Unix Systems Programming and Compiler Design Laboratory (10CSL68)

April 14, 2015

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, Banglore
- 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, Banglore
- Neeta Ann Jacob, Faculty, The Oxford College of Engineering, Bangalore
- Rajesh N, Faculty, Sir M. Visvesvaraya Institute of Technology, Banglore
- 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
- Nishchal Gautam, Student, Sri Venkateshwara College of Engineering, Bangalore
- 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
- Rakesh P Gowda, 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 150engineering colleges in Karnataka, we still wanted to find more avenues to reach out to studentsand 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 there by 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 lead 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.

Syllabus

1. Write a C/C++ POSIX compliant program to check the following limits:
   (a) No. of clock ticks
   (b) Max. no. of child processes
   (c) Max. path length
   (d) Max. no. of characters in a file name
   (e) 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. Write a C/C++ program that outputs the contents of its Environment list.
6. Write a C / C++ program to emulate the unix ln command
7. Write a C/C++ program to illustrate the race condition.
8. 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.
9. Write a C/C++ program to avoid zombie process by forking twice.
10. Write a C/C++ program to implement the system function.
11. Write a C/C++ program to set up a real-time clock interval timer using the alarm API.

12. 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).

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

Contents

Chapter 1

Limit Checking

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

1. Number of clock ticks
2. Maximum number of child processes
3. Maximum path length
4. Maximum number of characters in a file
5. Maximum number of open files

1.1 Function Reference

1.1.1 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).

1.1.2 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.

1.2 Code

01GetLimits/usp01.cc

```
1 #define _POSIX_SOURCE
2 #define _POSIX_C_SOURCE 199309L
3
4 #include "iostream"
5 #include <unistd.h>
```

```
6  using namespace std;
7
8  int main()
9  {
10         cout << "No of clock ticks: " <<
               sysconf(_SC_CLK_TCK) << endl;
11         cout << "Maximum no of child processes: " <<
               sysconf(_SC_CHILD_MAX) << endl;
12         cout << "Maximum path length: " << pathconf("/",
               _PC_PATH_MAX) << endl;
13         cout << "Maximum characters in a file name: " <<
               pathconf("/", _PC_NAME_MAX) << endl;
14         cout << "Maximum no of open files: " <<
               sysconf(_SC_OPEN_MAX) << endl;
15         return 0;
16 }
```

1.3 Output

Open a terminal. Change directory to the file location in the terminal.
Run

```
$ g++ usp01.cc -o usp01.out
```

If no errors, run

```
$ ./usp01.out
```

The output should be something like this.

```
No of clock ticks: 100
Maximum no of child processes: 7082
Maximum path length: 4096
Maximum characters in a file name: 255
Maximum no of open files: 1024
```

Chapter 2

Printing POSIX defined configuration options

Write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros.

2.1 Description

_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.

2.2 Code

02PosixConfiguration/02_posix_configuration.cc

```
1  #define _POSIX_SOURCE
2  #define _POSIX_C_SOURCE 199309L
3
4  #include "iostream"
5  #include <unistd.h>
6
7  using namespace std;
8
9  int main()
10 {
11         #ifdef _POSIX_JOB_CONTROL
12                 cout  << "System supports POSIX job
                       control:" << _POSIX_JOB_CONTROL <<
                       endl;
13         #else
14                 cout << "System does not support POSIX
                       job control" << endl;
15         #endif
16         #ifdef _POSIX_SAVED_IDS
17                 cout << "System supports saved set UID
                       and GID:" << _POSIX_SAVED_IDS << endl;
18         #else
19                 cout << "System does not support saved
                       set GID and UID" << endl;
20         #endif
21         #ifdef _POSIX_CHOWN_RESTRICTED
22                 cout << "Chown restricted option is :"
                       << _POSIX_CHOWN_RESTRICTED << endl;
```

```
23          #else
24                  cout << "Chown Restricted not defined"
                        << endl;
25          #endif
26          #ifdef _POSIX_NO_TRUNC
27                  cout << "Truncation option is :" <<
                        _POSIX_NO_TRUNC << endl;
28          #else
29                  cout << "Truncation Option not defined"
                        << endl;
30          #endif
31          #ifdef _POSIX_VDISABLE
32                  cout << "disable char for terminal
                        files" << _POSIX_VDISABLE << endl;
33          #else
34                  cout << "char for terminal device files
                        will not be diasbled" << endl;
35          #endif
36          return 0;
37  }
```

2.3 Output

Open a terminal. Change directory to the file location in the terminal.
Run

```
$ g++ 02_posix_configuration.cc
```

If no errors, run

```
$ ./a.out
```

The output should be something like this.

```
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
```