

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	2-03-21	Reeshav Rout	Vishaal Balaji, Kamran Akthar		
2	4-03-21	Reeshav Rout	Manisha Chandra, Rajath PD		
3	6-03-21	Reeshav Rout	Manisha Chandra, Rajath PD		

Document History

Contents

INDIVIDUAL ACTIVITIES.....	3
ACTIVITY 1 - DESIGN & LINK WITH LIBRARIES.....	4
ACTIVITY 2 - PROCESS, SYSTEM CALLS AND THREADS HANDLING.....	8
ACTIVITY 3 - SEMAPHORES AND MUTEX.....	8
ACTIVITY 4 - MEMORY MAPPING, STACK, MESSAGE AND SHARED MEMORY.....	9
ACTIVITY 5 -	5

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:

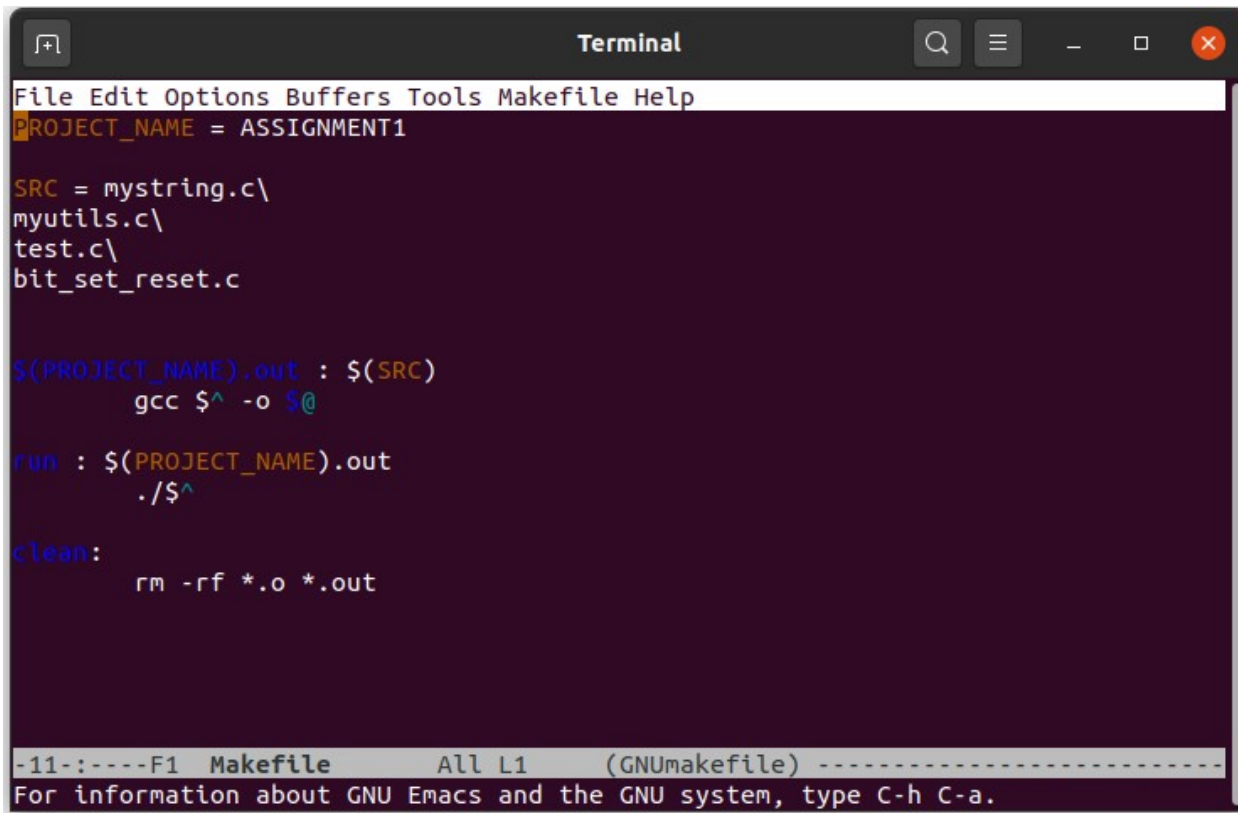
- **Part A – Preparation**
Make the various source file and header files as mentioned in the question.
- **Part B - Simple Make file**
Make a simple Makefile assuming all the source and header files are in the same folder.
- **Part C- Simple Make file with Inc and Src Folders**
Make a Makefile connecting 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 challenge while linking the Include header file with the whole executable code.
- Faced challenge with Linux commands on implementing the Makefile.

Makefile :

```
File Edit Options Buffers Tools Makefile Help
PROJECT_NAME = ASSIGNMENT1

SRC = mystring.c\
myutils.c\
test.c\
bit_set_reset.c

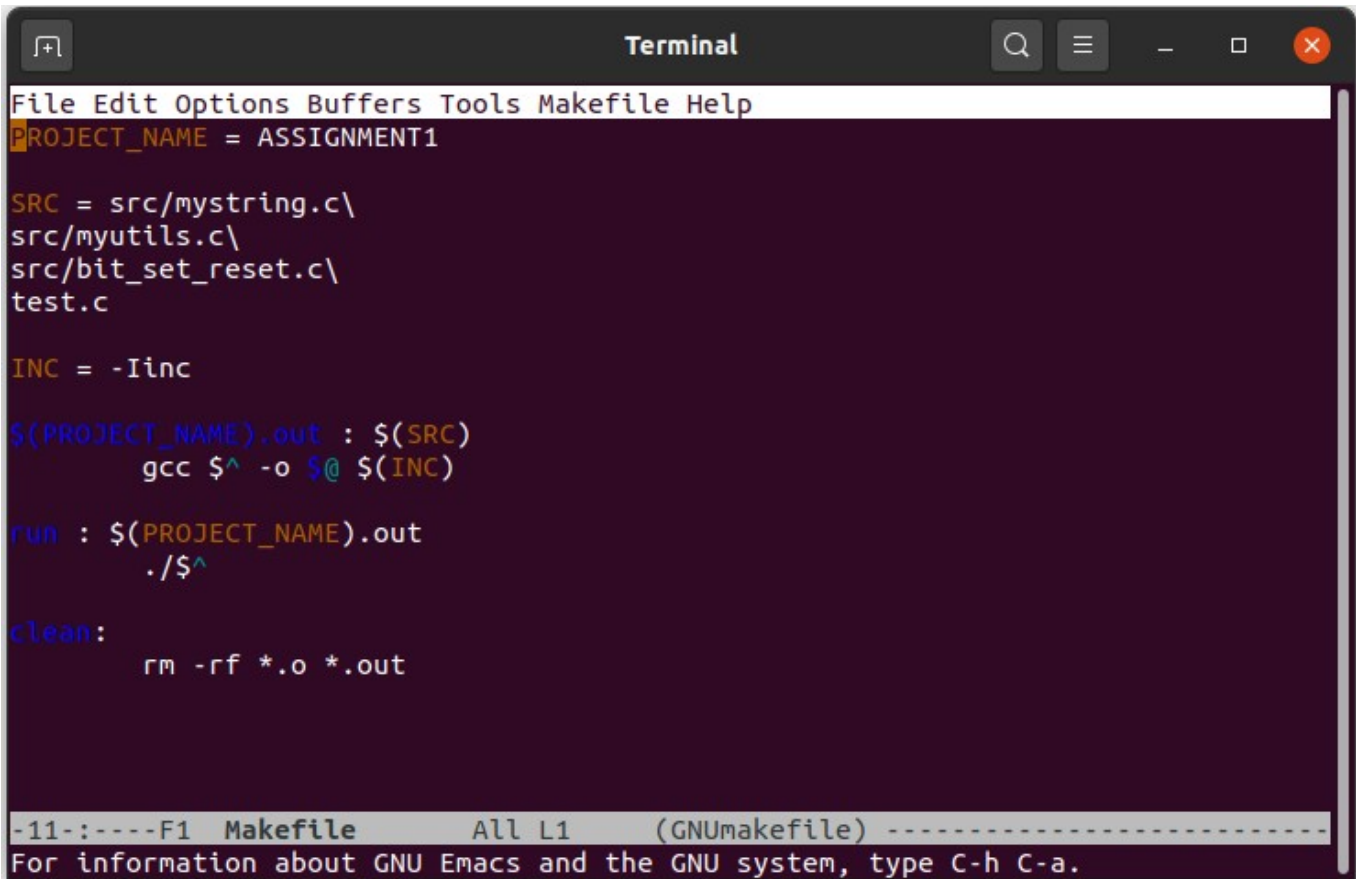
$(PROJECT_NAME).out : $(SRC)
    gcc $^ -o $@

run : $(PROJECT_NAME).out
    ./$^

clean:
    rm -rf *.o *.out

-11-:----F1  Makefile      All L1      (GNUmakefile) -----
For information about GNU Emacs and the GNU system, type C-h C-a.
```

Makefile when .c and .h files are in separate folders.



```
File Edit Options Buffers Tools Makefile Help
PROJECT_NAME = ASSIGNMENT1

SRC = src/mystring.c\
src/myutils.c\
src/bit_set_reset.c\
test.c

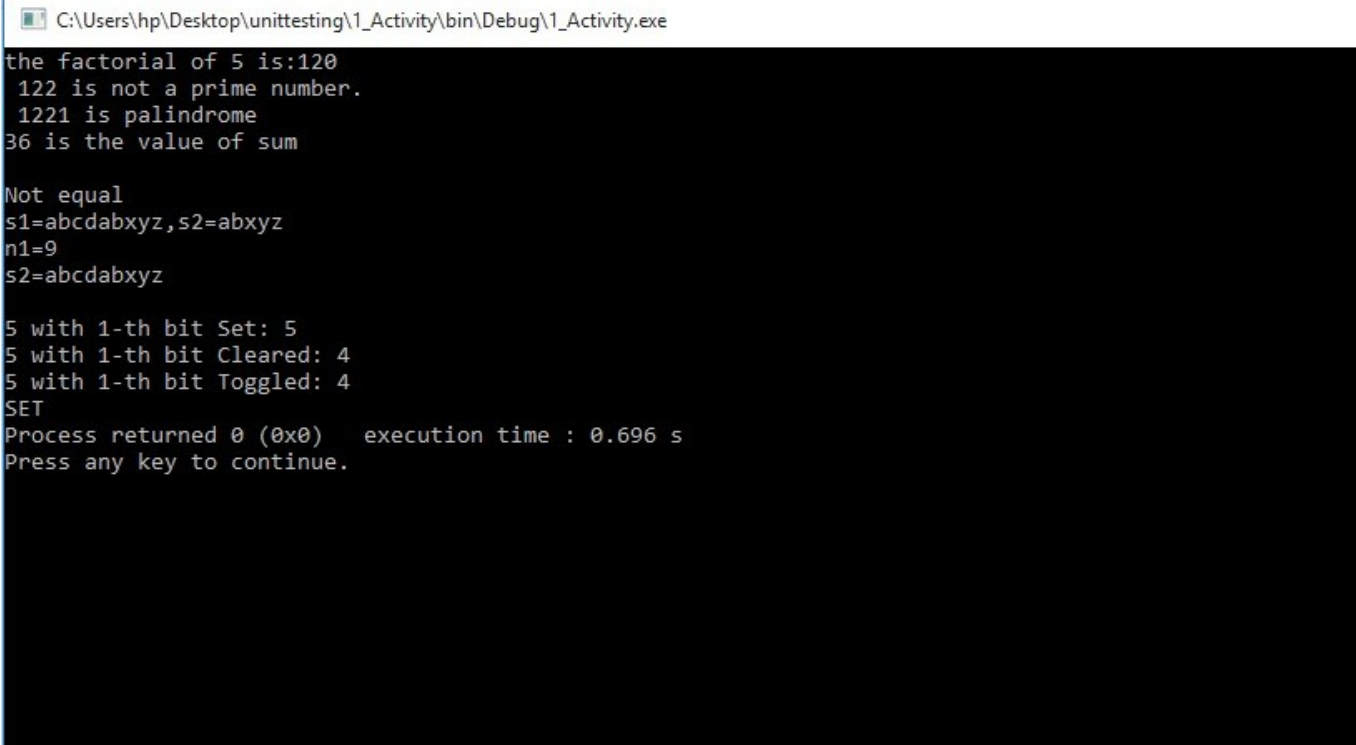
INC = -Iinc

$(PROJECT_NAME).out : $(SRC)
    gcc $^ -o $@ $(INC)

run : $(PROJECT_NAME).out
    ./$^

clean:
    rm -rf *.o *.out

-11-:----F1  Makefile      All L1      (GNUmakefile) -----
For information about GNU Emacs and the GNU system, type C-h C-a.
```

Output (on windows cmd):

```
C:\Users\hp\Desktop\unittesting\1_Activity\bin\Debug\1_Activity.exe
the factorial of 5 is:120
122 is not a prime number.
1221 is palindrome
36 is the value of sum

Not equal
s1=abcdabxyz,s2=abxyz
n1=9
s2=abcdabxyz

5 with 1-th bit Set: 5
5 with 1-th bit Cleared: 4
5 with 1-th bit Toggled: 4
SET
Process returned 0 (0x0) execution time : 0.696 s
Press any key to continue.
```

Github Link : https://github.com/99003696/LINUX_OS_PROGRAMMING.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 without using those commands directly.

Github Link : https://github.com/99003696/LINUX_OS_PROGRAMMING.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.

Github Link : https://github.com/99003696/LINUX_OS_PROGRAMMING.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.

Github Link : https://github.com/99003696/LINUX_OS_PROGRAMMING.git