

NAME: SPANDAN MUKHERJEE

REGISTRATION NUMBER: 21BCE1132

SUBJECT: OPERATING SYSTEMS LAB - 2

Questions

1) Simulate CP linux command using C

The screenshot shows a C program named `cp1.c` in a text editor. The program implements a simple copy function using `fopen`, `fgetc`, and `fputc`. It takes two command-line arguments: the source file and the destination file. The program is then executed in a terminal window, demonstrating its functionality by copying `a.txt` to `b.txt`.

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     FILE *fp1, *fp2;
6     fp1 = fopen(argv[1], "r");
7     fp2 = fopen(argv[2], "w");
8
9
10    int c;
11    while((c = fgetc(fp1)) != EOF)
12    {
13        fputc(c, fp2);
14    }
15    fclose(fp1);
16    fclose(fp2);
17
18    return 0;
19 }
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp1.c
^C
spandan@spandan-VirtualBox:~$ gcc cp1.c
spandan@spandan-VirtualBox:~$ cat > a.txt
769
898
90
23
900
^Z
[1]+  Stopped                  cat > a.txt
spandan@spandan-VirtualBox:~$ ./a.out a.txt b.txt
spandan@spandan-VirtualBox:~$ cat b.txt
769
898
90
23
900
spandan@spandan-VirtualBox:~$ gedit cp1.c
```

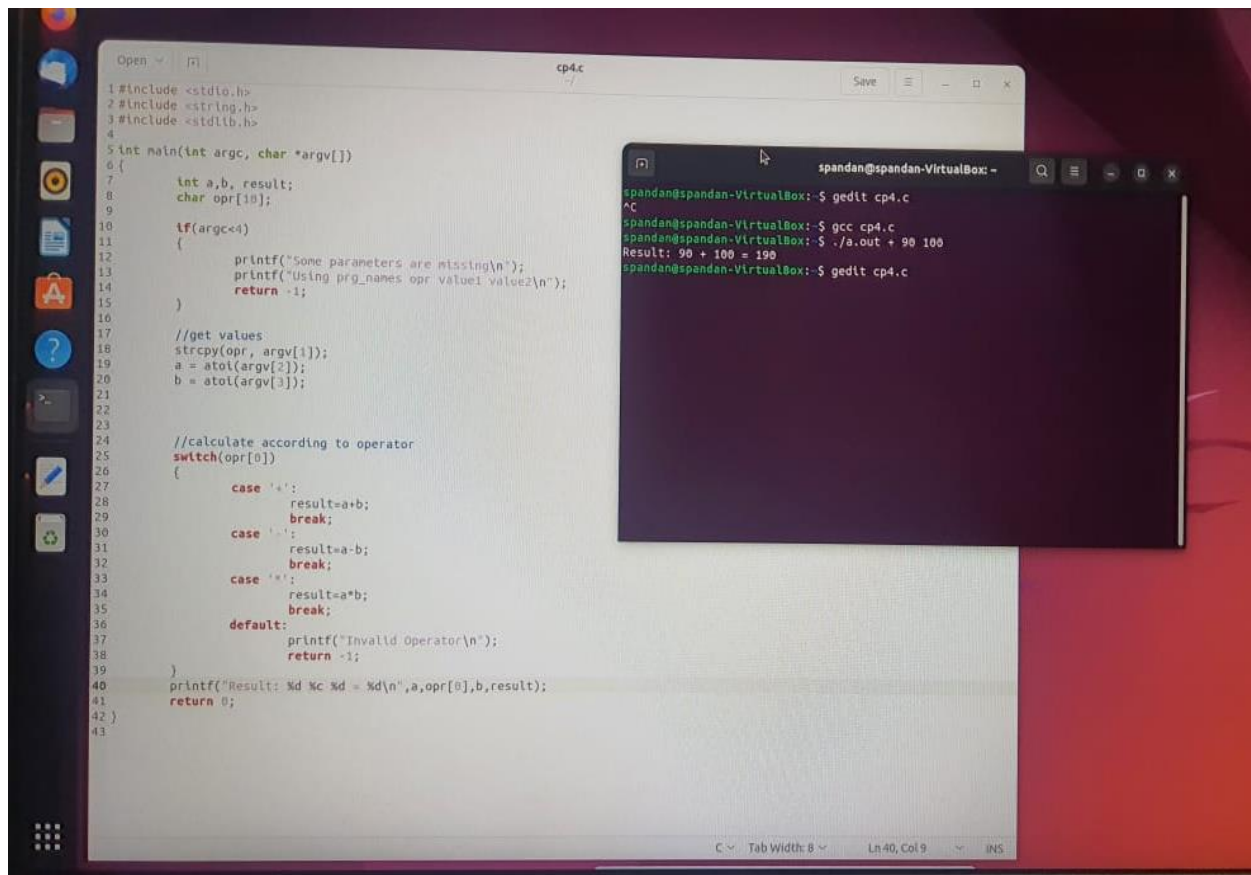
2) Simulate MV linux command using C

The screenshot shows a C program named `cp7.c` in a text editor. The program implements a simple move function using `fopen`, `fgetc`, `fputc`, and `remove`. It takes two command-line arguments: the source file and the destination file. The program is then executed in a terminal window, demonstrating its functionality by moving `c.txt` to `d.txt`.

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <unistd.h>
4
5 int main(int argc, char *argv[])
6 {
7     FILE *fp = fopen(argv[1], "r");
8     FILE *fp1 = fopen(argv[2], "w");
9
10    int c;
11
12    char *p = "/fp";
13
14    while((c=fgetc(fp)) != EOF)
15    {
16        fputc(c, fp1);
17        int ret = remove(p);
18
19        if(ret==0){
20            printf("File deleted");
21        }
22    }
23    fclose(fp);
24    fclose(fp1);
25
26    return 0;
27 }
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp7.c
^C
spandan@spandan-VirtualBox:~$ gcc cp7.c
spandan@spandan-VirtualBox:~$ ./a.out d.txt c.txt
Segmentation fault (core dumped)
spandan@spandan-VirtualBox:~$ cat > d.txt
^Z
[1]+  Stopped                  cat > d.txt
spandan@spandan-VirtualBox:~$ gedit cp7.c
^C
spandan@spandan-VirtualBox:~$ gcc cp7.c
spandan@spandan-VirtualBox:~$ cat d.txt
spandan@spandan-VirtualBox:~$ gedit cp7.c
```

3) Perform arithmetic operations using command line arguments



The image shows a Linux desktop environment with a dark theme. On the left is a vertical dock with various application icons. The main workspace contains two windows. The background window is a code editor titled 'cp4.c' showing the source code of a C program. The program takes command-line arguments and performs arithmetic operations based on the operator provided. The foreground window is a terminal titled 'spandan@spandan-VirtualBox: ~' showing the execution of the program. The terminal output shows the program being compiled with 'gcc cp4.c', the resulting binary being run with './a.out + 90 100', and the output 'Result: 90 + 100 = 190'.

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 int main(int argc, char *argv[])
6 {
7     int a, b, result;
8     char opr[10];
9
10    if(argc < 4)
11    {
12        printf("Some parameters are missing\n");
13        printf("Using prg_names opr value1 value2\n");
14        return -1;
15    }
16
17    //get values
18    strcpy(opr, argv[1]);
19    a = atoi(argv[2]);
20    b = atoi(argv[3]);
21
22
23    //calculate according to operator
24    switch(opr[0])
25    {
26        case '+':
27            result = a + b;
28            break;
29        case '-':
30            result = a - b;
31            break;
32        case '*':
33            result = a * b;
34            break;
35        default:
36            printf("Invalid Operator\n");
37            return -1;
38    }
39
40    printf("Result: %d %c %d = %d\n", a, opr[0], b, result);
41    return 0;
42 }
43
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox: ~$ gcc cp4.c
spandan@spandan-VirtualBox: ~$ ./a.out + 90 100
Result: 90 + 100 = 190
spandan@spandan-VirtualBox: ~$
```

- 4) Check whether the given string is palindrome or not and ensure to take the input while executing the program

The screenshot shows a Linux desktop environment with a purple and red geometric wallpaper. On the left is a sidebar with application icons. Two windows are open:

- cp5.c**: A text editor window showing the source code of a C program. The code is as follows:

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char string1[20];
7     int l, length;
8
9     int flag=0;
10
11     printf("Enter a string:");
12     scanf("%s",string1);
13
14     length = strlen(string1);
15
16     for(i=0; i<length;i++)
17     {
18         if(string1[i] != string1[length-i-1])
19         {
20             flag = 1;
21             break;
22         }
23     }
24     if(flag){
25         printf("%s is a not a palindrome",string1);
26     }
27     else{
28         printf("%s is a palindrome",string1);
29     }
30     return 0;
31 }
32 }
```
- Terminal**: A terminal window titled "spandan@spandan-VirtualBox: ~" showing the execution of the program. The commands and output are:

```
spandan@spandan-VirtualBox:~$ gcc cp5.c
spandan@spandan-VirtualBox:~$ ./a.out
Enter a string:abaaba
abaaba is a palindromespandan@spandan-VirtualBox:~$ gcc cp5.c
```

- 5) Create 3 child processes from the same parent process and show the child processes are created from the same parent process.

The screenshot shows the same Linux desktop environment. Two windows are open:

- cp6.c**: A text editor window showing the source code of a C program. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main()
6 {
7     int pld, pld1, pld2;
8
9     pld = fork();
10
11     if (pld == 0) {
12         sleep(3);
13         printf("child[1] --> pld = %d and ppid = %d\n",
14             getpid(), getppid());
15     }
16     else {
17         pld1 = fork();
18         if (pld1 == 0) {
19             sleep(2);
20             printf("child[2] --> pld = %d and ppid = %d\n",
21                 getpid(), getppid());
22         }
23         else {
24             pld2 = fork();
25             if (pld2 == 0) {
26                 printf("child[3] --> pld = %d and ppid = %d\n",
27                     getpid(), getppid());
28             }
29             else {
30                 sleep(1);
31                 printf("parent --> pld = %d\n", getpid());
32             }
33         }
34     }
35     return 0;
36 }
```
- Terminal**: A terminal window titled "spandan@spandan-VirtualBox: ~" showing the execution of the program. The commands and output are:

```
spandan@spandan-VirtualBox:~$ gcc cp6.c
spandan@spandan-VirtualBox:~$ ./a.out
child[3] --> pld = 10879 and ppid = 10876
child[2] --> pld = 10878 and ppid = 10876
parent --> pld = 10876
child[1] --> pld = 10877 and ppid = 813
spandan@spandan-VirtualBox:~$ gcc cp6.c
```

6) Discuss the use of Command line arguments in C

In C programming, command line arguments are an important concept. Using command line parameters, we can perform any task. It is mostly used when you need to control your program from outside.

Before we go any further, let us define certain terms that will be used in this article, such as Command-Line and Command-Line Arguments.

The command line is a text interface for your computer that allows you to enter commands for immediate execution. A command-line can perform almost everything that a graphical user interface can. Many tasks can be performed more rapidly and are easier to automate.

For example:

- You can navigate around your computer's files and directories using the command line.
- The command line can be scripted to automate complex tasks, like the example given below:

If a user wants to put 50+ files' data into a file, this is a highly time-consuming task. Copying data from 50+ files, on the other hand, can be done in less than a minute with a single command at the command line. And many more.