تمرین سری چهارم سیستم عامل

سوال دو

در این سوال به منظور رعایت ترتیب ۳ حرف از حروف موجود باید از ۲ سمافور استفاده کنیم چرا که در این سوال قصد داریم حرف A را قبل از حرف F و حرف F را قبل از حرف C چاپ کنیم . برای هر ترتیب دوتایی از این ها به یک سمافور نیازمندیم یعنی برای رعایت ترتیب A و F یک سمافور و برای رعایت ترتیب F و C سمافوری دیگر نیاز داریم .

در بخش زیر در ابتدای تابع main دو سمافور با نام های sem1 و sem2 ایجاد کردیم و با مقدار صفر مقداردهی اولیه کردیم . (نامگذاری آنها را خارج از تابع main و به صورت global در بالای کد انجام دادیم)

سپس برای اجرای هر کدام از فرآیند های p1 و p2 ۲ thread با نام های thread1 و thread2 نامگذاری کردیم و در ۲ شرط نوشته شده thread را با استفاده از تابع pthread_create ساختیم و برای thread1 فرآیند P1 و برای thread2 فرآیند p2 را صدازدیم و شرط گذاشتیم که اگر ساخت thread ها با توجه به attribute های آن ها با خطا مواجه شد پیام خطای مناسب چاپ شود و ۱ return شود که به معنای failure است .

```
// semaphore initializations
sem_init(&sem2, 0, 0);
sem_init(&sem1, 0, 0);

// thread creations
pthread_t thread1, thread2;

if (pthread_create(&thread1, NULL, p1, NULL) != 0) {
    perror("Thread creation failed");
    return 1;
}

if (pthread_create(&thread2, NULL, p2, NULL) != 0) {
    perror("Thread creation failed");
    return 1;
}
```

برای آنکه اطمینان داشته باشیم حرف A قبل از حرف F چاپ میشود تابع wait را روی sem2 قبل از چاپ حرف F در فرآیند p1 صدامیزنیم و تابع post را بعد از چاپ حرف A در فرآیند p2 صدا میزنیم در نتیجه اگر حرف A چاپ نشود مقدار sem2 برابر با صفر میماند و از sem_wait در فرآیند اول نمیتوانیم عبور کنیم و حرف F در این صورت قبل از حرف A چاپ نمیشود .

```c
// semaphore declarations
sem_t sem1, sem2;

void* p1(void* arg) {

    sem_wait(&sem2);
    printf("F\n");

    sem_post(&sem1);
    printf("E\n");
    printf("G\n");

    return NULL;
}

void* p2(void* arg) {


    printf("A\n");
    sem_post(&sem2);

    sem_wait(&sem1);
    printf("C\n");
    printf("B\n");

    return NULL;
}
```

حال برای رعایت ترتیب میان F و C نیز به همین صورت با یک semaphore دیگر که sem1 نامیدیم عمل میکنیم یعنی قبل از C برای sem1 تابع wait را صدا میزنیم و بعد از چاپ حرف F نیز تابع post را برای sem1 صدامیزنیم تا در این صورت C پس از F چاپ شود و ترتیب چاپ باقی حروف نیز اهمیتی ندارد .

سپس در main با استفاده از تابع pthread_join برای terminate شدن thread ها صبر میکنیم و با استفاده از sem_destroy سمافور های ساخته شده در ابتدای تابع main را از بین میبریم و با شرط موجودچک میکنیم تا در صورت موفقیت آمیز نبودن این عملیات پیام خطای مناسب چاپ شود .

```c
// Wait for threads to finish
pthread_join(thread1, NULL);
pthread_join(thread2, NULL);

// semaphore destroy
if (sem_destroy(&sem1) != 0 || sem_destroy(&sem2) != 0) {
    perror("Semaphore destruction failed");
    return 1;
}
```

در فایل Q2_1.c این سوال با ۳ سمافور پیاده سازی شده و تفاوت در ترتیب چاپ بقیه حرف است .

```
qazal@Qazal:~/Desktop/OS/OS-Assignments/OS-HW4/Q2$ make Q_Two_first
gcc -pthread  Q2.c -o Q2
./Q2
A
F
E
G
C
B
```

```
qazal@Qazal:~/Desktop/OS/OS-Assignments/OS-HW4/Q2$ make Q_Two_second
gcc -pthread Q2_1.c -o Q2_1
./Q2_1
A
F
C
B
E
G
```