Learning Report

Linux OS & Programming

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved By	Remarks/Revision Details
1	03-03-21	Devraj Sen	Manisha Chandra, Vishaal Balaji N		
2	04-03-21	Devraj Sen	Kamran Akhtar, Vinay Shirol		

Contents

Activities

Activity 1 – Design & Link with Libraries

Part A - Preparation
Part B - Simple Make file
Part C- Simple Make file with Inc and Src Folders
Part D- Static Libraries
Part E- Dynamic Libraries

GitHub Link for the codes: https://github.com/99003690/Activity1_Linux.git

Commands History:

- For producing .out and .o files (dep = dependencies)
  >> gcc dep1.c dep2.c dep3.c

- For executing the output considering a.out is the executable file
  >> ./a.out

- For creating a new file and editing
  >> nano file_name

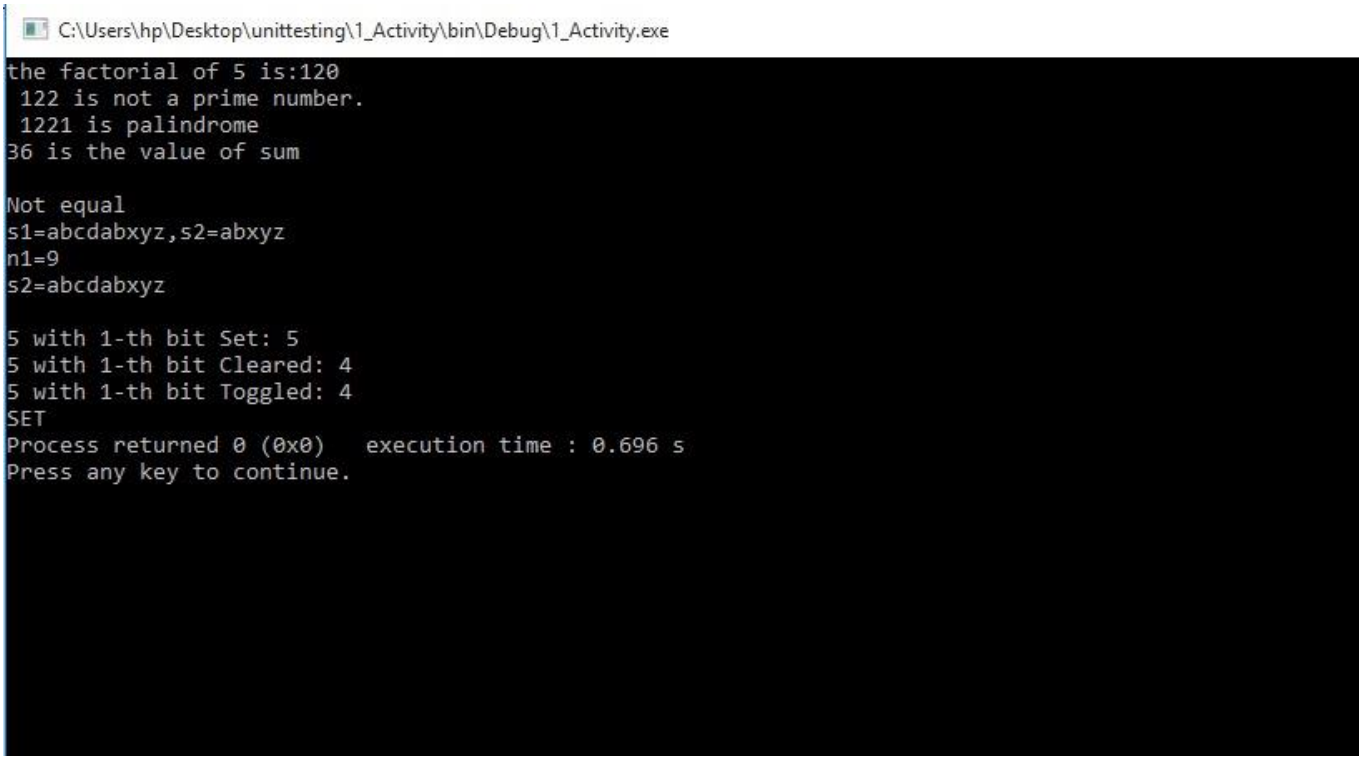
- For creating libraries
  >> ar rc libsimple.a dep1.o dep2.o
  >> gcc -L. dep1.o s1.out -lsimple
  >> gcc -L. dep1.o -o s1.out -lsimple
  >> gcc -L. dep1.o -o s2.out -lsimple -static

- Dynamic Library Linking:
  >>gcc -L. dep1.o -o d1.out -lsample

- Makefile:
  >> Target : Dependencies
  <Tab>gcc dependencies
  Run : Target
  <Tab> ./a.out

Makefile :

Output (on windows cmd):



Git Link :

https://github.com/99003690/1_Activity_LINUX.git

Activity 2 - <u>Process, System Calls and Threads Handling</u>

To do write simple programs using to execute Process, Systems Calls and Thread Handling, and to get familiarized with the unit testing using unity testing. In the program we need to code the given programs according to the Linux OS. Also we need to implement the concept of parent class and about zombie functions along with threads.

Leaning Outcome:

- Got to know how to make mini-shell, linking parent class and child class and threads.
- Getting familiarized with terminal usage and using of manuals in terminals.

Challenges:

- Implementing the concepts into working code.

Git Link :

https://github.com/99003690/1_Activity_LINUX.git

Activity 3: Semaphores and Mutex

Type of Activity: Individual

Goal of Activity: Implement producer consumer problem

Topics Covered:

- Mutex Lock
- Semaphores- Named and unnamed
- Race condition
- Deadlock
- Pipes
- Shared memory
- Message queue

Learning Outcomes:

- Learnt to implement sequencing and mutual exclusion.
- Prioritizing or locking a particular process for sequencing the flow of program.
- Working with named and unnamed semaphores, and using named semaphores in shared memory.
- Analyzing the return type for mutex to check for success or failure.
- Using threads for working with producer and customer.
- Handling context switching in order to avoid deadlocks.
- Using pipes and fifo to overcome limitations of semaphores and mutex.
- Using operations on shared memory such as read write and update.

Challenges: Understanding the race condition

Git link : https://github.com/99003690/Assignment3.git

References:

[1] https://www.tutorialspoint.com/gnu_debugger/index.htm

[2] https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

[3] https://tutorialspoint.com/operating_system/os_linux.htm