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 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# 

Contents

[Assignment 1 - Design & Link with Static and Dynamic Libraries 4](#_Toc65710630)

[Assignment 2 – Signal calls, processes and Threads 8](#_Toc65710631)

# 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>

**Creating Src Files:**

For implementing mystrlen, mystrcpy, mystrcat, mystrcmp functions, a source file with name mystring.c has been created. This source file consists of definitions of the above mentioned functions.

Mystrlen function – It will take string as input and returns the length of the string.

Mystrcpy function – It will take two strings as input and it will copy the string pointed by source to the

Destination. It returns the copied string.

Mystrcat function – It will concatenate two strings and returns the final concatenated string.

Mystrcmp function – It will compare two strings which are given as an input and it will return 0 if both

the strings are identical. It will return a negative value if the ASCII value of the

first unmatched character is less than the second. It will return a positive value if

the ASCII value of the first unmatched character is greater than the second.

For implementing factorial, isPrime, isPalindrome, vsum, a source file with name myutils.c has been created.

Factorial function – It will takes a positive integer from the user and computes the factorial using for

loop.

isPrime function – The function will take a number as input and it will check if it’s a prime number

or not and will return a statement mentioning the same.

isPalindrome function – The function will check if the number is a palindrome or not. It will return 0 if

the number is not a palindrome and it will return 1 if the number is palindrome.

Vsum function - The function which will take variable no.of arguments. It makes use va\_list,

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

For implementing set, reset, flip, query, a source file with name bitmask.c has been created.

Set function – It will set the bit position for the given number.

Reset function – It will reset all the bits of the given number to zero.

Flip function – It will flip the bit value of the number.

Query function –

**Include header Files:**

For implementing mystrlen, mystrcpy, mystrcat, mystrcmp functions, a source file with name mystring.c has been created. This source file consists of definitions of the above mentioned functions.

All the function declarations related to mystring.c is added in the mystring.h.

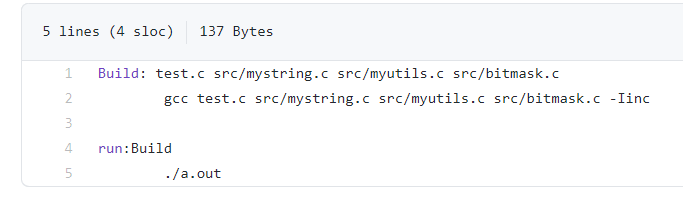
Similarly all the function declarations related to myutils.c and bitmask.c are added in the myutils.h and bitmask.h respectively.

**Test.c file:**

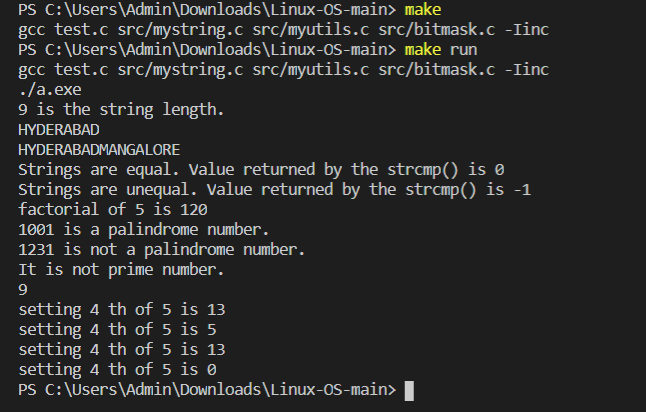
In test.c file we called all the functions that were defined in the source files and the outputs are printed using the printf functions.

**Simple Makefile:**

Makefiles are special format files that help build and manage the projects automatically. It aids in simplifying building program executable that may need various modules. To determine how the modules, need to be compiled or recompiled together, make takes the help of user-defined makefiles.



By running the command **make** in the terminal, it will compile the source files by including the header files. Then by running the **make run** command it will execute the a.out file and the output will be displayed in the terminal.



**Static Libraries:**

**Static Linking and Static Libraries** is the result of the linker making copy of all used library functions to the executable file. Static Linking creates larger binary files, and need more space on disk and main memory. Examples of static libraries (libraries which are statically linked) are, ***.a*** files in Linux and ***.lib***files in Windows.

Steps for creating static library:

1. Creating a source .c file
2. Creating the header file.
3. Creating .o files for all the .c files

Gcc mystring.c myutils.c bitmask.c test.c –o

1. Creating the static libraries.

ar rc libmystring.a mystring.o

ar rc libmyutils.a myutils.o

ar rc libmasking.o bitmask.o

1. Link the test.c program to the static library.

Gcc –L. test.o –o final\_static.out –lmystring –lmyutils –lbitmask –static

**Dynamic Libraries:**

A **Dynamic Library** is a collection of object files that can be linked to any program at run-time by inserting the location of the function in memory to the executable. This provides a way for code to be used even though it could be loaded anywhere in memory.

Gcc –shared –o libmystring.so libmyutils.so libmasking.so

Gcc –L. test.o –o final\_dynamic.out –lmystring –lmyutils –lbitmask

# 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/>