bangalore
- Shashank M C., Student, S.J.B. Institute of Technology, Bangalore
- Soheb Mohammed, Student, P.E.S Institute of Technology, Bangalore South Campus
- Indra Priyadarshini, Student, P.E.S Institute of Technology, Bangalore South Campus
- Suhas, Student, P.A. College of Engineering, Mangalore
- Vamsikrishna G., Student, P.E.S Institute of Technology, Bangalore South Campus
- Vikram S., Student, P.E.S Institute of Technology, Bangalore South Campus
- Vivek Basavraj, Student, Jnana Vikas Institute of Technology, Bidadi
- Aruna S, Core member of FSMK
- HariPrasad, Core member of FSMK
- Isham, Core member of FSMK
- Jayakumar , Core member of FSMK
- Jeeva J, Core member of FSMK
- Jickson, Core member of FSMK

- Karthik Bhat, Core member of FSMK
- Prabodh C P, Core member of FSMK
- RaghuRam N, Core member of FSMK
- Rameez Thonnakkal, Core member of FSMK
- Sarath M S, Core member of FSMK
- Shijil TV, Core member of FSMK
- Vignesh Prabhu, Core member of FSMK
- Vijay Kulkarni, Core member of FSMK
- Yajnesh, Core member of FSMK



Foreword

"Free Software is the future, future is ours" is the motto with which Free Software Movement Karnataka has been spreading Free Software to all parts of society, mainly amongst engineering college faculty and students. However our efforts were limited due to number of volunteers who could visit different colleges physically and explain what is Free Software and why colleges and students should use Free Software. Hence we were looking for ways to reach out to colleges and students in a much larger way. In the year 2013, we conducted two major Free Software camps which were attended by close to 160 students from 25 different colleges. But with more than 150 engineering colleges in Karnataka, we still wanted to find more avenues to reach out to students and take the idea of free software in a much larger way to them.

Lab Manual running on Free Software idea was initially suggested by Dr. Ganesh Aithal, Head of Department, CSE, P.A. Engineering College and Dr. Swarnajyothi L., Principal, Jnana Vikas Institute of Technology during our various interactions with them individually. FSMK took their suggestions and decided to create a lab manual which will help colleges to migrate their labs to Free Software, help faculty members to get access to good documentation on how to conduct various labs in Free Software and also help students by providing good and clear explanations of various lab programs specified by the university. We were very clear on the idea that this lab manual should be produced also from the students and faculty members of the colleges as they knew the right way to explain the problems to a large audience with varying level knowledge in the subject. FSMK promotes freedom of knowledge in all respects and hence we were also very clear that the development and release of this lab manual should under Creative Commons License so that colleges can adopt the manual and share, print, distribute it to their students and thereby helping us in spreading free software.

Based on this ideology, we decided to conduct a documentation workshop for college faculty members where they all could come together and help us produce this lab manual. As this was a first attempt for even FSMK, we decided to conduct a mock documentation workshop for one day at Indian Institute of Science, Bangalore on 12 Jan, 2014. Close to 40 participants attended it, mainly our students from various colleges and we tried documenting various labs specified by VTU. Based on this experience, we conducted a 3 day residential documentation workshop jointly organized with Jnana Vikas Institute of Technology, Bidadi at their campus from 23 January, 2014. It was attended by 16 faculty members of different colleges and 40 volunteers from FSMK. The documentation workshop was sponsored by Spoken Tutorial Project, an initiative by Government of India to promote IT literacy through Open Source software. Spoken Tutorials are very good learning material to learn about various Free Software tools and hence the videos are excellent companion to this Lab Manual. The videos themselves are released under Creative Commons license, so students can easily download them and share it with others. We would highly recommend our students to go through the Spoken Tutorials while using this Lab Manual and web links to the respective spoken tutorials are shared within the lab manual also.

Finally, we are glad that efforts and support by close to 60 people for around 3 months has led to creation of this Lab Manual. However like any Free Software project, the lab manual will go through constant improvement and we would like the faculty members and students to send us regular feedback on how we can improve the quality of the lab manual. We are also interested to extend the lab manual project to cover MCA departments and ECE departments and are looking for volunteers who can put the effort in this direction. Please contact us if you are interested to support us.



GCC- GNU C Compiler

GCC is an alternative to the Turbo C compiler. It provides a variety of features and supports many languages apart from C itself.

When you compile a program , "gcc" checks the source code for errors and creates a binary object file of that code (if no errors exist). It then calls the linker to link your code's object file with other pre-compiled object files residing in libraries. These linked object binaries are saved as your newly compiled program.

Options can be provided to gcc to dictate the way a process is performed. For example, you could tell "gcc" to just create the object file and skip the linking specially when developing large programs or building your own libraries.

Options used in GCC:

The important options commonly used in gcc are-

`-Wall -L{directory_name}`
`-l{library} -o{file_name}`

where:

{library} denotes a library file

{file_name} denotes the name of a Unix file

{directory_name} name of the directory

Example: main.c

```
#include<stdio.h>
int main(void)
{
    printf("FSMK");
    return 0;
}
```

Compilation

```
gcc main.c
```

Explanation of the options:

-Wall

This option enables all the warnings in GCC.

-O

This is to specify the output file name for the executable.

ex: **gcc main.c -o main**

-l

Tells the linker to search a standard list of directories for the library (i.e,used to link with shared libraries). The linker then uses this file as if it had been specified precisely by name.

ex: **gcc main.c -lccp**

-lm

To use the library math.h, use the -lm option during compilation.

ex: **gcc main.c -lm**

-L

Tells the linker to search standard system directories plus user specified directories.

-g

Generates additional symbolic debugging information for use with gdb debugger.

-C

Produce only the compiled code (without any linking)

ex: **gcc -C main.c**

-D

The compiler option -D can be used to define the macro MY_MACRO from command line.

ex: **gcc -DMY_MACRO main.c**

-fopenmp

This option is used to enable the different OpenMP directives (#pragma omp). This option along with -static is used to link OpenMP.

ex: **gcc main.c -fopenmp -o main**

Syntactic differences between Turbo C and GCC:

- The library conio.h is not used in GCC. Hence, getch() and other functions using conio.h do not work.
- The function clrscr() does not work.
- Since GNU/Linux environment always expects a running process to return an exit status when the process is completed, the main() function in C Programs should always return an integer instead of returning void.

Advantages of GCC:

1. GCC is free.
2. Supports multiple languages (C,C++,Java etc)
3. GCC is portable. Runs on almost all platforms.
4. Generates backend code.

Resources

- Please go through the video tutorials on C Programming and GCC developed and released by **Spoken Tutorial Project**, an initiative of National Mission on Education through ICT, Government of India, to promote IT literacy through Open Source Software. Students can go through these video tutorials to get better understanding of the subject. The tutorials can be downloaded from [here](#). More info about the project can be found [here](#)



Lab Programs list for Unix Systems Programming and Compiler Design Lab as specified by VTU for 6th Semester students:

1. Write a C/C++ POSIX compliant program to check the following limits:
 1. No. of clock ticks
 2. Max. no. of child processes
 3. Max. path length
 4. Max. no. of characters in a file name
 5. Max. no. of open files/ process
2. Write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros.
3. Consider the last 100 bytes as a region. Write a C/C++ program to check whether the region is locked or not. If the region is locked, print pid of the process which has locked. If the region is not locked, lock the region with an exclusive lock, read the last 50 bytes and unlock the region.
4. Write a C/C++ program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.
5.
 1. Write a C/C++ program that outputs the contents of its Environment list.
 2. Write a C / C++ program to emulate the unix ln command
6. Write a C/C++ program to illustrate the race condition.
7. Write a C/C++ program that creates a zombie and then calls system to execute the ps command to verify that the process is zombie.
8. Write a C/C++ program to avoid zombie process by forking twice.
9. Write a C/C++ program to implement the system function.
0. Write a C/C++ program to set up a real-time clock interval timer using the alarm API.
1. Write a C program to implement the syntax-directed definition of "if E then S1" and "if E then S1 else S2". (Refer Fig. 8.23 in the text book prescribed for 06CS62 Compiler Design, Alfred V Aho, Ravi Sethi, and Jeffrey D Ullman: Compilers- Principles, Techniques and Tools, 2nd Edition, Pearson Education, 2007).
2. Write a yacc program that accepts a regular expression as input and produce its parse tree as output.



Aim:

To Write a C/C++ POSIX compliant program to check the following limits:

- (i) No. of clock ticks
- (ii) Max. no. of child processes
- (iii) Max. path length
- (iv) Max. no. of characters in a file name
- (v) Max. no. of open files/ process

Theory:

sysconf - Get configuration information at runtime

SYNOPSIS

```
#include <unistd.h>
long sysconf(int name);
```

DESCRIPTION

POSIX allows an application to test at compile or run time whether certain options are supported, or what the value is of certain configurable constants or limits. At compile time this is done by including `<unistd.h>` and/or `<limits.h>` and testing the value of certain macros.

At run time, one can ask for numerical values using the present function `sysconf()`. One can ask for numerical values that may depend on the file system a file is in using the calls `fpathconf(3)` and `pathconf(3)`. One can ask for string values using `confstr(3)`. The values obtained from these functions are system configuration constants. They do not change during the lifetime of a process.

```
clock ticks - _SC_CLK_TCK
                The number of clock ticks per second. The corresponding variable is obsolete. It was of course called CLK_TCK.

CHILD_MAX - _SC_CHILD_MAX
                The max number of simultaneous processes per user ID. Must not be less than _POSIX_CHILD_MAX (25).

OPEN_MAX - _SC_OPEN_MAX
                The maximum number of files that a process can have open at any time. Must not be less than _POSIX_OPEN_MAX (20).
```

fpathconf, pathconf - Get configuration values for files

SYNOPSIS

```
#include <unistd.h>
```

```
long fpathconf(int fd, int name);
long pathconf(char *path, int name);
```

DESCRIPTION

fpathconf() gets a value for the configuration option name for the open file descriptor fd.

pathconf() gets a value for configuration option name for the filename path.

The corresponding macros defined in <unistd.h> are minimum values; if an application wants to take advantage of values which may change, a call to fpathconf() or pathconf() can be made, which may yield more liberal results.

_PC_PATH_MAX

returns the maximum length of a relative pathname when path or fd is the current working directory. The corresponding macro is _POSIX_PATH_MAX.

_PC_NAME_MAX

returns the maximum length of a filename in the directory path or fd that the process is allowed to create. The corresponding macro is _POSIX_NAME_MAX.

Code:

```
#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include "iostream"
#include <unistd.h>
using namespace std;
int main()
{
    cout<<"No of clock ticks:"<<sysconf(_SC_CLK_TCK)<<endl;
    cout<<"Maximum no of child processes:"<<sysconf(_SC_CHILD_MAX)<<endl;
    cout<<"Maximum path length:"<<pathconf("/",_PC_PATH_MAX)<<endl;
    cout<<"Maximum characters in a file name:"<<pathconf("/",_PC_NAME_MAX)<<endl;
    cout<<"Maximum no of open files:"<<sysconf(_SC_OPEN_MAX)<<endl;
    return 0;
}
```

Output:

Commands for execution:-

- Open a terminal.
- Change directory to the file location in the terminal.
- Run g++ usp01.c -o usp01.out in the terminal.
- If no errors, run ./usp01.out

##Screenshots:

```
nooran@nooran-Aspire-V3-571:~$ cd /home/nooran/usp-lab
nooran@nooran-Aspire-V3-571:~/usp-lab$ vi usp01.cpp
nooran@nooran-Aspire-V3-571:~/usp-lab$ g++ usp01.cpp -o usp01.out
nooran@nooran-Aspire-V3-571:~/usp-lab$ ./usp01.out
No of clock ticks:100
Maximum no of child processes:30133
Maximum path length:4096
Maximum characters in a file name:255
Maximum no of open files:1024
nooran@nooran-Aspire-V3-571:~/usp-lab$
```



##Aim : ###Write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros. **##Theory :** >POSIX allows an application to test at compile or run time whether certain options are supported, or what the value is of certain configurable constants or limits.

- **_POSIX_SOURCE**:If you define this macro, then the functionality from the POSIX.1 standard (IEEE Standard 1003.1) is available, as well as all of the ISO C facilities.
- **_POSIX_C_SOURCE**:Define this macro to a positive integer to control which POSIX functionality is made available. The greater the value of this macro, the more functionality is made available.
- **_POSIX_JOB_CONTROL**:If this symbol is defined, it indicates that the system supports job control. Otherwise, the implementation behaves as if all processes within a session belong to a single process group. See section Job Control.
- **_POSIX_SAVED_IDS**:If this symbol is defined, it indicates that the system remembers the effective user and group IDs of a process before it executes an executable file with the set-user-ID or set-group-ID bits set, and that explicitly changing the effective user or group IDs back to these values is permitted. If this option is not defined, then if a nonprivileged process changes its effective user or group ID to the real user or group ID of the process, it can't change it back again.
- **_POSIX_CHOWN_RESTRICTED**:If this option is in effect, the chown function is restricted so that the only changes permitted to nonprivileged processes is to change the group owner of a file to either be the effective group ID of the process, or one of its supplementary group IDs.
- **int _POSIX_NO_TRUNC**:If this option is in effect, file name components longer than NAME_MAX generate an ENAMETOOLONG error. Otherwise, file name components that are too long are silently truncated.
- **_POSIX_VDISABLE**:This option is only meaningful for files that are terminal devices. If it is enabled, then handling for special control characters can be disabled individually.

Code:

```
#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include "iostream"
#include<unistd.h>
using namespace std;
int main()
{
    #ifdef _POSIX_JOB_CONTROL
        cout<<"System supports POSIX job control:"<<_POSIX_JOB_CONTROL<<endl;
    #else
        cout<<"System does not support POSIX job control"<<endl;
    #endif
    #ifdef _POSIX_SAVED_IDS
        cout<<"System supports saved set UID and GID:"<<_POSIX_SAVED_IDS<<endl;
    #else
        cout<<"System does not support saved set GID and UID"<<endl;
    #endif
    #ifdef _POSIX_CHOWN_RESTRICTED
        cout<<"Chown restricted option is :"<<_POSIX_CHOWN_RESTRICTED<<endl;
    #else
        cout<<"Chown Restricted not defined"<<endl;
    #endif
    #ifdef _POSIX_NO_TRUNC
        cout<<"Truncation option is :"<<_POSIX_NO_TRUNC<<endl;
    #else
        cout<<"Truncation Option not defined"<<endl;
    #endif
    #ifdef _POSIX_VDISABLE
        cout<<"disable char for terminal files"<<_POSIX_VDISABLE<<endl;
    #else
```

```
        cout<<"char for terminal device files will not be diasbled"<<endl;  
    #endif  
    return 0;  
}
```

Output:

- Open a terminal.
- Change directory to the file location in the terminals.
- Open a file using command followed by program_name

```
vi 02_posix_configuration.cpp
```

and then enter the source code and save it.

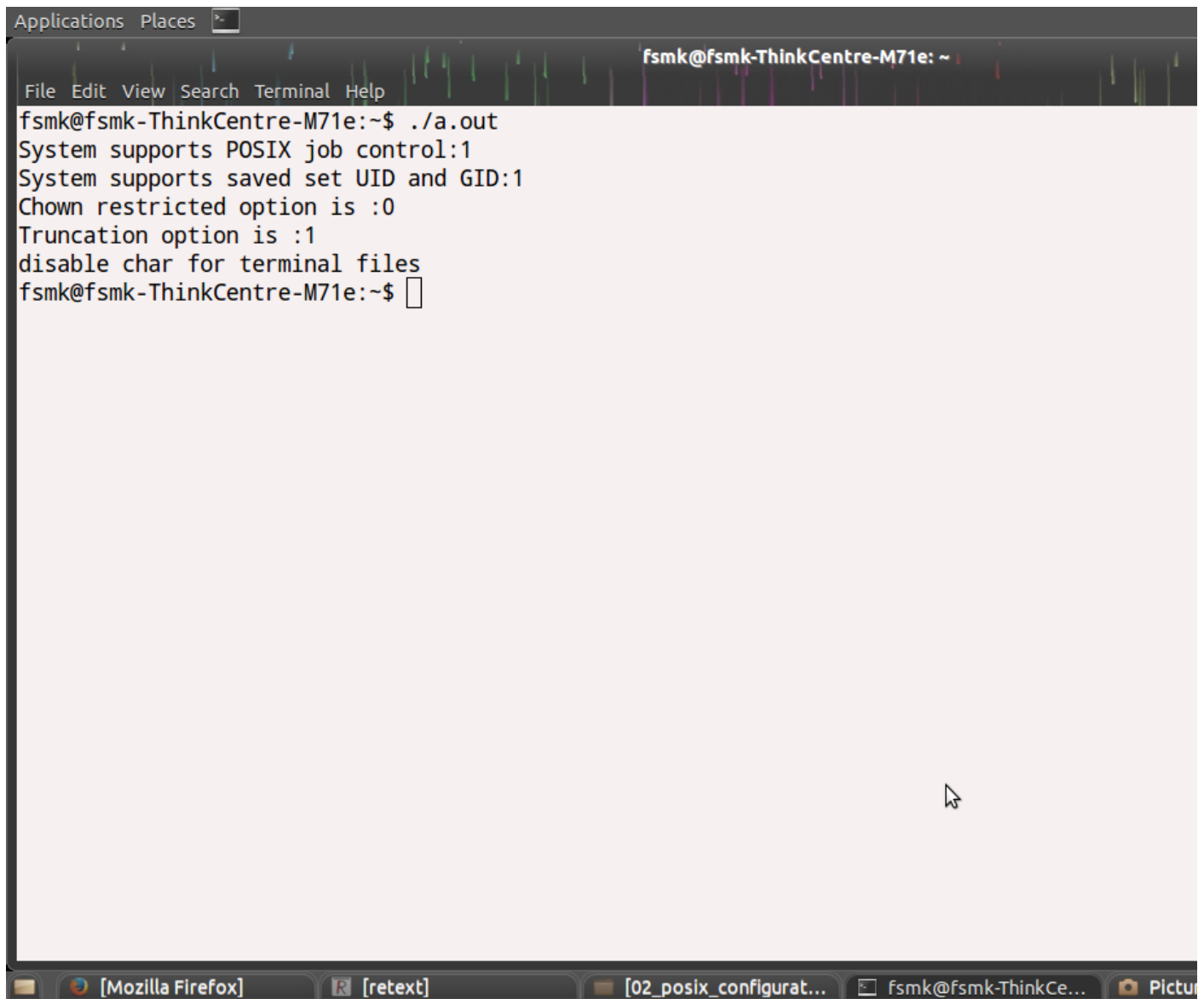
- Then compile the program using

```
g++ 02_posix_configuration.cpp
```

- If there are no errors after compilation execute the program using

```
./a.out
```

Screenshots:-



A terminal window titled "fsmk@fsmk-ThinkCentre-M71e: ~" with a menu bar containing "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows the command `./a.out` being executed, followed by several lines of output: "System supports POSIX job control:1", "System supports saved set UID and GID:1", "Chown restricted option is :0", "Truncation option is :1", and "disable char for terminal files". The prompt `fsmk@fsmk-ThinkCentre-M71e:~$` is shown at the end of the output. The terminal window is part of a desktop environment with a taskbar at the bottom showing icons for "Mozilla Firefox", "[retext]", "[02_posix_configurat...", "fsmk@fsmk-ThinkCe...", and "Pictur".

```
fsmk@fsmk-ThinkCentre-M71e:~$ ./a.out
System supports POSIX job control:1
System supports saved set UID and GID:1
Chown restricted option is :0
Truncation option is :1
disable char for terminal files
fsmk@fsmk-ThinkCentre-M71e:~$
```




Aim:

Consider the last 100 bytes as a region. Write a C/C++ program to check whether the region is locked or not. If the region is locked, print pid of the process which has locked. If the region is not locked, lock the region with an exclusive lock, read the last 50 bytes and unlock the region.

Theory:

File locking provides a very simple yet incredibly useful mechanism for coordinating file accesses.

flock - manage locks from shell scripts.

fcntl(used to manipulate the file descriptors) file commands can be used to support record locking, which permits multiple cooperating programs to prevent each other from simultaneously accessing parts of a file in error-prone ways.

fcntl() performs the operations on the open file descriptor fd.

F_GETLK, F_SETLK and F_SETLKW are used to acquire, release, and test for the existence of record locks. F_UNLK to remove the existing lock.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
int main(int argc, char *argv[])
{
    int fd;
    char buffer[255];
    struct flock fvar;
    if(argc==1)
    {
        printf("usage: ./a.out filename\n");
        return -1;
    }
    if((fd=open(argv[1], O_RDWR))==-1)
    {
        perror("open");
        exit(1);
    }
    fvar.l_type=F_WRLCK;
    fvar.l_whence=SEEK_END;
    fvar.l_start=SEEK_END-100;
    fvar.l_len=100;
    printf("press enter to set lock\n");
    getchar();
    printf("trying to get lock..\n");
    if((fcntl(fd, F_SETLK, &fvar))==-1)
    {
        fcntl(fd, F_GETLK, &fvar);
        printf("\nFile already locked by process (pid):    \t%d\n", fvar.l_pid);
        return -1;
    }
}
```

```

    }
    printf("locked\n");
    if((lseek(fd,SEEK_END-50,SEEK_END))== -1)
    {
        perror("lseek");
        exit(1);
    }
    if((read(fd,buffer,100))== -1)
    {
        perror("read");
        exit(1);
    }
    printf("data read from file..\n");
    puts(buffer);
    printf("press enter to release lock\n");
    getchar();
    fvar.l_type = F_UNLCK;
    fvar.l_whence = SEEK_SET;
    fvar.l_start = 0;
    fvar.l_len = 0;
    if((fcntl(fd,F_UNLCK,&fvar))== -1)
    {
        perror("fcntl");
        exit(0);
    }
    printf("Unlocked\n");
    close(fd);
    return 0;
}

```

Output:

Commands for instructions:

1. Compile the program
2. Run the program using ./a.out filename
3. Note: Do not stop execution
4. Open another terminal
5. Run the program using ./a.out filename
Note: Both the filename's should be same

Screenshots:

```
Applications Places
fsmk@fsmk-ThinkCentre-M71e: ~/Documents
File Edit View Search Terminal Help
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ vi 5a_envirionlist.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ gcc 5a_envirionlist.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ ./a.out
SSH_AGENT_PID=2096
GPG_AGENT_INFO=/home/fsmk/.cache/keyring-cZD46E/gpg:0:1
TERM=xterm
SHELL=/bin/bash
XDG_SESSION_COOKIE=7ff7ea0892dc049be04595930000000b-1390536177.446634-741959945
WINDOWID=46150419
GNOME_KEYRING_CONTROL=/home/fsmk/.cache/keyring-cZD46E
USER=fsmk
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:ca=30;41:tw=
;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzm=01;31:*.tlz=01;31:*.txz=01;31:*.zip
01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.d
=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=0
;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=0
35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=0
35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01
5:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35
.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:
mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.axa=00;36:*.oga=00;36:*.spx=00;36:*.xspf=00;36:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/home/fsmk/.cache/keyring-cZD46E/ssh
SESSION_MANAGER=local/fsmk-ThinkCentre-M71e:@/tmp/.ICE-unix/2062,unix/fsmk-ThinkCentre-M71e:/tmp/.ICE-unix/2062
DEFAULTS_PATH=/usr/share/gconf/gnome-fallback.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-gnome-fallback:/etc/xdg
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
DESKTOP_SESSION=gnome-fallback
```

```
Applications Places
fsmk@fsmk-ThinkCentre-M71e: ~/Documents
File Edit View Search Terminal Help
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/home/fsmk/.cache/keyring-cZD46E/ssh
SESSION_MANAGER=local/fsmk-ThinkCentre-M71e:@/tmp/.ICE-unix/2062,unix/fsmk-ThinkCentre-M71e:/tmp/.ICE-unix/2062
DEFAULTS_PATH=/usr/share/gconf/gnome-fallback.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-gnome-fallback:/etc/xdg
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
DESKTOP_SESSION=gnome-fallback
PWD=/home/fsmk/Documents
GNOME_KEYRING_PID=2051
LANG=en_US.UTF-8
MANDATORY_PATH=/usr/share/gconf/gnome-fallback.mandatory.path
GDMSESSION=gnome-fallback
SHLVL=1
HOME=/home/fsmk
LANGUAGE=en_US:en
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=fsmk
XDG_DATA_DIRS=/usr/share/gnome-fallback:/usr/share/gnome:/usr/local/share:/usr/share/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-fMDtYE3gGi,guid=b150d133115c3e7752f55f92000001d
LESSOPEN=| /usr/bin/lesspipe %s
DISPLAY=:0.0
XDG_CURRENT_DESKTOP=GNOME
LESSCLOSE=/usr/bin/lesspipe %s %s
COLORTERM=gnome-terminal
XAUTORITY=/home/fsmk/.Xauthority
OLDPWD=/home/fsmk
_=./a.out
fsmk@fsmk-ThinkCentre-M71e:~/Documents$
```

```
Applications Places fsmk@fsmk-ThinkCentre-M71e: ~/prog3
File Edit View Search Terminal Help
fsmk@fsmk-ThinkCentre-M71e:~$ cd prog3
fsmk@fsmk-ThinkCentre-M71e:~/prog3$ cc prog3.c
fsmk@fsmk-ThinkCentre-M71e:~/prog3$ ./a.out new
press enter to set lock

trying to get lock..
locked
data read from file..
nsider the 100 bytes.program to check the lock.

press enter to release lock

Unlocked
fsmk@fsmk-ThinkCentre-M71e:~/prog3$
```



Aim:

Write a C/C++ program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.

Algorithm:

Theory:

Pipes are the oldest form of UNIX System IPC and are provided by all UNIX systems. Pipes have two limitations.

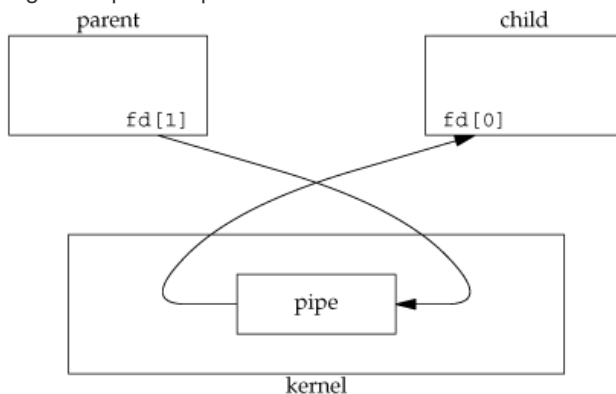
1. Historically, they have been half duplex (i.e., data flows in only one direction). Some systems now provide full-duplex pipes, but for maximum portability, we should never assume that this is the case.
2. Pipes can be used only between processes that have a common ancestor. Normally, a pipe is created by a process, that process calls fork, and the pipe is used between the parent and the child.

A pipe is created by calling the pipe function.

```
#include <unistd.h>
int pipe(int fildes[2]);
```

Returns: 0 if OK, 1 on error

Figure "Pipe from parent to child"



For a pipe from the child to the parent, the parent closes fd[1], and the child closes fd[0]. When one end of a pipe is closed, the following two rules apply.

1. If we read from a pipe whose write end has been closed, read returns 0 to indicate an end of file after all the data has been read. (Technically, we should say that this end of file is not generated until there are no more writers for the pipe. It's possible to duplicate a pipe descriptor so that multiple processes have the pipe open for writing. Normally, however, there is a single reader and a single writer for a pipe. When we get to FIFOs in the next section, we'll see that often there are multiple writers for a single FIFO.)
2. If we write to a pipe whose read end has been closed, the signal SIGPIPE is generated. If we either ignore the signal or catch it and return from the signal handler, write returns 1 with errno set to EPIPE.

When we're writing to a pipe (or FIFO), the constant PIPE_BUF specifies the kernel's pipe buffer size. A write of PIPE_BUF bytes or less will not be interleaved with the writes from other processes to the same pipe (or FIFO). But if multiple processes are writing to a pipe (or FIFO), and if we write more than PIPE_BUF bytes, the data might be interleaved with the data from the other writers. We can determine the value of PIPE_BUF by using pathconf or fpathconf.

Code:

4read.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<unistd.h>
#include<limits.h>
#include<fcntl.h>
using namespace std;
#define BUFFER_SIZE PIPE_BUF
int main(int argc,char *argv[])
{
    int pipe_fd,res=0;
    char buffer[BUFFER_SIZE+1];
    if(argc!=2)
    {
        cout<<"usage:./a.out pipe_name\n";
        return -1;
    }
    cout<<"\nFD of fifo in read mode:"<<pipe_fd<<endl;

    if((pipe_fd=open(argv[1],O_RDONLY))!=-1)
    {
        res=read(pipe_fd,buffer,BUFFER_SIZE);
        cout<<"\n data read..\n";
        cout<<buffer;
        (void) close(pipe_fd);
    }
    else
    {
        perror("\nfifo read\n");
    }

    cout<<"\nprocess "<<getpid()<<" finished reading\n"<<endl;
    return 0;
}
```

4write.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<limits.h>
#include<sys/types.h>
#include<sys/stat.h>

using namespace std;
#define BUFFER_SIZE PIPE_BUF
int main(int argc,char *argv[])
{
    int pipe_fd,res;
    char buffer[BUFFER_SIZE+1];
    if(argc!=2)
    {
```

```

        cout<<"usage:./a.out pipe_name\n";
        return 1;
    }
    if(access(argv[1],F_OK)==-1)
    {
        res=mknod(argv[1],0777);
        if(res!=0)
        {
            perror("\nmknod error\n");
            exit(0);
        }
    }

    cout<<"Process "<<getpid()<<"opening fifo in write mode"<<endl;
    pipe_fd=open(argv[1],O_WRONLY);
    cout<<"FD of fifo in write mode:"<<pipe_fd<<endl;
    if(pipe_fd!=-1)
    {
        cout<<"enter data\n";
        gets(buffer);
        res=write(pipe_fd,buffer,BUFFER_SIZE);
        if(res==-1)
        {
            perror("write error\n");
            exit(0);
        }
        close(pipe_fd);
    }
    else
        perror("fifo write");

    cout<<"\nprocess "<<getpid()<<" finished writing\n"<<endl;
    unlink(argv[1]);
    return 0;
}

```

Output:

Commands for execution:-

- Open a terminal.
- Change directory to the file location in the terminal.
- Run "g++ 4read.cpp -o 4read.out" in the terminal.
- If no errors run "g++ 4write.cpp -o 4write.out"
- If no errors, run ./4write.out pipe1"
- Open another terminal without closing the first one
- Change directory into the file location in the terminal
- Run "./4read.out pipe1"
- Shift to the original terminal and enter some string
- The same string should be displayed on the other terminal

Screenshots:

Update screenshots

```
fsmk@fsmk-ThinkCentre-M71e:~$ cd /home/fsmk/usp-lab-04/  
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-04$ vi usp-lab-04.cpp  
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-04$ g++ usp-lab-04.cpp -o usp04  
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-04$ ./usp04 pipe.txt
```

```
FD of fifo in read mode:0
```

```
data read..  
dasdasd
```

```
process 12990 finished reading
```

```
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-04$ █
```




Aim :

Write a C/C++ program that output the contents of its Environment list.

Theory :

Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer.

They are part of the operating environment in which a process runs. For example, a running process can query the value of the TEMP environment variable to discover a suitable location to store temporary files, or the HOME or USERPROFILE variable to find the directory structure owned by the user running the process.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
    int i;
    char **ptr;
    extern char **environ;
    for(ptr=environ; *ptr!=0; ptr++)
        printf("%s\n", *ptr);
    exit(0);
}
```

Output:

- Open a terminal.
- Change directory to the file location in both the terminals.
- Open a file using command followed by program_name

```
vi 5a_environlist.c
```

and then enter the source code and save it.

- Then compile the program using

```
gcc 5a_environlist.c
```

- If there are no errors after compilation execute the program using

```
./a.out
```

Screenshot :

```
Applications Places
fsmk@fsmk-ThinkCentre-M71e: ~/Documents
File Edit View Search Terminal Help
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ vi 5a_envirionlist.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ gcc 5a_envirionlist.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ ./a.out
SSH_AGENT_PID=2096
GPG_AGENT_INFO=/home/fsmk/.cache/keyring-cZD46E/gpg:0:1
TERM=xterm
SHELL=/bin/bash
XDG_SESSION_COOKIE=7ff7ea0892dc049be04595930000000b-1390536177.446634-741959945
WINDOWID=46150419
GNOME_KEYRING_CONTROL=/home/fsmk/.cache/keyring-cZD46E
USER=fsmk
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd=40;33:01:or=40;31:01:su=37;41:sg=30;43:ca=30;41:tw=
;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lzm=01;31:*.tlz=01;31:*.txz=01;31:*.zip
01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.d
=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=0
;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=0
35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=0
35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01
5:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35
.emf=01;35:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:
mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.axa=00;36:*.oga=00;36:*.spx=00;36:*.xspf=00;36:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/home/fsmk/.cache/keyring-cZD46E/ssh
SESSION_MANAGER=local/fsmk-ThinkCentre-M71e:@/tmp/.ICE-unix/2062,unix/fsmk-ThinkCentre-M71e:/tmp/.ICE-unix/2062
DEFAULTS_PATH=/usr/share/gconf/gnome-fallback.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-gnome-fallback:/etc/xdg
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
DESKTOP_SESSION=gnome-fallback

Mozilla Firefox Documents fsmk@fsmk-ThinkCe...
Applications Places
fsmk@fsmk-ThinkCentre-M71e: ~/Documents
File Edit View Search Terminal Help
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/home/fsmk/.cache/keyring-cZD46E/ssh
SESSION_MANAGER=local/fsmk-ThinkCentre-M71e:@/tmp/.ICE-unix/2062,unix/fsmk-ThinkCentre-M71e:/tmp/.ICE-unix/2062
DEFAULTS_PATH=/usr/share/gconf/gnome-fallback.default.path
XDG_CONFIG_DIRS=/etc/xdg/xdg-gnome-fallback:/etc/xdg
PATH=/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
DESKTOP_SESSION=gnome-fallback
PWD=/home/fsmk/Documents
GNOME_KEYRING_PID=2051
LANG=en_US.UTF-8
MANDATORY_PATH=/usr/share/gconf/gnome-fallback.mandatory.path
GDMSESSION=gnome-fallback
SHLVL=1
HOME=/home/fsmk
LANGUAGE=en_US:en
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME=fsmk
XDG_DATA_DIRS=/usr/share/gnome-fallback:/usr/share/gnome:/usr/local/share/:/usr/share/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-fMDtYE3gGi,guid=b150d133115c3e7752f55f920000001d
LESSOPEN=| /usr/bin/lesspipe %s
DISPLAY=:0.0
XDG_CURRENT_DESKTOP=GNOME
LESSCLOSE=/usr/bin/lesspipe %s %s
COLORTERM=gnome-terminal
XAUTHORITY=/home/fsmk/.Xauthority
OLDPWD=/home/fsmk
_=./a.out
fsmk@fsmk-ThinkCentre-M71e:~/Documents$
```



Aim :

Write a C / C++ program to emulate the unix ln command.

Theory :

Links are created by giving alternate names to the original file. The use of links allows a large file, such as a database or mailing list, to be shared by several users without making copies of that file. Not only do links save disk space, but changes made to one file are automatically reflected in all the linked files. The ln command links the file designated in the SourceFile parameter to the file designated by the TargetFile parameter or to the same file name in another directory specified by the TargetDirectory parameter. By default, the ln command creates hard links.

To create a link to a file named chap1, type the following:

```
ln -f chap1 intro
```

This links chap1 to the new name, intro. When the -f flag is used, the file name intro is created if it does not already exist. If intro does exist, the file is replaced by a link to chap1. Both the chap1 and intro file names refer to the same file.

To link a file named index to the same name in another directory named manual, type the following:

```
ln index manual
```

This links index to the new name, manual/index. To link several files to names in another directory, type the following:

```
ln chap2 jim/chap3 /home/manual
```

This links chap2 to the new name /home/manual/chap2 and jim/chap3 to /home/manual/chap3.

Code :

```
#include<stdio.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
    if(argc!=3)
    {
        printf("Usage: %s <src_file><dest_file>\n",argv[0]);
        return 0;
    }
    if(link(argv[1],argv[2])==-1)
    {
        printf("Link Error\n");
        return 1;
    }
    return 0;
}
```

Output :

- Open a terminal.
- Change directory to the file location in both the terminals.
- Open a file using command followed by program_name

```
vi 5b_unix_ln-command.c
```

and then enter the source code and save it.

- o Then compile the program using

```
g++ 5b_unix_ln-command.c
```

- o Then create a dummy file with any of the name like abc.c .
- o If there are no errors after compilation execute the program using

```
./a.out abc.c out.c
```

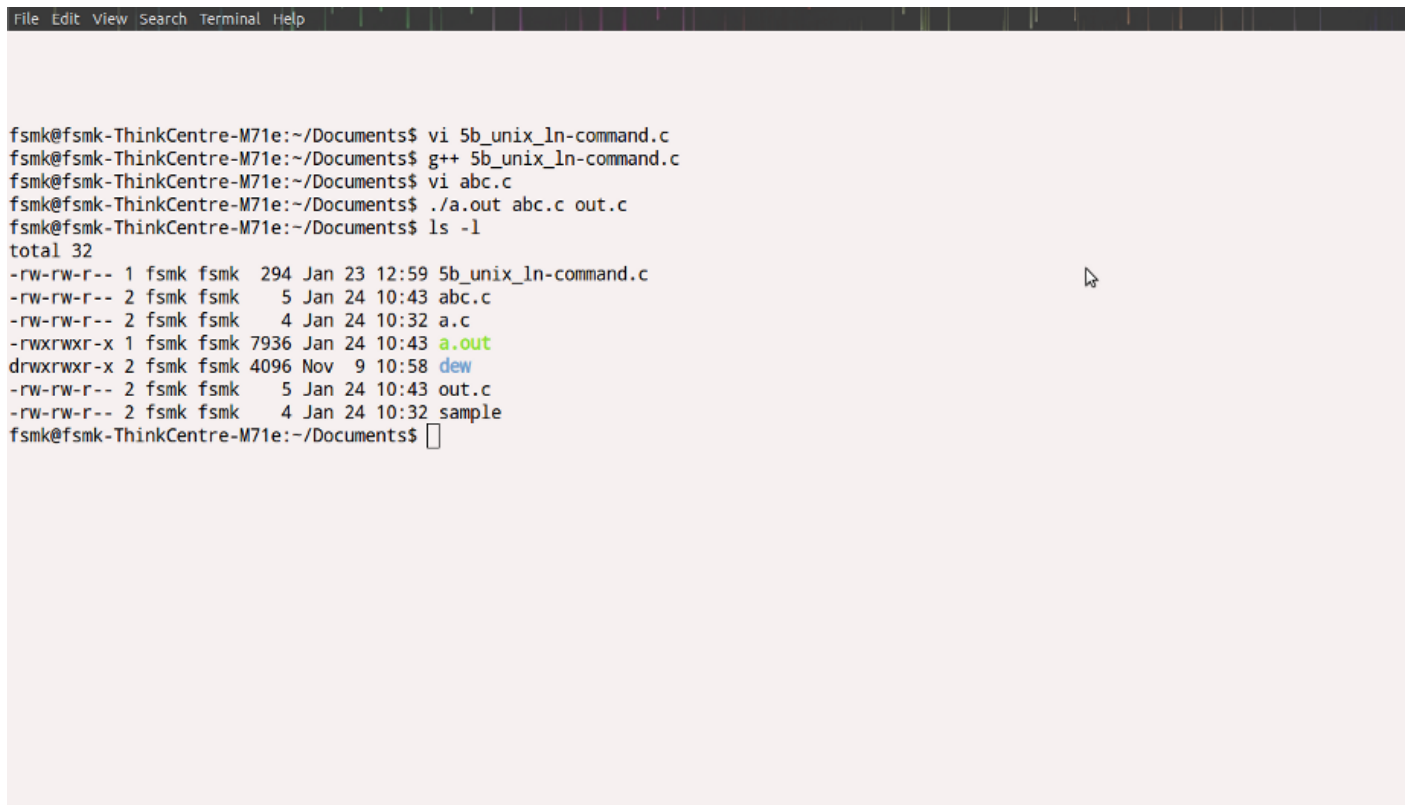
where abc.c is the source file and out.c is the new destination file to be given.

- o Then verify the creation of hard link using

```
ls -l
```

by checking the inode number of both the input files.

Screenshot :



```
File Edit View Search Terminal Help

fsmk@fsmk-ThinkCentre-M71e:~/Documents$ vi 5b_unix_ln-command.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ g++ 5b_unix_ln-command.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ vi abc.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ ./a.out abc.c out.c
fsmk@fsmk-ThinkCentre-M71e:~/Documents$ ls -l
total 32
-rw-rw-r-- 1 fsmk fsmk 294 Jan 23 12:59 5b_unix_ln-command.c
-rw-rw-r-- 2 fsmk fsmk 5 Jan 24 10:43 abc.c
-rw-rw-r-- 2 fsmk fsmk 4 Jan 24 10:32 a.c
-rwxrwxr-x 1 fsmk fsmk 7936 Jan 24 10:43 a.out
drwxrwxr-x 2 fsmk fsmk 4096 Nov 9 10:58 dew
-rw-rw-r-- 2 fsmk fsmk 5 Jan 24 10:43 out.c
-rw-rw-r-- 2 fsmk fsmk 4 Jan 24 10:32 sample
fsmk@fsmk-ThinkCentre-M71e:~/Documents$
```



Aim:

To Write a C/C++ program to illustrate the race condition.

Algorithm:

Theory:

A race condition occurs when multiple processes are trying to do something with shared data and the final outcome depends on the order in which the processes run. The fork function is a lively breeding ground for race conditions, if any of the logic after the fork either explicitly or implicitly depends on whether the parent or child runs first after the fork. In general, we cannot predict which process runs first. Even if we knew which process would run first, what happens after that process starts running depends on the system load and the kernel's scheduling algorithm.

Code:

```
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
static void charatotime(char *);
int main()
{
    int pid;
    if((pid=fork())<0)
        printf("fork error\n");
    else if(pid==0)
        charatotime("output from child\n");
    else
        charatotime("output from parent\n");
    _exit(0);
}
static void charatotime(char *str)
{
    char *ptr;
    int c;
    setbuf(stdout,NULL);
    for(ptr=str;(c=*ptr++)!=0;)
        putc(c,stdout);
}
```

Output:

Commands for execution:-

- Open a terminal.
- Change directory to the file location in the terminal.
- Run gcc usp06.c -o usp06.out in the terminal.
- If no errors, run ./usp06.out

Screenshots:

```
fsmk@fsmk-ThinkCentre-M71e:~$ cd /home/fsmk/usp-lab-06/
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ vi usp-lab-06.c
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ cc usp-lab-06.c -o usp06.out
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ ./usp06.out
output from parent
output from child
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ █
```

```
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ ./usp06.out
output from paoreuntp
ut from child
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-06$ █
```



Aim:

Write a C/C++ program that creates a zombie and then calls system to execute the ps command to verify that the process is zombie.

Theory:

In unix terminology, a process that has terminated, but whose parent has not yet waited for it, is called a zombie.

fork()

Syntax:

```
#include<unistd.h>
pid_t fork(void);
```

fork() creates a new process by duplicating the calling process. The new process, referred to as the child, is an exact duplicate of the calling process, referred to as the parent.

Sleep() : Delay for a specified amount of time.

System()

Syntax:

```
#include <stdlib.h>
int system(const char *command);
```

system() executes a command specified in command by calling /bin/sh -c command, and returns after the command has been completed.

Code:

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    int pid;
    if((pid=fork())<0)
        printf("fork error\n");
    else if(pid==0)
        _exit(0);
    sleep(2);
    system( "ps -o pid,ppid,state,ttty,command");
    _exit(0);
}
```

Output:

- Open a terminal
- Change directory to the file location in the terminal

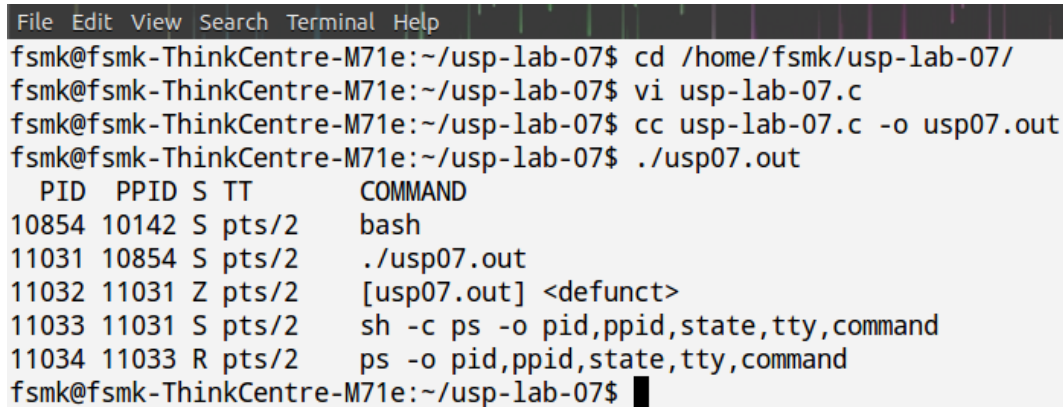
- Compile the program by using the command

```
cc usp-lab-07.c -o usp07.out
```

- Run

```
usp07.out
```

Screenshots:



A terminal window with a dark background and light text. The window title bar shows 'File Edit View Search Terminal Help'. The terminal content shows the following commands and output:

```
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-07$ cd /home/fsmk/usp-lab-07/
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-07$ vi usp-lab-07.c
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-07$ cc usp-lab-07.c -o usp07.out
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-07$ ./usp07.out
```

PID	PPID	S	TT	COMMAND
10854	10142	S	pts/2	bash
11031	10854	S	pts/2	./usp07.out
11032	11031	Z	pts/2	[usp07.out] <defunct>
11033	11031	S	pts/2	sh -c ps -o pid,ppid,state,TTY,command
11034	11033	R	pts/2	ps -o pid,ppid,state,TTY,command

```
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-07$
```




Aim:

Write a C/C++ program to avoid zombie process by forking twice.

Algorithm:

Theory:

If we want to write a process so that it forks a child but we don't want to wait for the child to complete and we don't want the child to become a zombie until we terminate, the trick is to call fork twice.

We call sleep in the second child to ensure that the first child terminates before printing the parent process ID. After a fork, either the parent or the child can continue executing; we never know which will resume execution first. If we didn't put the second child to sleep, and if it resumed execution after the fork before its parent, the parent process ID that it printed would be that of its parent, not process ID 1.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
    int pid;
    if((pid=fork())<0)
        printf("fork error\n");
    else if(pid==0)
    {
        if((pid=fork())<0)
            printf("fork error\n");
        else if(pid<0)
            _exit(0);
        sleep(2);
        printf("second child,parent pid:%d\n",getppid());
        _exit(0);
    }
    _exit(0);
}
```

Output:

Commands for execution:-

- Open a terminal.
- Change directory to the file location in the terminal.
- Run "gcc usp06.c -o usp08.out" in the terminal.
- If no errors, run "./usp08.out"

Screenshot:

File Edit View Search Terminal Help

```
fsmk@fsmk-ThinkCentre-M71e:~$ cd usp-lab-08
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-08$ vi usp-lab-08.c
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-08$ cc usp-lab-08.c -o usp08.out
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-08$ ./usp08.out
fsmk@fsmk-ThinkCentre-M71e:~/usp-lab-08$ second child,parent pid:1
█
```



Aim:

Write a C/C++ program to implement the system function.

Theory:

fork() creates a new process by duplicating the calling process. The new process, referred to as the child, is an exact duplicate of the calling process, referred to as the parent.

system() executes a command specified in command by calling /bin/sh -c command, and returns after the command has been completed. The exec() family of functions replaces the current process image with a new process image. The execl() function is one among the exec() family of functions. The waitpid() system call suspends execution of the calling process until a child specified by pid argument has changed state.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<errno.h>
#include<sys/types.h>
#include<sys/wait.h>
void sys(const char *cmdstr)
{
    int pid;
    pid=fork();
    if(pid==0)
        execl("/bin/bash", "bash", "-c", cmdstr, NULL);
    else
        waitpid(pid, NULL, 0);
}
int main(int argc, char *argv[])
{
    int i;
    for(i=1; i< argc; i++)
    {
        sys(argv[i]);
        printf("\n");
    }
    _exit(0);
}
```

Output:

- Open a terminal
- Change the present working directory to the location where the program exists using the cd command in the terminal
- Compile the program using the command `cc <program name> -o usp09.out`
- run the program using `./usp09.out`
Note: To run use `./a.out command1 command2 ,..., commandn` where each command is a shell command. Example : `./a.out ps date who`

Screenshot:

```
fsmk@fsmk-ThinkCentre-M71e:~/CS-VTU-Lab-Manual/VTU/Sem6/USP_CD_Lab$ cc 09_my_system.c
fsmk@fsmk-ThinkCentre-M71e:~/CS-VTU-Lab-Manual/VTU/Sem6/USP_CD_Lab$ ./a.out ps date who
```

PID	TTY	TIME	CMD
3185	pts/1	00:00:00	bash
3283	pts/1	00:00:00	a.out
3284	pts/1	00:00:00	ps

```
Fri Jan 24 10:12:03 IST 2014
```

fsmk	tty7	2014-01-24 09:32
fsmk	pts/0	2014-01-24 10:07 (:0.0)
fsmk	pts/1	2014-01-24 10:10 (:0.0)

```
fsmk@fsmk-ThinkCentre-M71e:~/CS-VTU-Lab-Manual/VTU/Sem6/USP_CD_Lab$ █
```



Aim :

Write a C/C++ program to set up a real-time clock interval timer using the alarm API.

Theory :

- First, every signal has a name. These names all begin with the three characters SIG .For example,SIGABRT is the abort signal that is generated when a process calls the abort function.
- SIGALRM is the alarm signal that is generated when the timer set by the alarm function goes off.
- Use the alarm API for generating a signal after certain time interval as specified by the user.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<signal.h>
#define INTERVAL 5
void callme(int sig_no)
{
    alarm(INTERVAL);
    printf("Hello!!\n");
}
int main()
{
    struct sigaction action;
    action.sa_handler=(void (*)(int))callme;
    sigaction(SIGALRM,&action,0);
    alarm(2);
    sleep(5);
    return 0;
}
```

Output :

1. Open a terminal.
2. Change directory to the file location in both the terminals.
3. Open a file using command followed by program_name

```
vi 10_alarm_signal_handler.cpp
```

and then enter the source code and save it.

4. Then compile the program using

```
g++ 10_alarm_signal_handler.cpp
```

5. If there are no errors after compilation execute the program using

```
./a.out
```

Screenshot:-

```
fsmk@fsmk-ThinkCentre-M71e:/home$ ./a.out  
Hello!!  
fsmk@fsmk-ThinkCentre-M71e:/home$
```



Aim:

Write a C program for a syntax directed definition of a "if E then S1" and "if E then S1 else S2"

Description:

A SYNTAX-DIRECTED DEFINITION is a context-free grammar in which each grammar symbol X is associated with two finite sets of values: the synthesized attributes of X and the inherited attributes of X , each production $A \rightarrow \alpha$ is associated with a finite set of expressions of the form

$$b : = f(c_1, \dots, c_k)$$

called semantic rules where f is a function and either b is a synthesized attribute of A and the values c_1, \dots, c_k are attributes of the grammar symbols of α or A , or b is an inherited attribute of a grammar symbol of α and the values c_1, \dots, c_k are attributes of the grammar symbols of α or A . Each terminal symbol has no inherited attributes.

It is usual to denote the attributes of a grammar symbol in the form $X.name$ where $name$ is a meaningful name for the attribute.

Algorithm:

1. Start
2. Output the if, if-else statement to the user for reference.
3. Manipulate the input string such that the if and if-else conditions are stored separately.
4. Generate the format of the if, if-else statements and output the same.
5. End.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char input[60],stmt[3][60];
int len,cur,i,j;
void gen()/*used for generation of if, if-else format statements*/
{
    int l1=101,l2=102,l3=103;
    printf("if %s goto %d\n",stmt[0],l1);
    printf("goto %d\n",l2);
    printf("%d:%s\n",l1,stmt[1]);
    if(cur<3)/*if statement*/
        printf("%d:STOP\n",l2);
    else/*if-else statement*/
    {
        printf("goto %d\n",l3);
        printf("%d:%s\n",l2,stmt[2]);
        printf("%d:STOP\n",l3);
    }
}
int main()
```

```

{
    printf("Format of if stmt\nExample\n");
    printf("if(a<b)then(s=a);\n");
    printf("if(a<b)then(s=a)else(s=b);\n");
    printf("enter stmt:");
    gets(input);
    len=strlen(input);
    int index=0;
    for(i=0;i<len&&input[i]!=' ';i++)
        if(input[i]=='(')
        {
            index=0;
            for(j=i;input[j-1]!=' ';j++)
                stmt[cur][index++]=input[j];
            cur++;
            i=j;
        }
    gen();
    return 0;
}

```

Output:

Commands for execution:-

- Open a terminal
- Change the directory to the file location
- Use gcc *filename.c* for compilation
- Run *./a.out* for execution

Screenshots:-

```

fsmk@fsmk-ThinkCentre-M71e: ~/usp_lab/sdd_cd
fsmk@fsmk-ThinkCentre-M71e:~/usp_lab/sdd_cd$ gcc sdd_cd.c
fsmk@fsmk-ThinkCentre-M71e:~/usp_lab/sdd_cd$ ./a.out
Format of if stmt
Example
if(a<b)then(s=a);
if(a<b)then(s=a)else(s=b);
enter stmt:if(3<5)then(a=100);
if (3<5) goto 101
goto 102
101:(a=100)
102:STOP
fsmk@fsmk-ThinkCentre-M71e:~/usp_lab/sdd_cd$

```




Aim:

Write a yacc program that accepts a regular expression as input and produce its parse tree as output.

Description:

Yacc- Yet another C Compiler defines what it is all by itself. Computer program input generally has some structure; in fact, every computer program that does input can be thought of as defining an "input language" which it accepts. An input language may be as complex as a programming language, or as simple as a sequence of numbers. Unfortunately, usual input facilities are limited, difficult to use, and often are lax about checking their inputs for validity.

Yacc provides a general tool for describing the input to a computer program. The Yacc user specifies the structures of his input, together with code to be invoked as each such structure is recognized. Yacc turns such a specification into a subroutine that handles the input process; frequently, it is convenient and appropriate to have most of the flow of control in the user's application handled by this subroutine.

Every Yacc specification file consists of three sections: the declarations, (grammar) rules, and programs. The sections are separated by double percent "%%" marks. (The percent % is generally used in Yacc specifications as an escape character.) In other words, a full specification file looks like

```
declarations
%%
rules
%%
programs
```

The declaration section may be empty. Moreover, if the programs section is omitted, the second %% mark may be omitted also; thus, the smallest legal Yacc specification is

```
%%
rules
```

The following command:

```
yacc grammar.y
```

draws yacc rules from the grammar.y file, and places the output in y.tab.c.

The following command: `yacc -d grammar.y` functions the same as example 1, but it also produces the y.tab.h file which would contain C-style #define statements for each of the tokens defined in the grammar.y file.

Algorithm:

1. Start
2. Accept an expression from the user.
3. Check for the structure that satisfies the conditions of a regular expression.
4. Print the parsed tree as output as when the conditions are satisfied.
5. End

Code:

```

%{ /*declaration part*/
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100 /*to store productions*/
int getREindex ( const char* );
signed char productions[MAX][MAX];
int count = 0 , i , j;
char temp[200] , temp2[200];
%}
%token ALPHABET
%left '|'
%left '.'
%nonassoc '*' '+'
%%/*rules section*/
S : re '\n' {
printf ( "This is the rightmost derivation--\n" );
for ( i = count - 1 ; i >= 0 ; --i ) {
    if ( i == count - 1 ) {
        printf ( "\nre => " );
        strcpy ( temp , productions[i] );
        printf ( "%s" , productions[i] );
    }
    else {
        printf ( "\n => " );
        j = getREindex ( temp );
        temp[j] = '\0';
        sprintf ( temp2 , "%s%s%s" , temp , productions[i] , (temp + j + 2) );
        printf ( "%s" , temp2 );
        strcpy ( temp , temp2 );
    }
}
printf ( "\n" );
exit ( 0 );
}
re : ALPHABET {
temp[0] = yylval; temp[1] = '\0';
strcpy ( productions[count++] , temp );/*copy the input to the prodcuton array*/
}/*only conditions defined here will be valid, this is the structure*/
| '(' re ')' /*adds the (expression) to the production array*/
{ strcpy ( productions[count++] , "(re)" ); }
| re '*'
{ strcpy ( productions[count++] , "re*" ); }
| re '+' /*adds expression+ type to the production array*/
{ strcpy ( productions[count++] , "re+" ); }
| re '|' re /*adds the expression|expression to the production array*/
{strcpy ( productions[count++] , "re | re" );}
| re '.' re/*adds the expression.expression to the production array*/
{strcpy ( productions[count++] , "re . re" );}
;
%%
int main ( int argc , char **argv )
{
/*
Parse and output the rightmost derivation,
from which we can get the parse tree
*/
    yyparse();/*calls the parser*/
    return 0;
}

```

```

yylex() /*calls lex and takes each character as input and feeds ALPHABET to check for the
structure*/
{
    signed char ch = getchar();
    yyval = ch;
    if ( isalpha ( ch ) )
        return ALPHABET;
    return ch;
}

yyerror() /*Function to alert user of invalid regular expressions*/
{
    fprintf(stderr , "Invalid Regular Expression!!\n");
    exit ( 1 );
}

int getREindex ( const char *str )
{
    int i = strlen ( str ) - 1;
    for ( ; i >= 0 ; --i ) {
        if ( str[i] == 'e' && str[i-1] == 'r' )
            return i-1;
    }
}

```

Output:-

Commands for execution

- Open a terminal
- Change your directory to the location of the file
- Run, `yacc -d filename.y`
- Run, `cc y.tab.c y.tab.h`
- An output file `a.out` is created.
- Run, `./a.out` to execute the program.

Screenshot:

```

naru@naru-R468-R418: ~
naru@naru-R468-R418:~$ ./a.out
a+[b*](b.c*)
This is the rightmost derivation--
re => re | re
=> re | (re)
=> re | (re . re)
=> re | (re . re*)
=> re | (re . c*)
=> re | (b . c*)
=> re | re | (b . c*)
=> re | re* | (b . c*)
=> re | b* | (b . c*)
=> re+ | b* | (b . c*)
=> a+ | b* | (b . c*)
naru@naru-R468-R418:~$
naru@naru-R468-R418:~$ ./a.out
a+[b*](b.c*)
This is the rightmost derivation--
re => re | re
=> re | (re)
=> re | (re . re)
=> re | (re . re*)
=> re | (re . c*)
=> re | (b . c*)
=> re | re | (b . c*)
=> re | re* | (b . c*)
=> re | b* | (b . c*)
=> re+ | b* | (b . c*)
=> a+ | b* | (b . c*)
naru@naru-R468-R418:~$

```


Tools Required



ubuntu



This book is done using free softwares only



Released under CC By
SA License