دانشگاه صنعتی امیرکبیر
( پلی تکنیک تهران )

دانشکده مهندسی کامپیوتر

به نام خدا

# تمرین سری پنجم درس سیستم های عامل

## پاییز 1403

## استاد درس: دکتر زرندی

توضیحات:

- پاسخ به تمرین ها باید به صورت انفرادی صورت پذیرد. درصورت مشاهده هر گونه تقلب نمره صفربرای کل تمرین منظور خواهد شد.
- تمیزی و خوانایی جواب تمرین ها از اهمیت بالایی برخوردار است. لطفا این مورد را رعایت کنید تا نمره ای به این سبب از شما کسر نگردد.
- لطفا پاسخ تمرین ها را در قالب یک فایلPDF با نام "HW5_StudentNumber.pdf" در سامانه کورسز و در مهلت معین شده بارگذاری فرمایید.
- در صورت برخوردن به هرگونه مشکل در رابطه با تمرین میتواند از طریق ایمیل os.fall1403@gmail.com و یا تلگرام با تدریسیاران در ارتباط باشید.

**سوال اول)**

به سوالات زیر پاسخ دهید.

الف) مزایای استفاده از ریسمان‌ها در مقابل فرایندها چیست؟

ب) وظایف (tasks) می توانند به دو صورت موازی و هم روند اجرا شوند. تفاوت این دو روش را توضیح دهید.

ج) انواع مدل‌های چندریسمانی را نام برده و توضیح دهید.

د) انواع حالت وضعیت ریسمان‌ها را نام برده و هرکدام را توضیح دهید.

ه) Thread-local storage (TLS) چیست و در چه مواقعی کاربرد دارد؟ تفاوت آن با متغیرهای داخلی را شرح دهید.

و) انواع روش‌های thread termination را نام برده و هرکدام را مختصر توضیح دهید.

**سوال دوم)**

کد زیر را در نظر بگیرید. تابع create_thread() یک ریسمان جدیدی را در فرایند فراخوانی شروع می‌کند. چند فرآیند منحصر به فرد ایجاد می شود؟ چه تعداد رشته منحصر به فرد ایجاد می‌شود؟ توضیح دهید.

```
pid_t pid;
 pid = fork();
 if (pid == 0) { // Child process
    fork();
    thread_create( . . . );
 }
 fork();
```

**سوال سوم)**

الف) race condition چه مواقعی پیش می‌آید و باعث چه مشکلی می‌شود؟ چطور می‌توان از آن جلوگیری کرد؟

ب) در قطعه کد زیر توضیح دهید race condition در کدام قسمت ممکن است به وجود بیاید و یک سناریو که باعث ناسازگاری داده می شود مثال بزنید.

```
int shared_counter = 0;

void* increment_counter(void* arg) {
    for (int i = 0; i < 1000000; ++i) {
        shared_counter++;
    }
    return NULL;
}

int main() {
    pthread_t thread1, thread2;
    pthread_create(&thread1, NULL, increment_counter, NULL);
    pthread_create(&thread2, NULL, increment_counter, NULL);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    printf("Final value of shared_counter: %d\n",
shared_counter);
    return 0;
}
```