Learning Report

Linux OS & Programming

LTTS
GLOBAL
ENGINEERING
ACADEMY

L&T Technology Services

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved By	Remarks/Revision Details
1	1-03-21	Mohammad Hasaan Zafar	Manisha Chandra, Vinay Shirol		
2	04-03-21	Mohammad Hasaan Zafar	Vishal Balaji Dangeti Ravi		
3	06-03-21	Mohammad Hasaan Zafar	Vishal Balaji Dangeti Ravi		

Contents

Activity 1 – 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:

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

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
  ```

GitHub Link for the codes: https://github.com/99003674/LinuxAssignments

Activity 2 - Process, System Calls and Threads Handling

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 processes:

- 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:

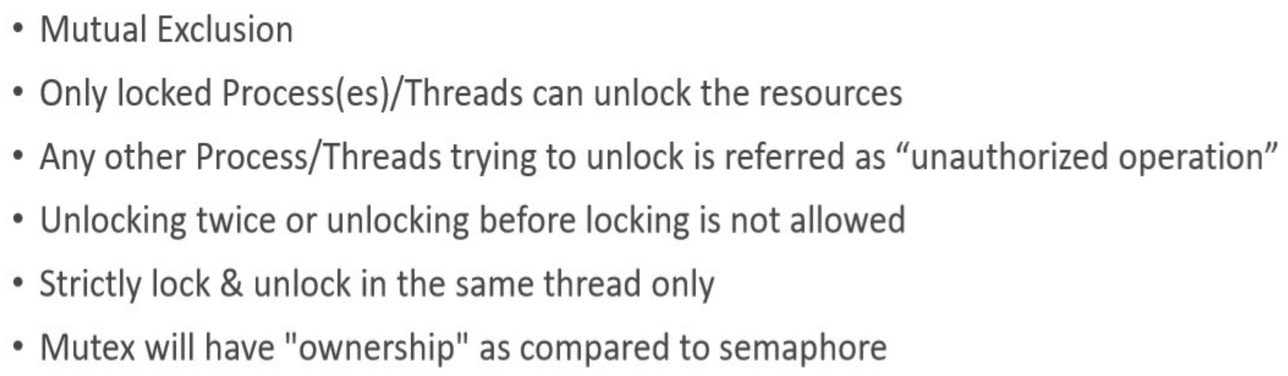
- Faced challenge while creating a multifile program using system calls as the program contains multiple source files holding some functions.
- Faced challenge while writing code to perform functions which Linux commands does without using those commands directly.

Github Link :https://github.com/99003674/LinuxAssignments

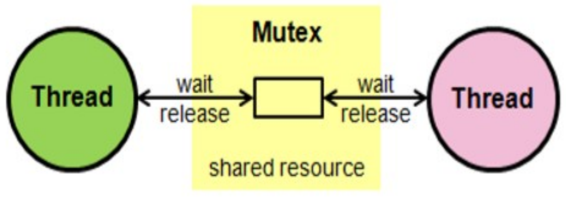
Activity 3 – Semaphores and Mutex

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.



- Mutual Exclusion
- Only locked Process(es)/Threads can unlock the resources
- Any other Process/Threads trying to unlock is referred as "unauthorized operation"
- Unlocking twice or unlocking before locking is not allowed
- Strictly lock & unlock in the same thread only
- Mutex will have "ownership" as compared to semaphore



Semaphores

- Sequencing, Signaling mechanism, used for process/thread synchronization
- Manage and protect access to shared resources
- Kernel level data structure Types of usage
- Binary Semaphore • Value of semaphore ranges between 0 & 1
- Mutual Exclusion / Access to a single resource
- Counting Semaphore

• Value of semaphore can be 0 (zero) & any positive value
• Accessing/sharing multiple similar resources Two (2) varieties of semaphores
• Traditional System V semaphores
• POSIX semaphores.

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.

Github Link :https://github.com/99003674/LinuxAssignments

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.

Github Link : https://github.com/99003674/LinuxAssignments