va\_start,va\_end from stdarg.h for the implementation va\_start,va\_end from stdarg.h for the implementation ./

Learning Report –

Linux and OS programming

Course Code: <CODE>



Vinti Sai Pranitha

PS no: 99003567

Module: Linux and OS programming

Version Number:

Team Members :

Team No:

Module: Model Based System Engineering

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1.0 | 2.03.2021 | Vinti Sai Pranitha |  |  | 1st draft |
| 2.0 | 3.03.2021 | Vinti Sai Pranitha |  |  | 2nd draft |
| 3.0 | 5.03.2021 | Vinti Sai Pranitha |  |  | 3rd draft |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# 

Contents

[Assignment 1 - Design & Link with Static and Dynamic Libraries 4](#_Toc65879208)

[Assignment 2 – Signal calls, processes and Threads 5](#_Toc65879209)

[Assignment 3 – IPC – Semaphores and mutex 7](#_Toc65879210)

# Assignment 1 - Design & Link with Static and Dynamic Libraries

**Type of activity:** Individual

**Problem Statement:**

* Develop the following functions in respective source files, in a subdir called 'src`

- mystrlen, mystrcpy, mystrcat, mystrcmp

- factorial, isPrime, isPalindrome, vsum

- set, reset, flip, query

* Provide prototypes in different header files, in a subdir called `include`

- mystring.h, myutils.h, bitmask.h

* Write a simple test code to invoke above functions, say `test.c`.

**Topics Covered:**

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

**Learning Outcomes:**

* Able to write the source files and header files without making use of scanf() functions and aslo created a makefile to compile all the source files.

**Challenges faced:**

* Faced problems in creating static and dynamic linking makefile.

**References:**

<https://web.microsoftstream.com/video/9a2b1eba-61a3-4547-8292-374b2eeb5265?channelId=04fdad23-021c-4e64-bb7c-06b2469801f9%20%E2%80%A2>

<https://web.microsoftstream.com/video/5cc492de-e71c-4c15-98ff-53727580a5b6?channelId=04fdad23-021c-4e64-bb7c-06b2469801f9>

# Assignment 2 – Signal calls, processes and Threads

**Type of activity:** Individual

**Problem Statement:**

**System Calls and Signals**

* Write a program to copy one file contents to other using open,read,write,close system calls (like cp command, which takes source, destination files as cmd line args).
* Write a program to count no.of lines, words, characters in given file (like wc command)
* Write a program to send specific signal to a target process (with given id, like kill command)

**Processes**

* Design a mini shell(5 - 10 commands) of your own as follows
* take command name as input from user (string format)
* launch the command in the child process using execl/execlp
* parent will wait for completion of child and print exit status
* Write a program to compile & link any c/c++ program within child process by launching gcc using execl/execlp.
* Write a program to build multifile program using fork & exec as follows
* There are multiple source files holding some functions
* Create multiple child processes to compile each source file (execl/execlp).
* On successful completion, launch another child process for linking.
* On successful linking, launch another child process for executing.

**Threads**

* Write a program to compute parallel sum of large array using threads.
* For e.g. if there are 1000 elements in array, you may create 10 threads where each thread will compute some of 100 elements each, on completion of threads main thread will consolidate final total
* Write a program to find min/max element from large array(1000 data points) using parallel computations (multthreading)
* Write a program to print current time periodically (Hint:- alarm, time, ctime)

**Topics Covered:**

* Linux OS Architecture
* Interrupts
* System calls
* Processes
* Process related commands

**Learning Outcomes:**

* able to write the program to copy one file contents to other using open, read, write, close system calls.
* Able to write a program to send specific signal to a target process.
* To design the mini shell and program to compile & link any program within child process by launching gcc using execl and program to build multifile program using fork & exec.
* Able to write a program to print current time periodically.

**Challenges faced:**

* Understanding the threads concept.

**References:**

<https://linuxhint.com/linux-exec-system-call/>

<https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/>

# Assignment 3 – IPC – Semaphores and mutex

**Type of activity:** Individual

**Problem Statement:**

* Implement producer consumer problem using Stack operations using semaphores
* Implement producer consumer problem using circular buffer operations using semaphores
* Repeat above two problem by replacing semaphores with mutex wherever possible
* Implement a simple client-server scenario using message queues
* Client process send a string over message queue
* Server process toggles the string and send back to client.
* Implement a simple client-server scenario using message queues
* Client process send requested command over message queue
* Server process receives requested command over message queue and executes in a child process using execl/execlp
* Write a program to retrieve file attributes using message queue as follows
* Client process sends requested file name to server over message queue
* Server process retrieves file attributes using lstat and send back to client
* Client will display the file attributes
* Write a program to implement simple chat application between two processes using

named pipes (FIFOs)

* Implement producer consumer problem between two processes using shared memory and named semaphores (POSIX APIs)

**Topics Covered:**

* Semaphores
* Shared memory
* Mutex
* FIFO/pipes
* Deadlocks
* Race conditions
* Message queues

**Learning Outcomes:**

* Implementation of 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.
* Usage of threads for producer consumer problem.
* Usage of mutex()
* Shared memory
* FIFO/pipes concept
* Race conditions

**Challenges faced:**

* Understanding the race conditions concept and semaphores.

**References:**

<https://www.youtube.com/watch?v=ukM_zzrIeXs>

<https://youtu.be/UM4tk3J6WxQ>

<https://youtu.be/G2vwkBZy894>

<https://www.youtube.com/watch?v=xKqO04SN6C0>