گزارشِ بررسیِ عملکرد و عیب‌یابیِ کد ارائه‌شده در تمرینِ pThread

تهیه و تنظیم: مبین خیری

شماره دانشجویی: 994421017

استاد راهنما: دکتر لیلا شریفی

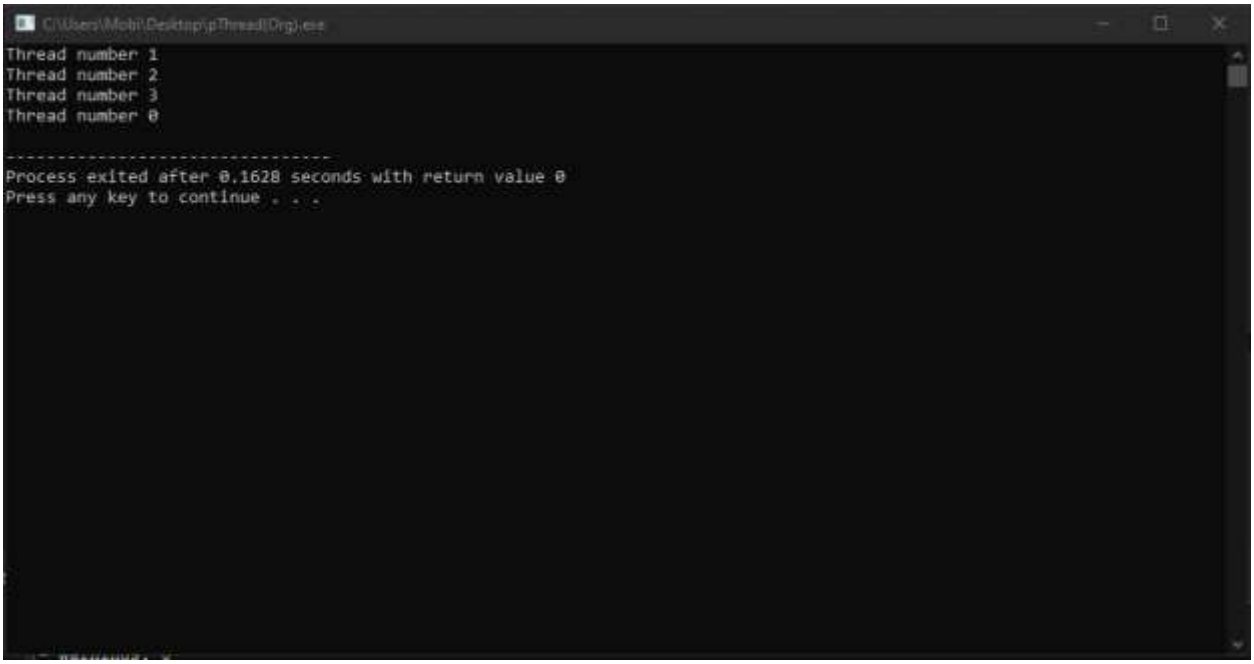
چکیده:

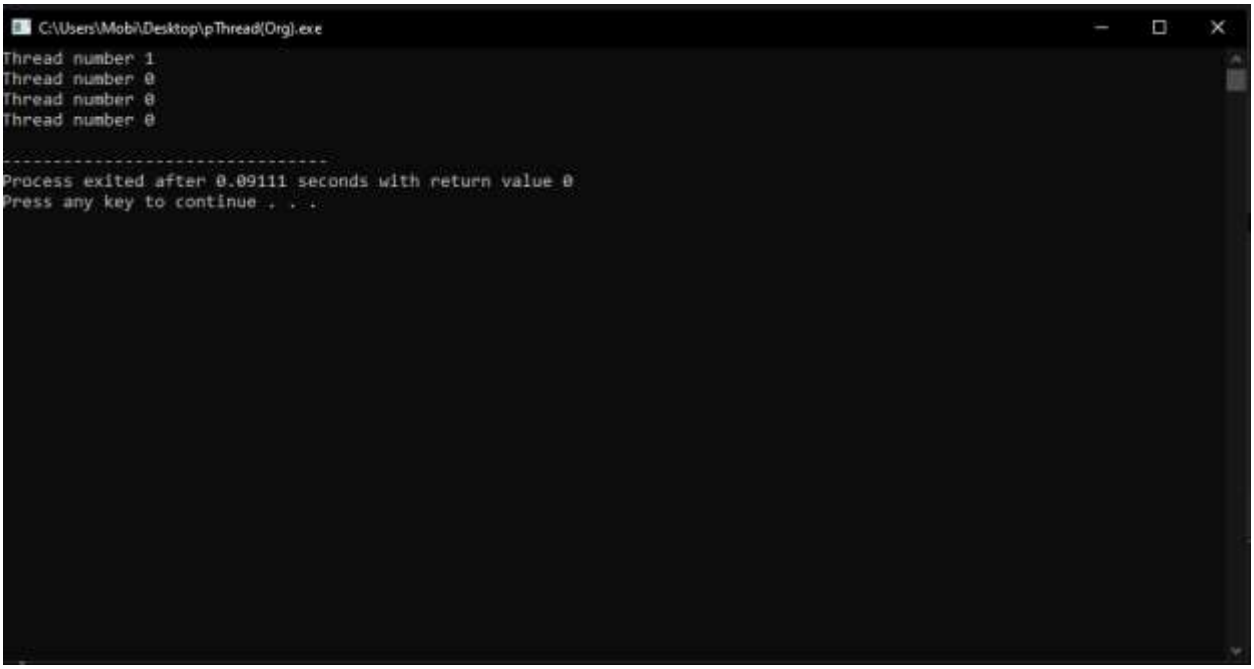
در این گزارش قصد داریم کدِ معرفی‌شده در مبحثِ pThread را اجرا کرده و شیوه‌ی عملکردِ آن را ارزیابی کنیم. همچنین در صورتِ مشاهده‌ی ایراد در کد یا الگوی نادرست در اعدادِ خروجی، الگوریتمِ آن را اصلاح کنیم.

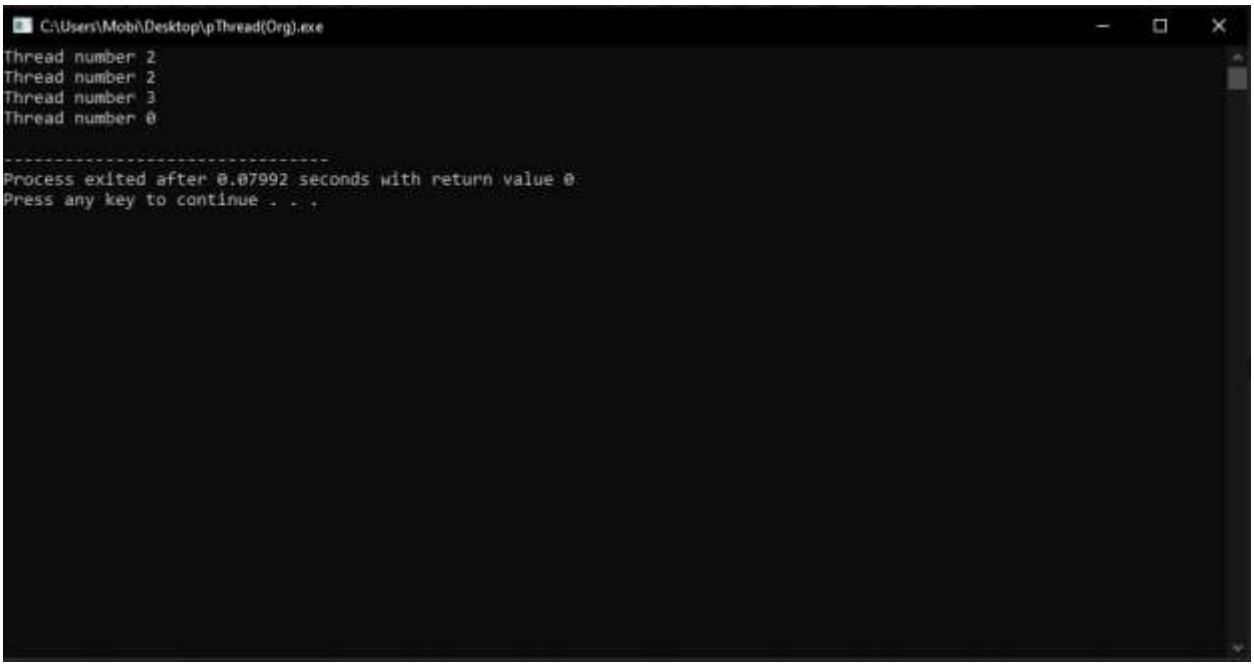
کدِ pThread در تصویر زیر نشان داده شده:

```c
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 4

void *threadFunc(void *pArg) { /* thread main */
    int *p = (int*)pArg;
    int myNum = *p;
    printf("Thread number %d\n", myNum);
    return 0;}

int main(void) {
    int i;
    pthread_t tid [NUM_THREADS];
    for(i = 0; i < NUM_THREADS; i++) { /* create/fork threads */

        pthread_create(&tid[i], NULL, threadFunc, &i);}
    for(i = 0; i < NUM_THREADS; i++) { /* wait/join threads */

        pthread_join(tid [i], NULL);}
    return 0;}
```

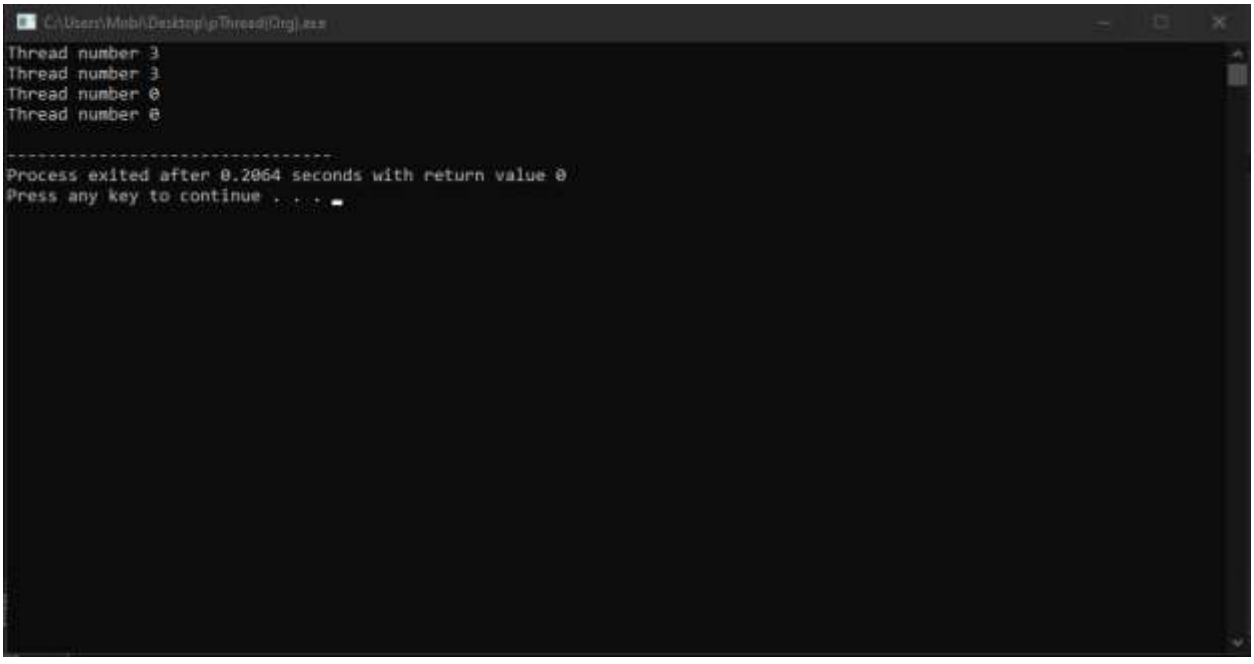
حال، کدِ بالا را پنج مرتبه اجرا کرده و خروجی‌ها را برای استفاده در روند تحلیل، بررسی می‌کنیم:



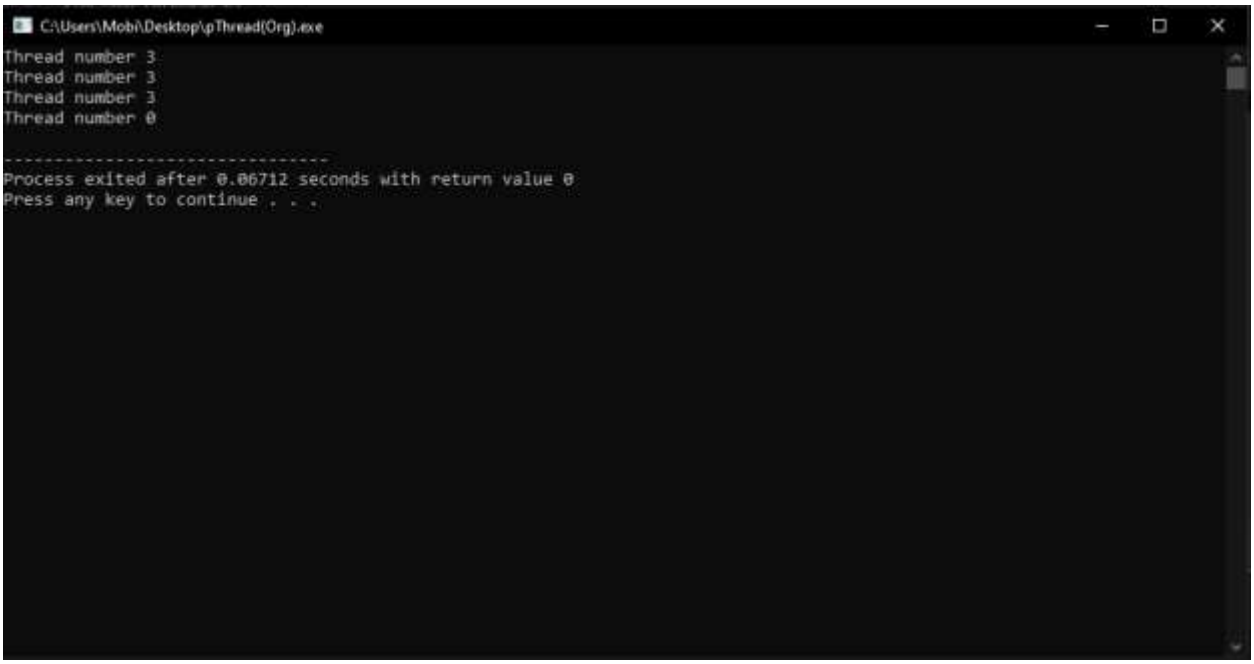



C:\Users\Mobi\Desktop\pThread(Org).exe
Thread number 2
Thread number 2
Thread number 3
Thread number 0

Process exited after 0.07992 seconds with return value 0
Press any key to continue . . .


Thread number 3
Thread number 3
Thread number 0
Thread number 0

Process exited after 0.2064 seconds with return value 0
Press any key to continue . . .



همانطور که در تصاویرِ بالا می‌بینید، کد با موفقیت اجرا می‌شود، اما اعدادی که در خروجی چاپ شده‌اند، الگوی نادرستی دارند.

ما با اجرای این کد، در حقیقت انتظار داریم که اعدادِ 0، 1، 2 و 3 با هر ترتیبِ دلخواهی، اما بدونِ تکرار در خروجی چاپ شوند، اما همانطور که تکرار می‌کنید، این قانون بارها در خروجی‌های مختلف نقض شده.

مشکلِ اصلیِ کد در واقع این است که متغیرِ حیاتیِ i در تابعِ main تعریف شده و به همه‌ی زیربخش‌ها یا بلوک‌های فرزند این تابع، اجازه‌ی دسترسی به این متغیر داده شده. همانطور که در کد می‌بینید، وقتی که مقدار i در یک thread عوض می‌شود، تمام thread های دیگر می‌توانند مقدار جدید را خوانده و پردازشِ خود را بر اساسِ آن دنبال کنند. مجموعِ این شرایط موجب می‌شود که شرایطی تحت عنوانِ "شرایطِ زقابتی" یا "Race Condition" در مسئله به‌وجود بیاید.

در واقع، درست زمانی که یک thread نیاز به خواندنِ مقدارِ i دارد، یک thread دیگر مشغولِ دستکاری و تغییر مقدارِ این متغیر است.

با تحلیلِ خروجی‌های نشان‌داده‌شده در تصاویرِ بالا درمی‌یابیم که رقابتی که میانِ thread ها ایجاد شده، موجب می‌شود که به طور مثال، تحتِ شرایطی پیش از آنکه thread اول به تابع threadfunc دسترسی پیدا کند، مقدارِ مورد نیازش در تابعِ pthread_create تغییر کند (افزایش پیدا کند). توجه کنید که آرگومانِ دریافت شده توسطِ تابعِ pthread_create از نوعِ اشاره‌گر به آدرسِ متغیردر حافظه است و این عامل در بر هم زدن ترتیب و توالیِ اجرای thread ها بی‌تاثیر نیست.

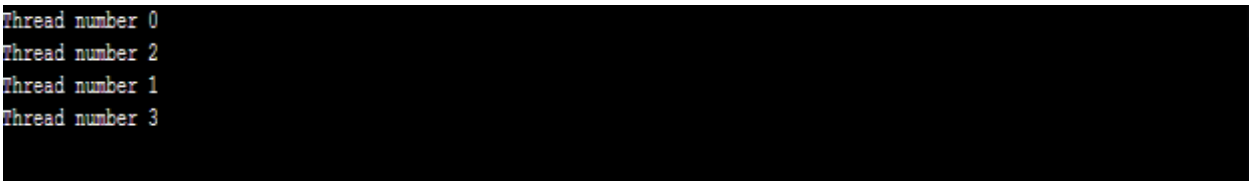
به طورِ خلاصه، رقابت بر سر زنجیره‌ی منابعِ مشترک موجب می‌شود که برخی thread ها در یک بازه‌ی زمانیِ مشخص تمامِ منابع را تصاحب کرده و برخی دیگر از thread ها از دسترسیِ منصفانه به منابع باز بمانند.

برای حلِ مشکلِ به‌وجودآمده، ابتدا یک متغیرِ موقت با عنوانِ temp_i می‌سازیم که نقش نگهدارنده‌ی نسخه‌ی مخصوصِ هر thread ار مقدارِ کنونیِ متغیرِ i را خواهد داشت. حال در زمانِ تعریفِ یک thread جدید، آدرسِ خانه‌ای از آرایه که به آن thread اشاره می‌کند، را به عنوانِ آرگومانِ چهارم به تابعِ pthread_create پاس می‌کنیم. حال با اجرای کد در خواهیم یافت که همین نسخه‌ی خصوصی از متغیرِ i باعث می‌شود که مشکلِ شرایطِ رقابتی به کلی حل شود.

تصویرِ زیر کدِ اصلاح‌شده را نشان می‌دهد:

```c
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 4

void *threadFunc(void *pArg) { /* thread main */
    int *p = (int*)pArg;
    int myNum = *p;
    printf("Thread number %d\n", myNum);
    return 0;}

int main(void) {
    int i;
    pthread_t tid [NUM_THREADS];
    int temp_i[NUM_THREADS];
    for(i = 0; i < NUM_THREADS; i++) { /* create/fork threads */

        temp_i[i] = i;
        pthread_create(&tid[i], NULL, threadFunc, &temp_i[i]);}

    for(i = 0; i < NUM_THREADS; i++) { /* wait/join threads */

        pthread_join(tid [i], NULL);}
    return 0;}
```

در ادامه 3 نمونه از خروجی‌های تولیدشده توسط این کد را مشاهده می‌کنید:



پایان.