

In the first part of Q1, we are asked to use `fork()` to create two processes. In this case, the memory space is not shared by the processes.

That means that the global variable is different for the two processes. So, the change in the global variable by one process is not reflected in the other.

Hence when the processes are running concurrently, their output is also displayed together and the value of one global variable keeps on decreasing from 10 to -90 whereas the value of the other keeps on increasing to 100. we used `wait()` to ensure that the child process is finished before the parent.

In the second part of Q1, we are asked to use `pthread_create()` to create threads. In this case, since the threads are not independent of each other,

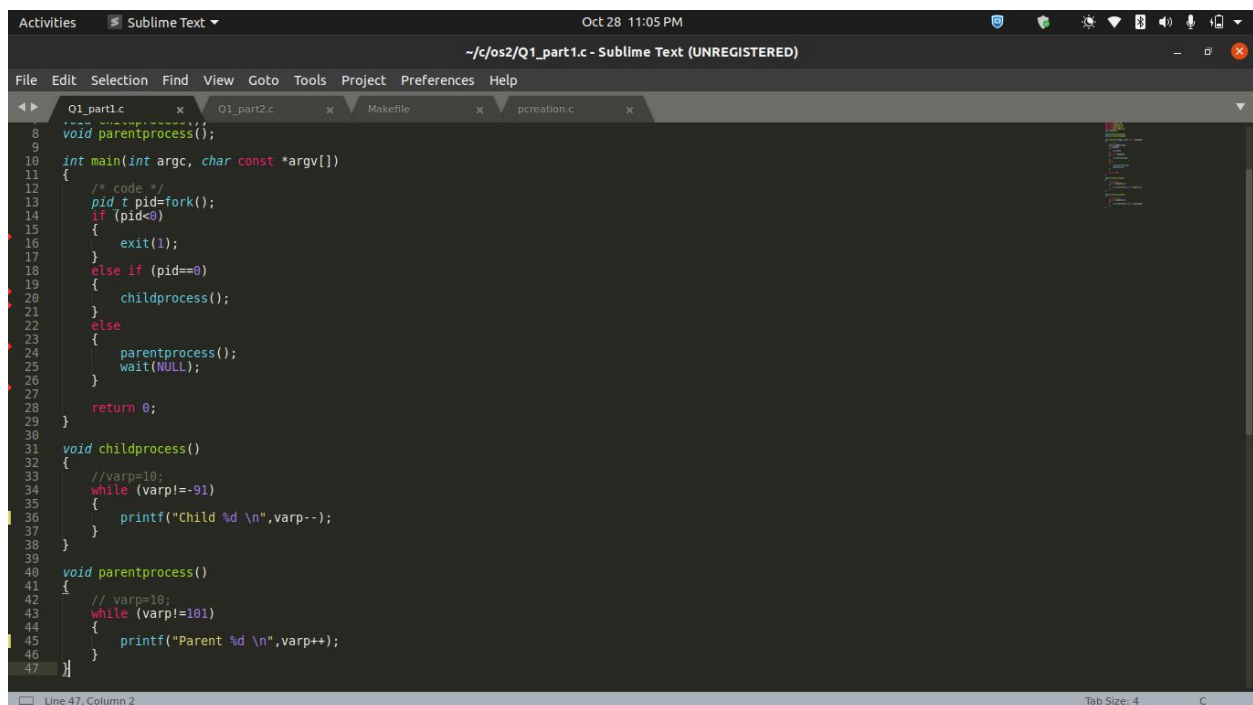
the memory space and code section is shared by the two threads. Here, the change in the global variable by one thread is reflected in the other.

So, if one thread changes the value of the global variable, it is changed for the other thread as well. Hence we ran the threads sequentially but waited for the child thread to end first.

Differences in the output: For the first case, since the variables are independent, the value of the child global variable decreases to -90, and the parent increases to 100.

For the second case, since the variables are dependent, the child process first decreases the value of global variable from 10 to -90, then, the parent process increases the value from -90(since the global variable is the same) to 100.

Q1 Output:

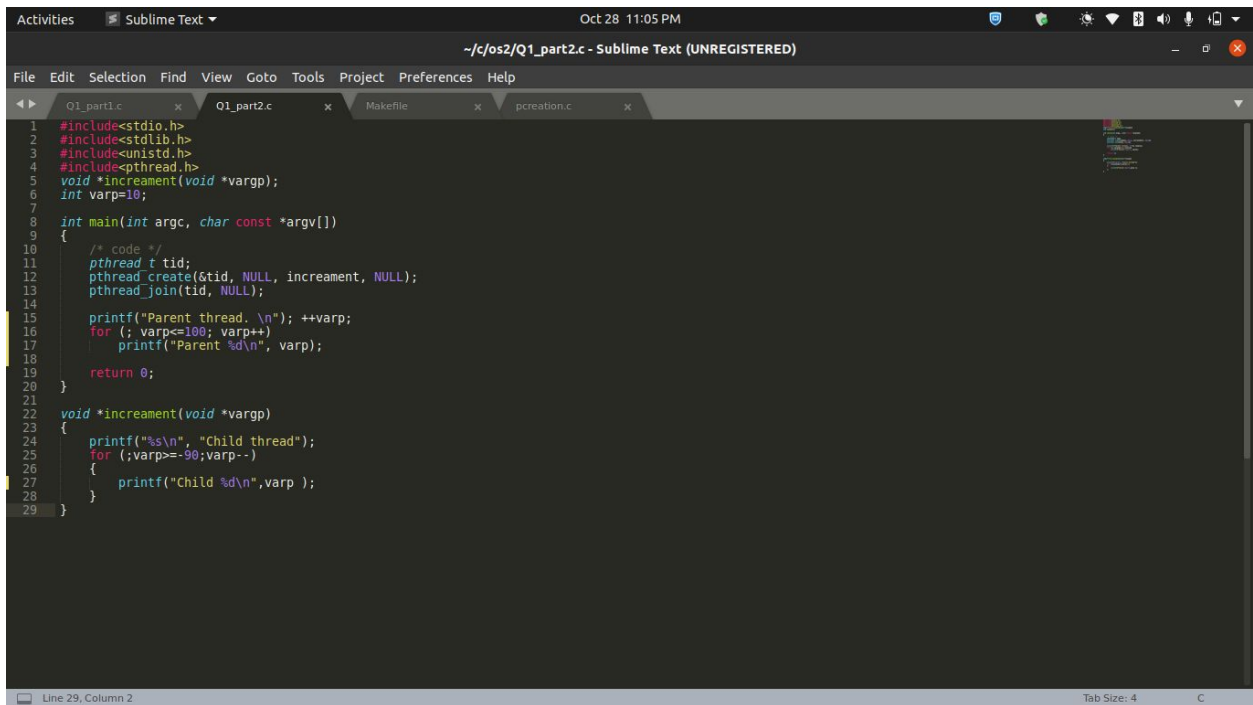


```
Oct 28 11:05 PM
~/c/os2/Q1_part1.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Q1_part1.c x Q1_part2.c x Makefile x pcreation.c x
8 //varp=10;
9 void parentprocess();
10 int main(int argc, char const *argv[])
11 {
12     /* code */
13     pid_t pid=fork();
14     if (pid<0)
15     {
16         exit(1);
17     }
18     else if (pid==0)
19     {
20         childprocess();
21     }
22     else
23     {
24         parentprocess();
25         wait(NULL);
26     }
27     return 0;
28 }
29
30 void childprocess()
31 {
32     //varp=10;
33     while (varp!=-91)
34     {
35         printf("Child %d \n",varp--);
36     }
37 }
38
39 void parentprocess()
40 {
41     // varp=10;
42     while (varp!=101)
43     {
44         printf("Parent %d \n",varp++);
45     }
46 }
47 }
```

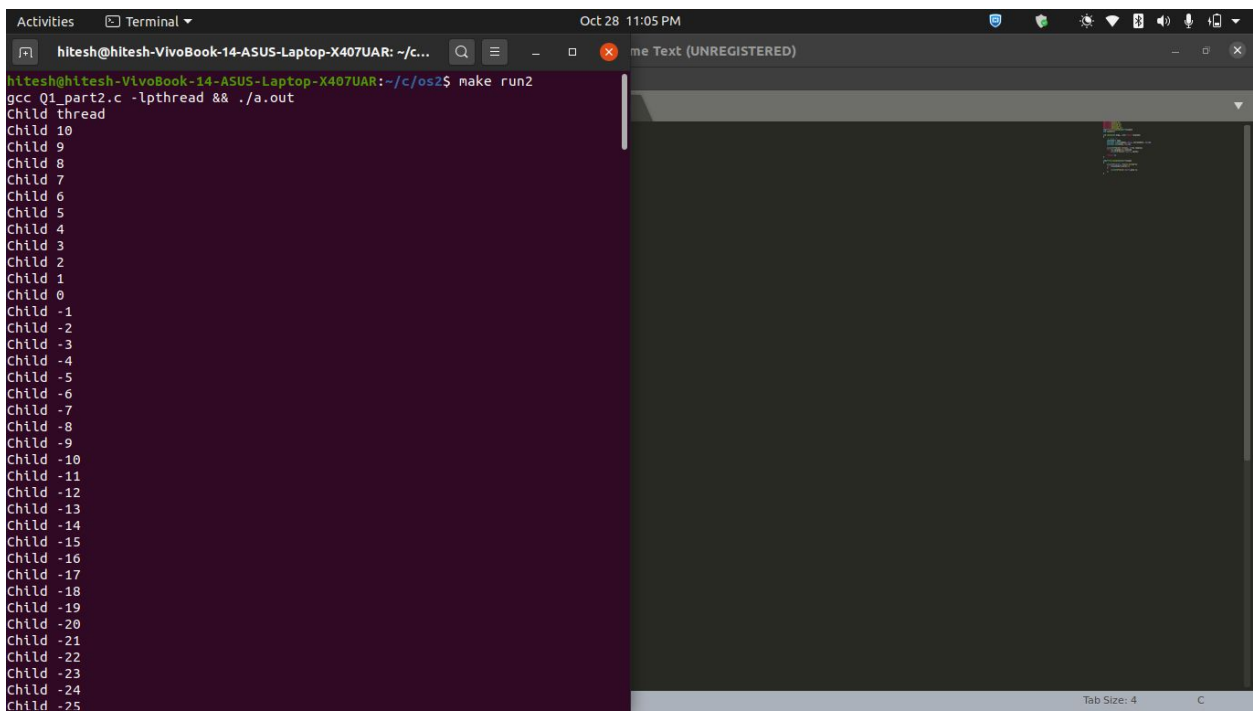
```
Activities Terminal Oct 28 11:05 PM
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR: ~/c... me Text (UNREGISTERED)
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR:~/c/os2$ make run1
gcc Q1_part1.c && ./a.out
Parent 10
Parent 11
Parent 12
Parent 13
Parent 14
Parent 15
Parent 16
Parent 17
Parent 18
Parent 19
Parent 20
Parent 21
Parent 22
Parent 23
Parent 24
Parent 25
Parent 26
Parent 27
Parent 28
Parent 29
Parent 30
Parent 31
Parent 32
Parent 33
Parent 34
Parent 35
Parent 36
Parent 37
Parent 38
Parent 39
Parent 40
Parent 41
Parent 42
Parent 43
Parent 44
Parent 45
Parent 46
```

```
Activities Terminal Oct 28 11:05 PM
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR: ~/c... me Text (UNREGISTERED)
Child -54
Child -55
Child -56
Child -57
Child -58
Child -59
Child -60
Child -61
Child -62
Child -63
Child -64
Child -65
Child -66
Child -67
Child -68
Child -69
Child -70
Child -71
Child -72
Child -73
Child -74
Child -75
Child -76
Child -77
Child -78
Child -79
Child -80
Child -81
Child -82
Child -83
Child -84
Child -85
Child -86
Child -87
Child -88
Child -89
Child -90
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR:~/c/os2$
```

Q2 Output:



```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<unistd.h>
4 #include<pthread.h>
5 void *increment(void *vargp);
6 int varp=10;
7
8 int main(int argc, char const *argv[])
9 {
10     /* code */
11     pthread_t tid;
12     pthread_create(&tid, NULL, increment, NULL);
13     pthread_join(tid, NULL);
14
15     printf("Parent thread. \n"); ++varp;
16     for (; varp<=100; varp++)
17         printf("Parent %d\n", varp);
18
19     return 0;
20 }
21
22 void *increment(void *vargp)
23 {
24     printf("%s\n", "Child thread");
25     for (; varp>=-90; varp--)
26     {
27         printf("Child %d\n", varp );
28     }
29 }
```



```
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR: ~/c...
hitesh@hitesh-VivoBook-14-ASUS-Laptop-X407UAR:~/c/os2$ make run2
gcc Q1_part2.c -lpthread && ./a.out
Child thread
Child 10
Child 9
Child 8
Child 7
Child 6
Child 5
Child 4
Child 3
Child 2
Child 1
Child 0
Child -1
Child -2
Child -3
Child -4
Child -5
Child -6
Child -7
Child -8
Child -9
Child -10
Child -11
Child -12
Child -13
Child -14
Child -15
Child -16
Child -17
Child -18
Child -19
Child -20
Child -21
Child -22
Child -23
Child -24
Child -25
```

