Learning Report

Linux OS & Programming

L&T Technology Services

MANISHA CHANDRA
PS.NO: 99003684

Document History

1	1-03-21	Manisha Chandra	Mohammad Hasaan Zafar, Devraj Sen		
2	4-03-21	Manisha Chandra	Poojashri N, Reeshav Rout		
3	6-03-21	Manisha Chandra			

CONTENTS

ACTIVITY 1 –MAKEFILE DESIGN & LINK WITH LIBRARIES

Description:

The activity dealt with the implementation of C program in Linux OS and the linking of those programs with the libraries. It includes the following execution process:

- **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.

Learning Outcomes:

- Learnt to club the files into two separate folders as source and header files and to link them with the libraries.
- Learnt to create a Makefile using those source and header files by executing Linux commands.
- Generated the required libraries and linked the static libraries with test code.
- Learnt to test the statically linked and dynamically linked executable files.

Challenges:

- Faced challenge while linking the Include header file with the whole executable code.
- Faced challenge with Linux commands on implementing the Makefile.

GitHub Link: 99003684/Linux_Assignments (github.com)

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 without using those commands directly.

GitHub Link: 99003684/Linux_Assignments (github.com)

ACTIVITY 3: INTER PROCESS COMMUNICATION (IPC)

Description: When threads and child processes are made there is a need of communication between them i.e. return some data. This is where Inter process communication (IPC) comes into play.

Requirement of IPC

- Data exchange
- Synchronization
- Dependency / Sequencing
- Mutual Exclusion

For the various needs the following are used

- Data exchange -- shared memory, message queues, FIFOs/pipes
- Mutual exclusion -- semaphore, MUTEX, spinlocks
- Dependency -- semaphores, condition variables / event flags

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:

- Race Condition
- Deadlock

GitHub link: 99003684/Linux_Assignments (github.com)

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