Learning Report

Linux OS & Programming

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved By	Remarks/Revision Details
1	1-03-21	Alen Reji Kokkad			

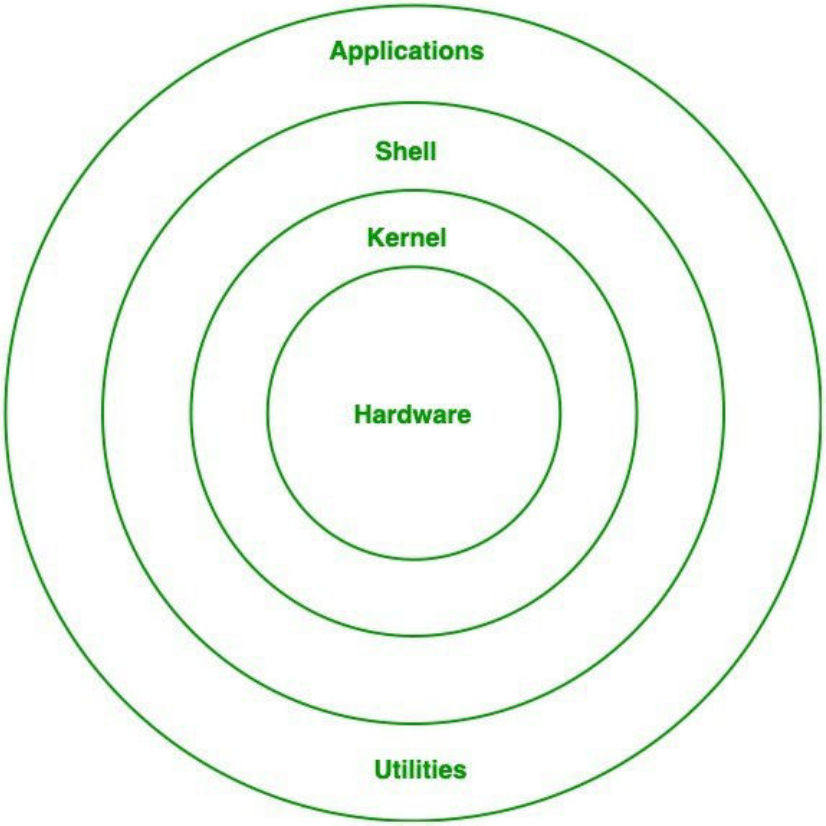
Document History

Contents

LINUX OS ARCHITECTURE

Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It was initially released by **Linus Torvalds** on September 17, 1991. It is a free and open-source operating system and the source code can be modified and distributed to anyone commercially or noncommercially under the GNU General Public License.

Initially, Linux was created for personal computers and gradually it was used in other machines like servers, mainframe computers, supercomputers, etc. Nowadays, Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc. The biggest success of Linux is Android (operating system) it is based on the Linux kernel that is running on smartphones and tablets. Due to android Linux has the largest installed base of all general-purpose operating systems. Linux is generally packaged in a Linux distribution.

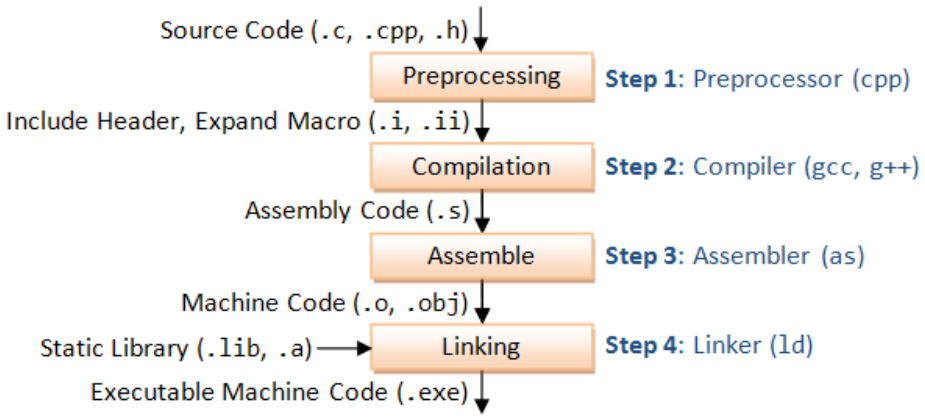


Kernel: Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes. Different types of the kernel are:

- Monolithic Kernel
- Hybrid kernels
- Exo kernels
- Micro kernels

1 **System Library: Is**the special types of functions that are used to implement the functionality of the operating system.
2 **Shell:** It is an interface to the kernel which hides the complexity of the kernel's functions from the users. It takes commands from the user and executes the kernel's functions.
3 **Hardware Layer:** This layer consists all peripheral devices like RAM/ HDD/ CPU etc.
4 **System Utility:** It provides the functionalities of an operating system to the user.

GCC & BUILD PROCESS



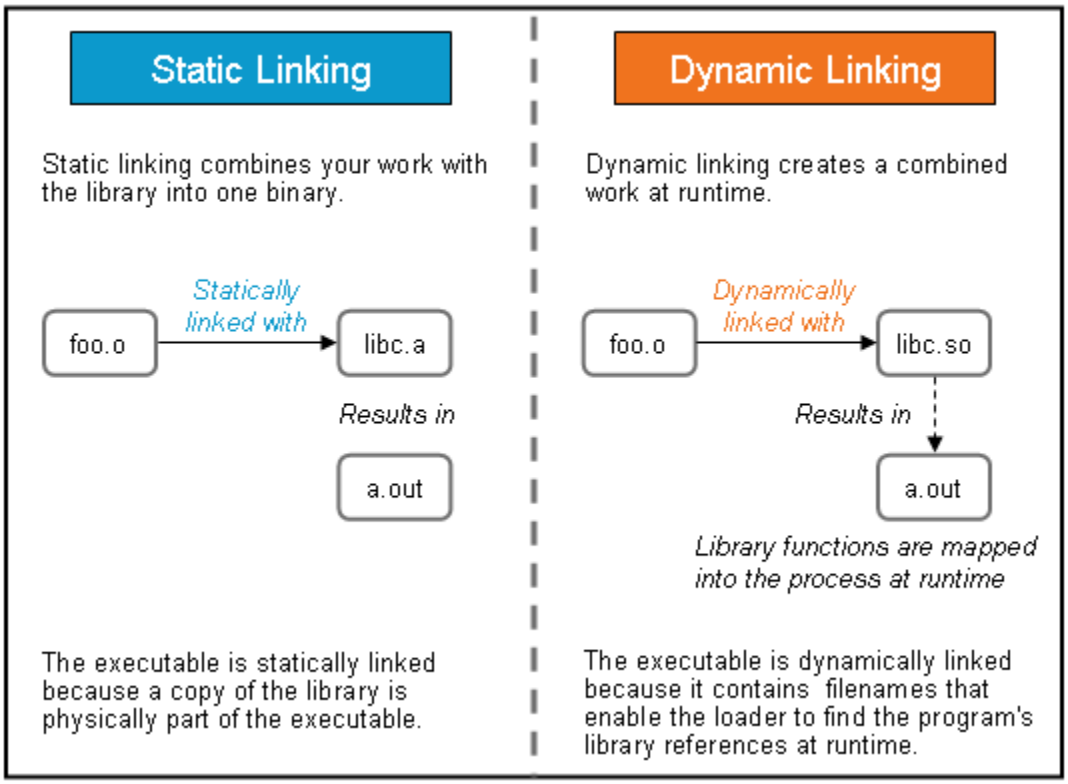
GCC compiles a C/C++ program into executable in 4 steps as shown in the above diagram. For example, a "gcc -o hello.exe hello.c" is carried out as follows:
Pre-processing: via the GNU C Preprocessor (cpp.exe), which includes the headers (#include) and expands the macros (#define). > **cpp hello.c > hello.i** The resultant intermediate file "hello.i" contains the expanded source code.

Compilation: The compiler compiles the pre-processed source code into assembly code for a specific processor. > **gcc -S hello.i** The -S option specifies to produce assembly code, instead of object code. The resultant assembly file is "hello.s".

Assembly: The assembler (as.exe) converts the assembly code into machine code in the object file "hello.o". > **as -o hello.o hello.s**

Linker: Finally, the linker (ld.exe) links the object code with the library code to produce an executable file "hello.exe". > **ld -o hello.exe hello.o ...libraries...**

STATIC & DYNAMIC LIBRARIES



When using a **dynamic library**, the programmer is referencing that library when it needs to at runtime. For instance, to access the string length function from the standard input/output header file — you can access it dynamically. It will find the program's library reference at runtime because of the dynamic loader. It then loads that string length function into memory. Thus, the dynamic library accessibility must be readily available or it becomes powerless.

ACTIVITY-1

Learning Outcomes: Was given activity based on the topics tutored like creating multiple functions with many operations like string compare, concatenation and creating Makefiles depending on the conditions.

Challenges: While doing dynamic and static Makefiles

Learning Resources:

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

References:

Linuxmint.com

```c
#include "mystring.h"

// Function to get the length of string
size_t mystrlen(const char *str)
{
    return strlen(str);
}
// Function to get the copy of a string
char* mystrcpy(char* str, const char* str1)
{
    return strcpy(str,str1);
}
// Function to concatinate two strings
char* mystrcat(char *str, const char *str1)
{
    return strcat(str,str1);
}
// Function to compare two strings
int mystrcmp(const char *str, const char *str1)
{
    return strcmp(str,str1);
}
```

```c
#include<string.h>
#ifndef __MYSTRING_H__
#define __MYSTRING_H__
size_t mystrlen(const char *str);
char* mystrcpy(char* str, const char* str1);
char* mystrcat(char *str, const char *str1);
int mystrcmp(const char *str, const char *str1 );
#endif
```

```c
#include "mystring.h"
#include "myutils.h"
#include "bitmask.h"
#include <stdio.h>
//Main Function
int main()
{
  char s[10]="Hello";
  char s1[10]="World";
  int fact=8,palindrome=131,prime=7,num=6,n=2;
  // String Functions
  printf("Length of the string is %lu\n ",mystrlen(s));
  printf("Copy of the strings %s %s is %lu\n ", s, s1, strcpy(s,s1));
  printf("Concatination of the strings %s %s is %lu\n ", s, s1, mystrcat(s,s1));
  printf("Comparison of the strings %s %s is %lu\n ", s, s1, mystrcmp(s,s1));

  // Number Functions
  printf("factorial of %d is %d \n",num,factorial(fact));

  if(ispalindrome(palindrome))
    printf("The number is palindrome\n");
  else
    printf("The number is not palindrome\n");

  if(isPrime(prime))
     printf("The number is prime number\n");
  else
    printf("The number is not a prime number\n");

  vsum(2,10,20);
```

```
TARGET=activity.out
OBJS=test.o src/mystring.o src/myutils.o src/bitmask.o
TOPDIR=${PWD}

run:
	gcc
clean:
	rm -rf ${OBJS} ${TARGET}
```

ACTIVITY 2

Learning Outcomes: Was given activity based on the topics tutored like signals, processes and threads.

We were asked to do a bunch of activities based on these topics.

Challenges: While doing threads and activity 3 on threads

Learning Resources:

https://www.informit.com/articles/article.aspx?p=370047&seqNum=3

GitHub Link: https://github.com/99003553/Activity1/tree/main/Activity%202

ACTIVITY 3

Learning Outcomes: Was given activity based on the topics tutored like IPC, Mutex and Semaphores.

We were asked to do a bunch of activities based on these topics.

Challenges: While doing shared memory and named semaphores

Learning Resources:

https://www.interserver.net/tips/kb/everything-about-linux-semaphore/

GitHub Link: https://github.com/99003553/Activity1/tree/main/Activity3

ACTIVITY 4

Learning Outcomes: Was given activity based on the topics tutored like pipes, shared memory and message queues.
We were asked to do a bunch of activities based on these topics.
Challenges: While doing shared memory.
Learning Resources: https://opensource.com/article/18/8/introduction-pipes-linux
GitHub Link: https://github.com/99003553/Activity1/tree/main/Activity%204