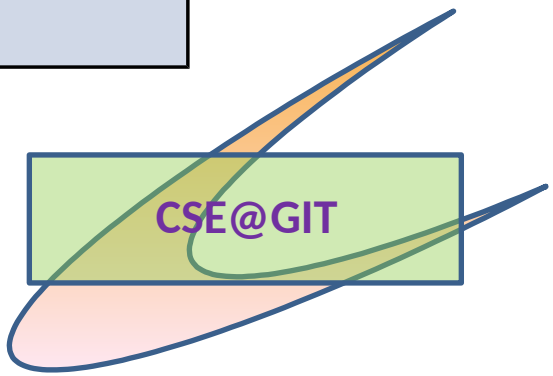
Experiment No. 7

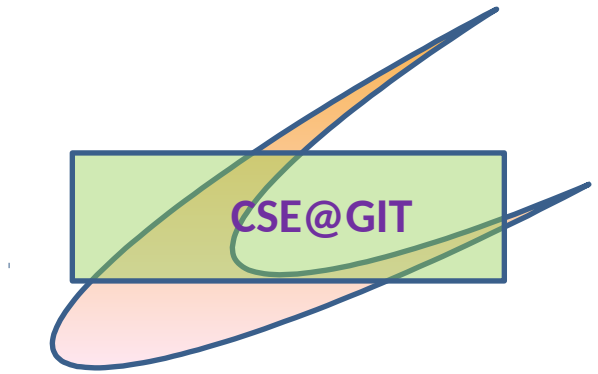
Problem Definition: Write a C/C++ POSIX compliant program to check the compile time and run time configuration limits:

SL. NO.	Limits
1	Number of clock ticks
2	Max number of child processes
3	Max path length
4	Max number of chars in file name
5	Max number of open files per process



Objectives of the Experiment:

1. To demonstrate the use of macros that are defined by POSIX.1 standard
2. To develop an understanding of checking the system and file related configuration limits using API's
3. To be able to differentiate between compile time and run time limits



Need of the Experiment

Why do we need to know the limits?

Applications should not assume any particular value for a limit. To achieve maximum portability, an application should not require more resource than the Minimum Acceptable Value. However, an application wishing to avail itself of the full amount of a resource available on an implementation may make use of the value according to the limits on that particular system.

For Ex: Max no of allowed chars in a filename

Theoretical Background of the Experiment

- **Compile Time Limits**

  Limits known at compile time

  Can be determined using manifested constants – defined in limits.h

- **Run Time Limits**

  Limits known at run time

  Can be queried using POSIX APIs

Theoretical Background of the Experiment

POSIX – Portable Operating System Interface for Computer Environments family of standards specified by the IEEE for maintaining compatibility between operating systems

- POSIX.1 & POSIX .1b define set of system configuration limits in the form of manifested constants; Many of these are inherited from UNIX.
- These constants are formed as: _POSIX_ and UNIX constant name
  Example : _POSIX_ and CHILD_MAX
  _POSIX_CHILD_MAX
  These constants are defined in <limits.h>

Compile Time Limits Checking

In POSIX Compile time system configuration limits can be queried using the following constants or macros

CONSTANT/MACRO NAME	MIN VAL	MEANING
_POSIX_CHILD_MAX	6	max no of child processes created at any time
_POSIX_OPEN_MAX	16	max no of files simultaneously opened
_POSIX_CLK_TCK		number of clock ticks per second
_POSIX_ARG_MAX	4096	max no in bytes passed as argument to exec function call
_POSIX_MQ_PRIO_MAX	2	max no of message priorities that can be assigned to messages
_POSIX_VERSION		version of posix running on a system

Run Time Limits Checking

In POSIX Run time system configuration limits can be queried using following three API's

API NAME	Dedicated for
long sysconf(const int limit_name);	Process related limits
long pathconf(const char *pathname, int limit_name);	file related limits
long fpathconf(const int filedescriptor, int limit_name);	File related limits

POSIX Compliance

To ensure program confirms to POSIX.1

#define _POSIX_SOURCE

#define _POSIX_C_SOURCE *_POSIX_C_SOURCE*

POSIX Compliance

_POSIX_C_SOURCE value----Meaning

198808L---- First version of POSIX.1 compliance

199009L---- Second version of POSIX.1 compliance

199309L---- POSIX.1 and POSIX.1b compliance

Algorithm for the experiment

- Declare the required header files limits.h, unistd.h, iostream.h
- Define POSIX standard constants
- Menu driven code for compile or run time limits checking
- Use macros for querying compile time limits
- Use sysconf for process related limits checking and pathconf or fpathconf for File related limits checking

Pseudo Code / Outline of the Algorithm

```
#define _POSIX_SOURCE

#define _POSIX_C_SOURCE 199309L

#include<iostream.h>

#include<unistd.h>

#include<limits.h>

int main()

{

    /* Menu for Compile Time Limits and Run Time Limits */

}
```

Pseudo Code / Outline of the Algorithm

For Compile Time Limits:

If the macro is defined output it's value

else a message indicating that it is not defined

Ex:

```
#ifdef _POSIX_OPEN_MAX
    cout<<"Max no of files simultaneously opened"<< POSIX_OPEN_MAX
 else
    cout<<"Macro _POSIX_OPEN_MAX not defined"
```

Pseudo Code / Outline of the Algorithm

For Run Time Limits:

Use sysconf for process related limits and

pathconf or fpathconf for file related limits

Pseudo Code / Outline of the Algorithm

Working with sysconf function:

```
if ((res=sysconf(_SC_CLK_TCK))==-1)
      perror("sysconf");
else
      cout << "Number of Clock Ticks is :  "
            << res << endl;
```

Pseudo Code / Outline of the Algorithm

Working with pathconf function:

```
if ((res=pathconf("/",_PC_OPEN_MAX))==-1)
        perror("pathconf");
else
        cout << "Maximum Number of Files opened per
        process : " << res << endl;
```

Pseudo Code / Outline of the Algorithm

Working with fpathconf function:

```
if ((res=fpathconf(0,_PC_NAME_MAX))==-1)
    perror("pathconf");
 else
    cout << "Maximum Length in Bytes in a File name:
     " << res << endl;
```

Sample Run

Compile Time Limits:

Max number of child processes	24
Max path length	80
Max number of chars in file name	255
Max number of open files per process	20

RunTime Limits:

Number of clock ticks	100
Max path length	255
Max number of chars in file name	255
Max number of open files per process	20

Learning Outcomes of the Experiment

At the end of the session, students should be able to :

1) Choose the macros and POSIX API for a task. [L1]

2) Make use of various macros for querying compile time configuration limits.[L3]

3) Experiment with the usage of APIs for run time configuration limits. [L3]

4) Compare compile time and run time limits. [L 4]