Learning Report – C and Linux OS programming

L&T Technology Services

GLOBAL ENGINEERING ACADEMY

Genesis

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be approved By	Remarks/Revision Details
1		Name/PS No	Name/PS No	Module Owner Name	Comments
2	01/03/21	Himanshu Kumar/99003721			
3					

Table of Contents

INTRODUCTION

This is an attempt to learn and build c code and Linux programming we have implemented many problem statement and worked on some group activity project as well as Individual project.

BASIC COMMANDS FOR LINUX

What Is Linux?

Linux is an operating system's kernel. You might have heard of UNIX. Well, Linux is a UNIX clone. But it was actually created by Linus Torvalds from Scratch. Linux is free and open-source, that means that you can simply change anything in Linux and redistribute it in your own name! There are several Linux Distributions, commonly called “distros”.

- Ubuntu Linux
- Red Hat Enterprise Linux
- Linux Mint
- Debian
- Fedora

Linux Shell or “Terminal”

a shell is a program that receives commands from the user and gives it to the OS to process, and it shows the output. Linux's shell is its main part.

Linux Commands

Basic Commands

```
nayso@Alok-Aspire:~$ ls
Desktop           itsuserguide.desktop  reset-settings     VCD_Copy
Documents         Music                 School_Resources   Videos
Downloads         Pictures              Students_Works_10
examples.desktop  Public                Templates
GplatesProject    Qgis Projects         TuxPaint-Pictures
```

ls — Use the **"ls"** command to know what files are in the directory you are in.

pwd — When you first open the terminal, you are in the home directory of your user.

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

Q1. Write a function to break the given string and return the correct string between '$' and ';' with no '$' or ';' in between.

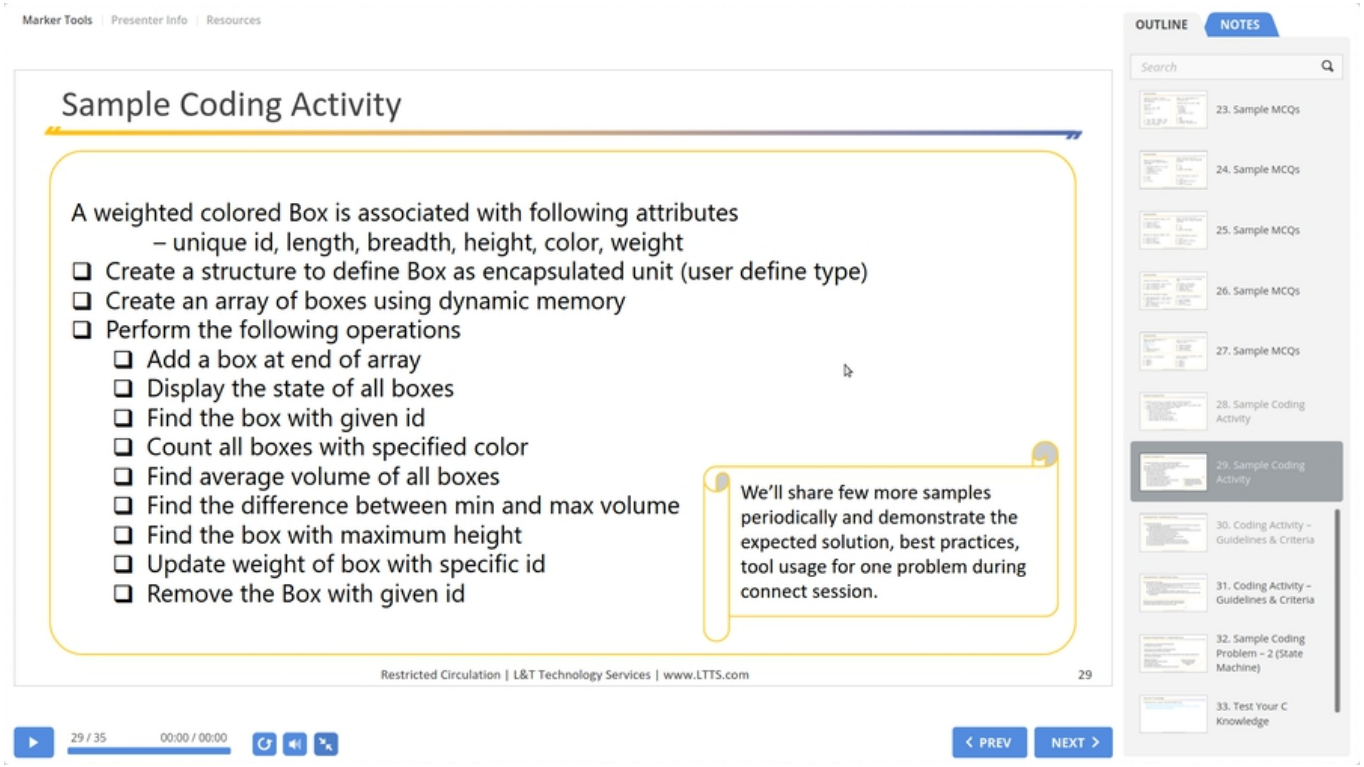
```
#include <stdio.h>
int main() {
   char str_input[1000];
   int start=0,end=0;
   printf("Enter a string_input ");
   fgets(str_input, sizeof(str_input), stdin);

     for (int i = 0; str_input[i] != '\0'; ++i) {
```

```
        if (str_input[i] == '$')
        start=i;

        if (str_input[i] == ';') {
            end=i;
            break;
        }
    }

for (int k = start; k<=end; ++k) {
    printf("%c",str_input[k]);
  }
    return 0;
}
```

The Build Process - C/C++

Sequence of Events : **Editor** => **Type the Code** => **Press Build & Run** => **Wait[Build Process]...** => **Prompt/Window**

Group Coding Activity



Git Link : - https://github.com/99003774/BOX_ACTIVITY.git

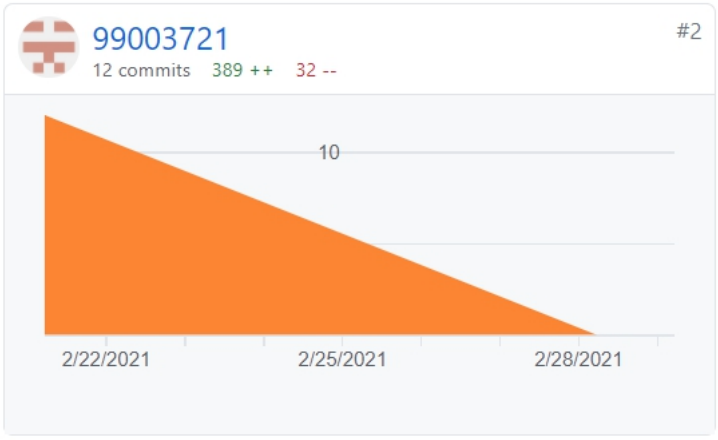
99003774 / BOX_ACTIVITY

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Go to file | Add file | Code

99003763 Update new_codde 942beaf 2 days ago 35 commits

Inc	Add files via upload	4 days ago
src	Update Box_Main.h	3 days ago
test	Add files via upload	4 days ago
unity	Add files via upload	6 days ago
Box_Main.c	Add files via upload	7 days ago
README.md	Update README.md	6 days ago
addArr.c	Add files via upload	5 days ago
box_operation.c	Update box_operation.c	7 days ago
cuboid.c	Add files via upload	6 days ago
memory.c	Create memory.c	7 days ago
new_codde	Update new_codde	2 days ago



Topic we have learned :

1. OS and its features
2. Process and Process Life cycle
3. Types of Kernel
4. Scheduling

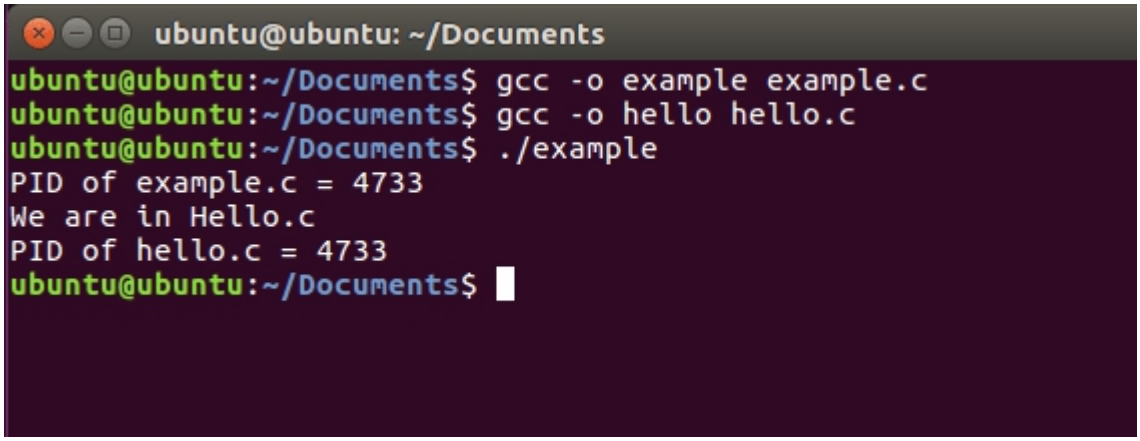
Linux Exec System Call

The exec system call is used to execute a file which is residing in an active process. When exec is called the previous executable file is replaced and new file is executed.
the standard names for these functions are as follows:

1. **execl**
2. **execle**
3. **execlp**
4. **execv**
5. **execve**
6. **execvp**

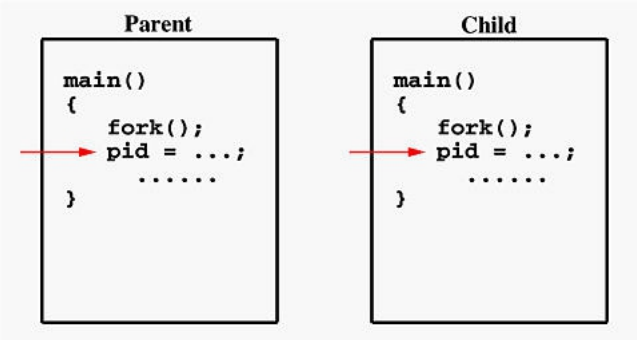
Getpid code Example : -

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
printf("We are in Hello.c\n");
printf("PID of hello.c = %d\n", getpid());
return 0;
}
```



fork() System Call :-

System call **fork()** is used to create processes. It takes no arguments and returns a process ID



Threads and Signals

Thread has its own signal mask, but the signal disposition is shared by all threads in the process.

If the signal is related to a hardware fault or expiring timer, the signal is sent to the thread whose action caused the event.

REFERENCES –

https://www.youtube.com/watch?v=kjvy_zwhBuA
https://www.csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/create.html
https://linuxhint.com/linux-exec-system-call/
https://www.youtube.com/watch?v=83M5-NPDeWs
Inter Process Communication: https://www.youtube.com/watch?v=G2vwkBZy894
Locking (Software Solutions): https://www.youtube.com/watch?v=B_lH2Xov_g4
Semaphores: https://www.youtube.com/watch?v=UM4tk3J6WxQ
Mutex: https://www.youtube.com/watch?v=xKqO04SN6C0