./

Learning Report-Linux OS and Programming



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1 | 01/03/21 | 99003509 |  |  |  |
| 2 | 02/03/21 | 99003509 |  |  |  |
| 3 | 04/03/21 | 99003509 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

Contents

[Contents 3](#_Toc65820780)

[Activity 1: Design & Link with Libraries 4](#_Toc65820781)

[Activity 2: System Calls and Signals, Processes and Threads 5](#_Toc65820782)

[Activity 3: Semaphores and Mutex 6](#_Toc65820783)

# Activity 1: Design & Link with Libraries

**Type of Activity**: Individual

**Goal of Activity**: Static library and Dynamic library. Creating user defined libraries and linking user defined functions as library both statically and dynamically.

**Topics Covered:**

* Linux OS Architecture
* Linux OS commands
* GCC & Build Process
* Utilities
* Static & Dynamic Libraries
* Makefile creation

**Learning Outcomes:**

* Implementing C program builder.
* Using utilities and implementing codes in separate header and C files.
* Created our library and learnt to link that to a static and dynamic type.
* Usage of id config to link a dynamic library.
* Implementing Makefile for the same.

**Challenges:** Linking static, dynamic Makefile

**References:**

1. <https://web.microsoftstream.com/video/5cc492de-e71c-4c15-98ff-53727580a5b6>
2. <https://web.microsoftstream.com/video/ab1d8a45-bfb2-4187-9eda-cd83d9c31f5b>
3. <https://web.microsoftstream.com/video/9e33e60e-91e3-4b6f-ac23-937e83897e86>
4. <https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html>
5. https://embetronicx.com/tutorials/unit\_testing/unit-testing-in-c-testing-with-unity/

# Activity 2: System Calls and Signals, Processes and Threads

**Type of Activity**: Individual

**Goal of Activity**:

* To count number of lines, words, characters in given file.
* To copy one file contents to other using open, read,write,close system calls.
* to send specific signal to a target process
* Compile & link any c/c++ program within child process by launching gcc using execl/execlp.
* Designing a mini shell.
* Building multifile program using fork & exec.
* Print current time periodically.
* Finding min/max element from large array using parallel computations.
* Compute parallel sum of large array.

**Topics Covered:**

* Kernel
* System calls
* Scheduling
* Interrupts
* Process life cycle

**Learning Outcomes:**

* How to make system call and implement different system calls. Based on file descriptors by any process.
* How to handle and run a process.
* How to create parent and child process.
* Creating multiple child processes.
* Kill or stop process.
* Implementing how to wait a process and override in a child process to give our own.
* Learnt to avoid making blocking calls in thread to avoid getting the whole process blocked.
* Over writing child process using exec signals
* Blocking parent process till completion of child process.

**Challenges:** Program to send specific signal to a target process

**References:**

1. <https://web.microsoftstream.com/video/5cc492de-e71c-4c15-98ff-53727580a5b6>
2. <https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/>
3. [https://www.cs.uregina.ca/Links/class-info/330/SystemCall\_IO/SystemCall\_IO.html#FileIO](https://www.cs.uregina.ca/Links/class-info/330/SystemCall_IO/SystemCall_IO.html" \l "FileIO)

# 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 contidition

**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>

**Git link: https://github.com/99003509/Linux/**