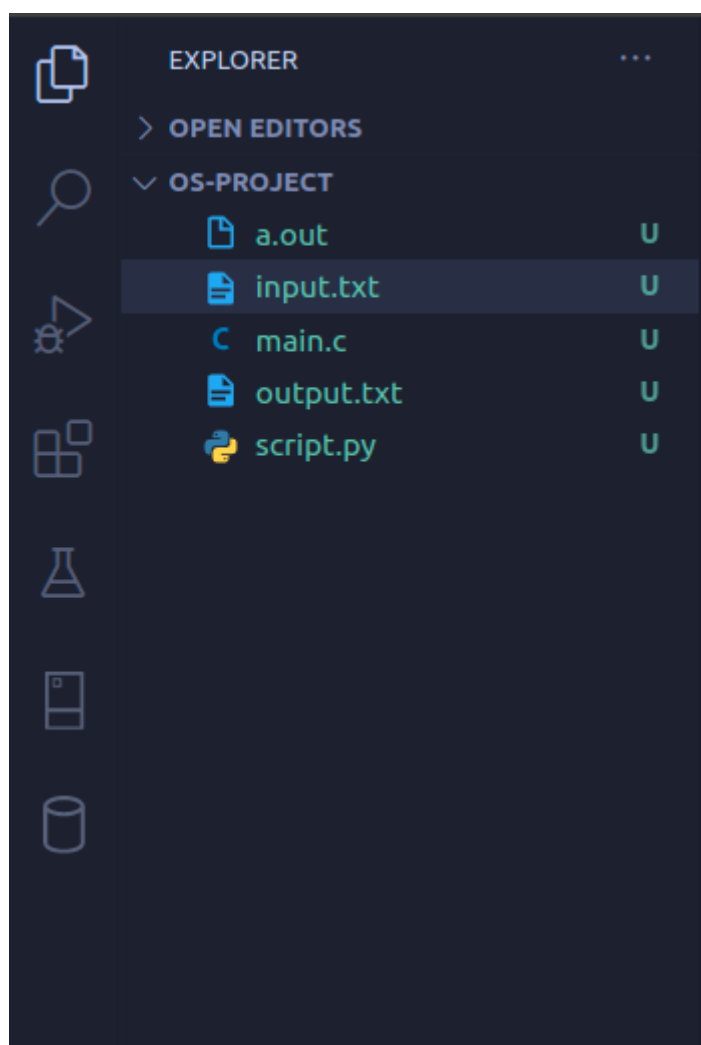


پروژه سیستم عامل  
دکتر خانمیرزا

علی الهی

توضیحات فایل های موجود در پروژه



## فایل اصلی برنامه :

```
C main.c X input.txt output.txt script.py
C main.c > main()
6 #include <sys/wait.h>
7 #include <sys/time.h>
8
9 int main()
10 {
11     struct timeval tv1, tv2;
12     gettimeofday(&tv1, NULL);
13
14     FILE *file = fopen("./input.txt", "r");
15     int n = 0, m = 0;
16     fscanf(file, "%d", &m);
17     fscanf(file, "%d", &n);
18
19     int fd1[2];
20     pipe(fd1);
21
22     int fd2[2];
23     pipe(fd2);
24     pid_t main_id = getpid();
25
26     for (int i = 0; i < m; i++)
27     {
28         int level1 = fork();
29
30         if (level1 == 0)
31         {
32             pid_t main_id2 = getpid();
33
34             for (int j = 0; j < n; j++)
35             {
36                 int level2 = fork();
37                 if (level2 == 0)
38                 {
39                     int holder;
40                     read(fd2[0], &holder, sizeof holder);
41                     usleep(holder * 1000);
42                     exit(0);
43                 }
44                 else
45                 {
46                     int holder;
47                     read(fd1[0], &holder, sizeof holder);
48                     write(fd2[1], &holder, sizeof holder);
49                 }
50             }
51         }
52         if (getpid() == main_id2)
53     }
```

```
C main.c X input.txt output.txt script.py
C main.c > main()
53     if (getpid() == main_id2)
54     {
55         for (int j = 0; j < n; j++)
56         {
57             wait(NULL);
58         }
59         exit(0);
60     }
61     else
62     {
63         for (int j = 0; j < n; j++)
64         {
65             int holder = 0;
66             fscanf(file, "%d", &holder);
67             write(fd1[1], &holder, sizeof holder);
68         }
69     }
70 }
71
72 if (getpid() == main_id)
73 {
74     for (int i = 0; i < m; i++)
75     {
76         wait(NULL);
77     }
78 }
79
80 gettimeofday(&tv2, NULL);
81
82 printf("=====\n");
83 printf("\nConfigure : %d and %d", m, n);
84 printf("\nTotal time = %f seconds\n",
85        (double)(tv2.tv_usec - tv1.tv_usec) / 1000 +
86        (double)(tv2.tv_sec - tv1.tv_sec));
87
88 FILE *fp;
89
90 fp = fopen("./output.txt", "a+");
91 fprintf(fp, "%f\n",
92        (double)(tv2.tv_usec - tv1.tv_usec) / 1000 +
93        (double)(tv2.tv_sec - tv1.tv_sec));
94
95 return 0;
96 }
97
98 }
```

فایل **input** که در آن کانفیگ برنامه و sleep Time های هرکدام وجود دارد :



The image shows a code editor window with a dark theme. The title bar at the top reads "input.txt". The editor contains the following text:

```
1 2 3
2 2
3 10
4 4
5 10
6 6
7 7
```

The status bar at the bottom of the editor displays the following information: "Ln 1, Col 4", "Spaces: 4", "UTF-8", "LF", "Plain Text", "Go Live", and a search icon.

## فایل اسکریپت پایتون :

وظیفه این فایل اجرای کامند های gcc main.c و ./a.out برای 20 بار پشت سر هم است تا بتوان میانگین زمان ها را حساب کرد.

```
script.py > ...
1 import subprocess
2 for i in range(20):
3     status = subprocess.call(["gcc","main.c"])
4     if status == 0:
5         print(subprocess.check_output("./a.out").decode('utf-8'))
6     else:
7         print("Failed to compile")
```

Ln 7, Col 35 Spaces: 4 UTF-8 LF Python Go Live

## فایل output.txt که زمان اجرا را ذخیره میکند :

```
output.txt
1 16.465000,
2 10.571000,
3 11.403000,
4 11.363000,
5 10.585000,
6 10.575000,
7 10.699000,
8 10.604000,
9 10.621000,
10 11.842000,
11 17.058000,
12 11.389000,
13 11.808000,
14 10.536000,
15 11.856000,
16 10.674000,
17 11.359000,
18 10.744000,
19 10.557000,
20 11.746000,
21 10.698000,
22 11.054000,
23 11.816000,
24 10.623000,
25 11.852000,
26 10.647000,
27 15.816000,
28 11.395000,
29 10.589000,
30 15.890000,
31 11.401000,
32 10.574000,
33 10.608000,
34 10.639000,
35 11.371000,
36 10.710000,
37 11.296000,
38 18.356000,
39 12.614000,
40 10.571000,
```

## مثال 1 :

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE 1: bash
ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 1 and 6
Total time = 11.360000 seconds
ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 2 and 3
Total time = 11.842000 seconds
ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 3 and 2
Total time = 12.614000 seconds
ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ gcc main.c
ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 6 and 1
Total time = 16.434000 seconds
ali@ali:~/Desktop/os-project$
```

همانطور که مشخص است زمان اجرای کانفیگ ها به ترتیب :

$$[1,6] < [2,3] < [3,2] < [6,1]$$

است.

هرچه level اول عدد کوچکتری داشته باشد زمان اجرا نیز کمتر خواهد بود.

## مثال 2 :

```
ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 1 and 8
Total time = 12.483000 seconds

ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 2 and 4
Total time = 13.645000 seconds

ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ gcc main.c
ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 4 and 2
Total time = 16.631000 seconds

ali@ali:~/Desktop/os-project$

ali@ali:~/Desktop/os-project$ ./a.out
=====
Configure : 8 and 1
Total time = 17.846000 seconds

ali@ali:~/Desktop/os-project$
```

همانطور که مشخص است زمان اجرای کانفیگ ها به ترتیب :

$$[1,8] < [2,4] < [4,2] < [8,1]$$

است.

مانند مثال قبل هر چه level اول عدد کوچکتری داشته باشد زمان اجرا نیز کمتر خواهد بود.

### نکته:

زمان های اجرا به مشخصات فنی سیستمی که برنامه روی آن اجرا میشود بستگی دارد.  
و نزدیک بودن زمان های اجرای کانفیگ های مختلف به علت تعداد core های لپتاپ میباشد.

### نمودار مثال اول :



### نمودار مثال دوم :

