



دانشگاه علم و صنعت

دانشکده مهندسی کامپیوتر

درجه تحصیلی: کارشناسی

## گزارشکار تکلیف OS 2

گردآورنده:

پرnian شاكریان - 99400064

استاد:

دکتر انتظاری

سال تحصیلی: اسفند ۱۴۰۱

خلاصه:

در تکلیف دوم سیستم عامل قصد داریم از Thread و Process در حل تمارین استفاده کنیم. اگر بخواهیم به طور خلاصه در مورد ماهیت کدها توضیح دهیم Process، پردازش مجموعه‌ای از کدها، حافظه، داده و سایر منابع است. Thread توالی از کدها به حساب می‌آید که در داخل محدوده یک پردازش اجرا می‌شود. هر رشته می‌تواند بخشی از یک وظیفه را اجرا کند (به عنوان مثال، جمع کردن عناصر یک آرایه)، یا می‌تواند همان کار را برای client مختلف در client-server اختصاص دهد.

شرح کلی:

۱. اضافه کردن کتابخانه‌ها (Include the header file)
۲. هر رشته دارای یک شی از نوع pthread\_t مرتبط با آن است که شناسه آن را می‌گوید. (نمی‌تواند توسط چندین رشته به طور همزمان استفاده شود)
۳. یک رشته ایجاد می‌شود و با استفاده از تابع pthread\_create() شروع می‌شود و نیازمند چهار پارامتر است (id, Attributes, Starting routine, Arguments)
۴. pthread\_exit () برای خروج از thread استفاده می‌شود. (معمولاً در پایان روتین شروع نوشته می‌شود). اگر مقداری پس از پایان توسط یک رشته برگردانده شود، مرجع آن به عنوان یک آرگومان ارسال می‌شود.
۵. parent با استفاده از pthread\_join() منتظر یک child می‌شود. دو پارامتر این تابع عبارتند از Thread ID و Reference to return value

نکته: (نوع برگشتی شروع و آرگومان آن معمولاً بر روی void\* تنظیم می‌شود).

تمام کتابخانه ها و headerهای مورد نیاز ما برای تمرین ۲ به طور کلی و برای آسانی کار اینجا نوشته و اشاره شده است.

```
#include <assert.h>
```

```
#include <stdio.h>
```

```
#include <pthread.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <sys/wait.h>
```

```
#include <string.h>
```

```
#include "stdlib.h"
```

```
#include "stdio.h"
```

```
#include "unistd.h"
```

```
#include "string.h"
```

```
#include "pthread.h"
```

برای ساخت makefile ها از لینک های زیر کمک گرفته ام.

<https://youtu.be/bPMDEdjKBWA>

[https://youtu.be/\\_r7i5X0rXJk](https://youtu.be/_r7i5X0rXJk)

سوال ۲: برنامه ای بنویسید که با استفاده از N تا Thread در N فایل مختلف ۱۰۰ عدد رندوم جنریت و رایت کند و سپس با استفاده از N تا Thread دیگر اون  $N*100$  عدد را بخواند و میانگین آن ها را محاسبه کرده و چاپ کند.

در ابتدا کتابخانه ها و headerهای مورد نیاز خود را include خواهیم کرد. (کتابخانه های مورد نیاز ما برای این تمرین در اول داک نوشته شده است) در حالت کلی این برنامه چهار فایل حاوی 100 تصادفی تولید می کند (تعداد اعداد را میتوانیم خودمان تعیین کنیم) و سپس میانگین آن اعداد را محاسبه می کند.

در ابتدا ساختاری به نام custom\_data را تعریف می کنیم که دارای دو متغیر thread\_num (یک عدد صحیح) و thread\_avg است. متغیر thread\_num برای شناسایی فایلی که thread روی آن کار می کند و متغیر thread\_avg برای ذخیره میانگین اعداد در آن فایل استفاده می شود. تابع main برخی از متغیرها را مقداردهی اولیه می کند. با ایجاد (gen\_threads) برای تولید فایل های تصادفی شروع می شود. سپس منتظر می ماند تا با استفاده از تابع pthread\_join() تولید را به پایان برسانند. پس از تکمیل رشته جدید (calc\_threads) برای محاسبه میانگین فایل های تولید شده ایجاد می شود. آرایه ای از ساختارهای calc\_thread\_args ایجاد می کند تا تعداد رشته و میانگینی را که هر رشته محاسباتی محاسبه می کند ذخیره کند. هر رشته یک آرگومان عدد صحیح (thread\_num) می گیرد که مشخص می کند روی کدام فایل باید عمل کند. سپس ۱۰۰ عدد تصادفی (Num\_lines) بین ۱ تا ۱۰۰۰۰۰ تولید می کند، آنها را در فایل نوشته و خارج می شود. هر رشته محاسباتی یک آرگومان ساختار custom\_data را می گیرد که مشخص می کند روی کدام فایل باید عمل کند سپس فایل را باز کرده، اعداد را می خواند و آنها را جمع می کند، میانگین را محاسبه می کند و آن را در متغیر thread\_avg ساختار custom\_data ذخیره می کند.

پس از تکمیل همه محاسبات، تابع main میانگین های هر رشته را جمع می کند و بر تعداد رشته ها تقسیم می کند تا میانگین نهایی همه فایل های تولید شده را بدست آورد. این میانگین روی کنسول چاپ می شود. در نهایت، برنامه خارج می شود.

به طور خلاصه برنامه از ۳ قسمت اصلی تشکیل شده است:

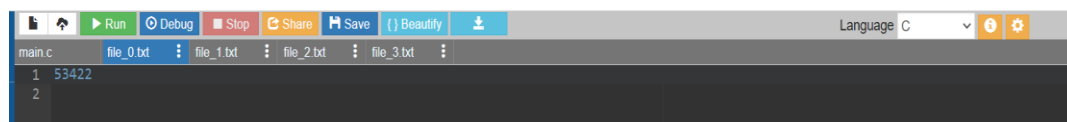
۱. قسمت اول تابعی به نام `gene_random_files()` را برای تولید آرایه ای از اعداد تصادفی تعریف می کند و آنها را بر اساس شماره رشته در فایلی به نام `file_{thread_num}.txt` می نویسد.

۲. در قسمت دوم تابع `calault_average()` را تعریف می کنیم که اعداد هر فایلی را که قبلاً ایجاد شده بود بر اساس شماره رشته خوانده و میانگین آنها را محاسبه می کند. پارامترهای لازم برای محاسبه میانگین را در ساختار `"custom_data"` ذخیره می کند.

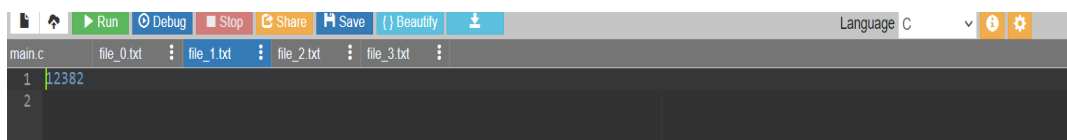
۳. قسمت آخر عملکرد اصلی برنامه است. دو آرایه رشته ایجاد می کند، یکی برای تولید اعداد تصادفی و نوشتن آنها در فایل ها و دومی برای محاسبه میانگین اعداد برای هر فایل. سپس منتظر می ماند تا همه رشته ها کار خود را به پایان برسانند، میانگین نهایی هر چهار رشته را محاسبه می کند و آن را در کنسول چاپ می کند.

وقتی برنامه را ران کنیم چنین خروجی را به ما خواهد داد:

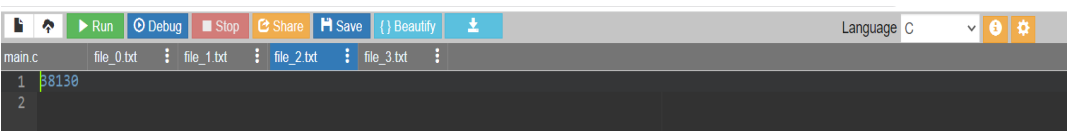
(توجه برای راحتی تست `num` را به جای 100, 1 گذاشته ام)



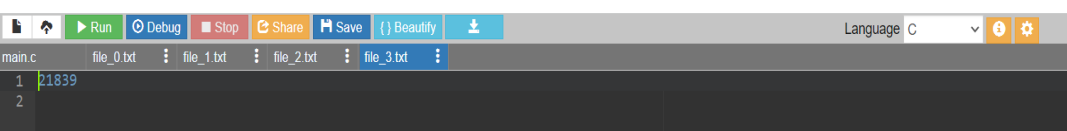
```
main.c | file_0.txt | file_1.txt | file_2.txt | file_3.txt |
1 53422
2
```



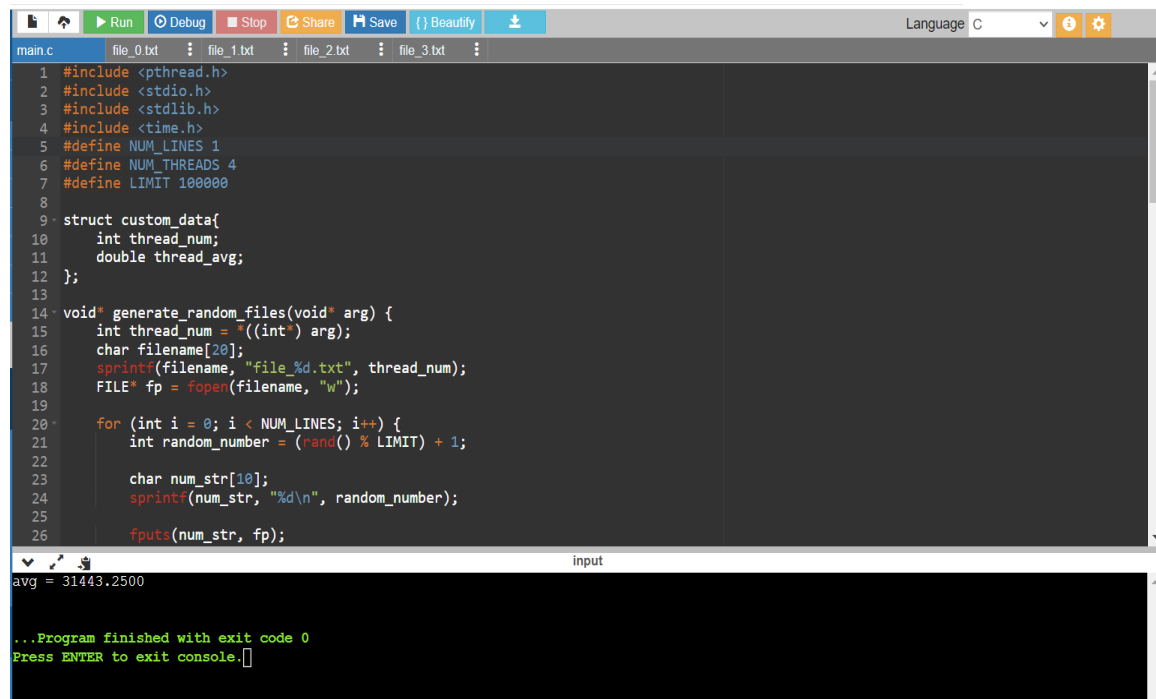
```
main.c | file_0.txt | file_1.txt | file_2.txt | file_3.txt |
1 12382
2
```



```
main.c | file_0.txt | file_1.txt | file_2.txt | file_3.txt |
1 8130
2
```



```
main.c | file_0.txt | file_1.txt | file_2.txt | file_3.txt |
1 21839
2
```



```
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5 #define NUM_LINES 1
6 #define NUM_THREADS 4
7 #define LIMIT 100000
8
9 struct custom_data{
10     int thread_num;
11     double thread_avg;
12 };
13
14 void* generate_random_files(void* arg) {
15     int thread_num = *((int*) arg);
16     char filename[20];
17     sprintf(filename, "file_%d.txt", thread_num);
18     FILE* fp = fopen(filename, "w");
19
20     for (int i = 0; i < NUM_LINES; i++) {
21         int random_number = (rand() % LIMIT) + 1;
22
23         char num_str[10];
24         sprintf(num_str, "%d\n", random_number);
25
26         fputs(num_str, fp);
27     }
28 }
29
30 int main() {
31     pthread_t threads[NUM_THREADS];
32     struct custom_data data[NUM_THREADS];
33
34     for (int i = 0; i < NUM_THREADS; i++) {
35         data[i].thread_num = i;
36         pthread_create(&threads[i], NULL, generate_random_files, &data[i]);
37     }
38
39     for (int i = 0; i < NUM_THREADS; i++) {
40         pthread_join(threads[i], NULL);
41         printf("Thread %d finished\n", data[i].thread_num);
42     }
43
44     double avg = 0;
45     for (int i = 0; i < NUM_THREADS; i++) {
46         avg += data[i].thread_avg;
47     }
48     avg /= NUM_THREADS;
49     printf("avg = %f\n", avg);
50
51     return 0;
52 }
```

avg = 31443.2500

...Program finished with exit code 0  
Press ENTER to exit console.

اگر اعداد داخل هر فایل رو دستی جمع کرده و سپس میانگین بگیریم جواب  
همان ۳۱۴۴۳.۲۵ avg که خودش حساب میکند میشود.