./

Learning Report – LINUX OS and Programming

Course Code: <CODE>



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1 | 2-03-21 | Sri Naga Anjaneya D |  |  |  |
| 2 | 4-03-21 | Sri Naga Anjaneya D |  |  |  |
| 3 | 6-03-21 | Sri Naga Anjaneya D |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Introduction to Linux 4](#_Toc65650977)

[**Activity 1** 4](#_Toc65650979)

[Case Study - Design & Link with Libraries 4](#_Toc65650980)

[**Activity 2** –](#_Toc65650981) 7

Processes, Signals and Threads...................................................................................................7

[**Activity 3**](#_Toc65650982) 8

Semaphores and Mutex…………………………………………………………………………………………….8

[**Activity 4**](#_Toc65650983) 9

Memory Mapping, Stack, Message and Shared Memory………………………………………………….[9](#_Toc65650984)

# ACTIVITY 1 – DESIGN & LINK WITH LIBRARIES

**Part A – Preparation**

Make the various source file and header files as mentioned in the question.

**Part B - Simple Make file**

Make a simple Makefile assuming all the source and header files are in the same folder.

**Part C- Simple Make file with Inc and Src Folders**

Make a Makefile connecting all the source and header files assuming they are in separate folders – inc and src respectively.

**Part D- Static Libraries**

Generate all the required libraries. Link the static libraries with the test code and test the statically linked executable. At last analyze all the outcomes.

**Part E- Dynamic Libraries**

Generate all the required libraries. Link the static libraries with the test code and test the dynamically linked executable. At last analyze all the outcomes.

# Introduction to Linux

Linux is a family of open source Unix like operating system based on Linux kernel.

## Concepts learnt as part of Activity 1 are as follows:

1. Understanding Tool chain
2. GNU Tools
3. Makefile
4. Static Libraries & Linking
5. Dynamic Libraries & Linking
6. Static Analysis of Code
7. Debugging Tool

Software or packages required are:

1. GNU tools
2. Vaalgrind
3. Make
4. Git

## **Case Study - Design & Link with Libraries**

This case study has the following requirements:

* Source folder containing the source files of mystring, myutils, bit\_set\_reset
* Header folder containing the header files of mystring, myutils, bit\_set\_reset
* Finally the test file

**Functions in mystring file:**

* Mystrlen
* Mystrcpy
* Mystrcat
* mystrcmp

**Functions in myutils file:**

* Factorial
* isPrime
* isPalindrome
* vsum

**Functions in bit\_set-reset:**

* Set
* Reset
* Flip
* Query
* Generated Makefile

**Command to generate makefile**

Build: test.c Mystring.c myutils.c bit\_set\_reset.c

gcc test.c Mystring.c myutils.c bit\_set\_reset.c

run: ./a.out

**Static library generation**

Commands to generate static library file are as follows:

gcc filename.c -c

gcc filename.c -c

ar rc libsimple.a fiename.o

gcc test.c -c

gcc -L.test.o -o filename.out -lsimple

gcc -L.test.o -o filename.out –lsimple -static

**Dynamic library generation**

Commands to generate Dynamic library are as follows:

gcc filename.c-c

gcc filename.c-c

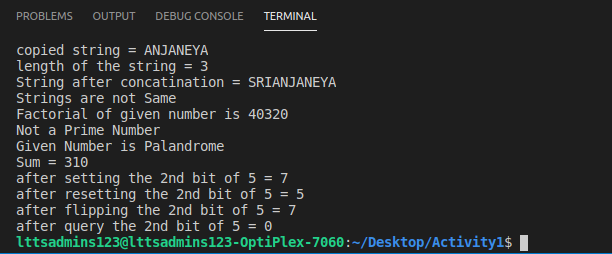
gcc -shared-o libsample.so filename.os filename.o

gcc test.c-c

gcc -L.test.o-o d1.out-lsample

LD\_LIBRARY\_PATH=../d1.out

**Makefile Output:**



Link: <https://github.com/99003527/Assessment1.git>

**Activity 2 – Process, Signals and Threads**

**Description:**

The activity dealt with the process types in Linux OS and its implementation in code, sending signals to the specified target process and performing various functions using threads concept. It includes the following process:

- Parent and Child process

- Zombie and Daemon process

- Context Switching

- System calls

- Generating process id of the process

- Sending signal using appropriate Linux commands to the target process

- Thread process cycle

**Learning Outcomes:**

· Learnt to create child process and to generate the process ids of both parent and child.

· Learnt to load the process control block of the process table onto the memory which includes context saving and context loading.

· Implemented C programs to generate multiple child processes and launch commands in the processes using the appropriate system calls.

· Learnt to execute programs using threads and performed functions like printing current time periodically and creating a calendar.

Challenges:

· Faced challenge while creating a multifile program using system calls as the program contains multiple source files holding some functions.

· Faced challenge while writing code to perform functions which Linux commands does with-out using those commands directly.

**Learning Outcomes:**

1. Able to write the program to copy one file contents to other using open, read, write, close system calls.

2. Able to write a program to send specific signal to a target process.

3. To design the mini shell and program to compile & link any program within child process by launching gcc using execlp and program to build multifile program using fork & exec.

Able to write a program to print current time periodically.

**Link:** [**https://github.com/99003527/Assessment1.git**](https://github.com/99003527/Assessment1.git)

**Activity 3 – Semaphores and Mutex**

**Description:**

Semaphores is basically a variable or abstract data type used to control access to a common resource by multiple processes and avoid critical section problems in a concurrent system such as a multitasking operating system whereas Mutex or Mutual Exclusion Object is used to give access to a resource to only one process at a time. So in today’s activities we have learned about these two kinds of variable and exclusion objects so that we can execute our system in a better way.

**Learning Outcomes:**

· Learnt to create mutex and semaphores.

· Learnt to work with either of these two and using these two simultaneously in a program.

· Implemented C programs to generate multiple interrupt functions like these two and work on these.

Challenges:

· Faced challenge while writing raw codes using these and executing them.

**Link:** **https://github.com/99003527/Assessment1.git**

**Activity 4 – Memory Mapping, Stack, Message and Shared Memory**

**Description:**

Assigning different memories, stacks and other memory processes to work on and use this memory to execute out code accordingly.

**Learning Outcomes:**

· Learnt to create mutex and semaphores with Stack and Message commands.

· Learnt to work with either of these two and using these two simultaneously in a program.

· Implemented C programs to generate multiple interrupt functions like these two and work on these along with Memory Mapping, Stack, Message and Shared Memory

Challenges:

· Faced challenge while writing raw codes using these and executing them.

**Link:** [**https://github.com/99003527/Assessment1.git**](https://github.com/99003527/Assessment1.git)