به نام خدا

دانشکده مهندسی کامپیوتر

تمرین شماره ۱

درس: سیستم‌های عامل
نیم‌سال دوم ۰۴-۰۳
استاد: دکتر جلیلی

- توجه داشته باشید که نیازی به تحویل پاسخ این تمرین نیست، اما حل‌کردن آن می‌تواند کمک شایانی در پاسخگوئی شما به سؤالات آزمونک‌ها کند.
- در صورت وجود هرگونه ابهام، می‌توانید با این آدرس ایمیل در ارتباط باشید.

۱- روش دسترسی مستقیم به حافظه[1] (DMA) را به طور خلاصه توضیح داده و سپس به سؤال‌های زیر پاسخ دهید.

الف) چگونه پردازنده با دستگاه ارتباط برقرار کرده و یک عملیات DMA را اجرا می‌کند، و چگونه از اتمام عملیات مطلع می‌شود؟

ب) چرا استفاده از DMA در مقایسه با روش‌های مبتنی بر وقفه[2]، کارایی سیستم را افزایش می‌دهد؟ در چه شرایطی ممکن است این فرض در ارتباط با DMA برقرار نباشد؟

ج) زمانی که کنترل‌کننده DMA در حال انتقال داده به حافظه است، پردازنده می‌تواند هم‌زمان پردازه[3]های دیگری را اجرا کند. آیا در اجرای هم‌زمان عملیات DMA و سایر پردازه‌ها، ممکن است مشکلی وجود داشته باشد؟ توضیح دهید و اولویت اجرای هرکدام را مشخص کنید.

۲- الف) ساختارهای میکرو کرنل، لایه‌ای و یکپارچه را از نظر کارایی و انعطاف‌پذیری مقایسه کنید.

ب) شباهت‌ها و تفاوت‌های معماری‌های پیمانه‌ای[4] و لایه‌ای را توضیح دهید.

ج) معماری متقارن و نامتقارن را در سامانه‌های چند پردازه‌ای مقایسه کنید.

۳- سه روش انتقال داده[5] از نرم‌افزار به سیستم‌عامل در هنگام اجرای یک فراخوانی سیستمی[6] را توضیح دهید.

۴- الف) پردازه زامبی[7] چیست و در چه شرایطی ایجاد می‌شود.

ب) پردازه یتیم[8] چیست و در چه شرایطی ایجاد می‌شود.

ج) مشخص کنید که هر یک از برنامه‌های زیر، منجر به ایجاد کدام یک از پردازه‌های فوق خواهد شد. انتخاب خود را با ارائه دلیل توجیه نمایید.

[1] Direct Memory Access
[2] Interrupt
[3] Process
[4] Modular
[5] Parameter Passing
[6] System Call
[7] Zombie
[8] Orphan

```c
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{

    pid_t child_pid = fork();

    // Parent process
    if (child_pid > 0)
        sleep(60);

    // Child process
    else
        exit(0);

    return 0;
}
```

برنامه ۱

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int pid;
    pid = fork();

    if (pid == 0)
    {
        printf("I am the child, my process ID is %d\n", getpid());
        printf("My parent's process ID is %d\n", getppid());
        sleep(30);
        printf("\nAfter sleep\nI am the child, my process ID is %d\n", getpid());
        printf("My parent's process ID is %d\n", getppid());
        exit(0);
    }
    else
    {
        sleep(20);
        printf("I am the parent, my process ID is %d\n", getpid());
        printf("The parent's parent, process ID is %d\n", getppid());
        printf("Parent terminates\n");
    }
    return 0;
}
```

برنامه ۲

۵- الف) تفاوت بین چند برنامگی تک پردازه‌ای[9] و چندوظیفگی تک پردازه‌ای[10] را با رسم شکل بیان کنید.

ب) چگونه بهره‌گیری از چند برنامگی باعث افزایش کارایی سیستم‌های تک پردازنده‌ای می‌شود؟

ج) مزایا و معایب استفاده از چند پردازه‌ای[11] در مقایسه با سیستم‌های تک پردازنده‌ای چیست؟

د) چگونه استفاده از چندوظیفگی می‌تواند کاربرپسندی سیستم‌عامل را بهبود ببخشد؟

ه) آیا ممکن است در سامانه‌های چندپردازنده، در دسترسی به حافظه چالش خاصی وجود داشته باشد؟ توضیح دهید.

و) مدیریت حافظه نهان در پردازنده‌های چندهسته‌ای چه تفاوتی با پردازنده‌های تک‌هسته‌ای دارد؟

۶- پس از اجرای قطعه کد زیر، چه تعداد پردازه بر حسب n خواهیم داشت؟ توضیح دهید. (با احتساب پردازه والد)

```
for(i = 0; i < n; i++)
    fork();
```

۷- شرح دهید که پس از اجرای برنامه ۳، خروجی در خط A چه خواهد بود؟

[9] Single Processing Multi-Programming
[10] Single Processing Multi-Tasking
[11] Multi-Processing

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
    pid t pid;
    pid = fork();
    if (pid == 0)
    { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0)
    { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d", value); /* LINE A */
        return 0;
    }
}
```

برنامه ۳

۸- خروجی برنامه ۴ در خطوط C و P چه خواهد بود؟ توضیح دهید.

```c
#include <pthread.h>
#include <stdio.h>
int value = 0;
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
    pid t pid;
    pthread t tid;
    pthread attr t attr;
    pid = fork();
    if (pid == 0)
    { /* child process */
        pthread attr init(&attr);
        pthread create(&tid, &attr, runner, NULL);
        pthread join(tid, NULL);
        printf("CHILD: value = %d", value); /* LINE C */
    }
    else if (pid > 0)
    { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d", value); /* LINE P */
    }
}

void *runner(void *param)
{
    value = 5;
    pthread exit(0);
}
```

برنامه ۴

۹- الف) ارتباطات بین پردازه‌ای[12] (IPC) را شرح دهید.

ب) تفاوت بین دو مدل ارتباطی حافظه مشترک[13] و پیام‌رسانی[14] را در بستر این نوع ارتباطات بررسی کرده، مزایا و معایب استفاده از هرکدام را نام ببرید.

ج) در هنگام استفاده از حافظه مشترک بین پردازه‌ها، امکان وقوع مشکلی تحت عنوان ناسازگاری کش در پردازنده‌های چندهسته‌ای وجود دارد. این مشکل و نحوه مدیریت آن توسط سیستم‌عامل را شرح دهید،

۱۰- الف) مراحل مختلف بوت شدن یک سیستم کامپیوتری، از لحظه‌ای که دکمه روشن کردن کامپیوتر فشرده می‌شود، تا زمانی که سیستم‌عامل قادر به سرویس‌دهی به کاربر است را در نظر بگیرید. سپس مفهوم بوت امن[15] را بررسی کنید و بررسی کنید که این مفهوم، یک سیستم کامپیوتری را در برابر چه تهدیدهایی امن خواهد کرد.

ب) حال فرض کنید که UEFI کامپیوتر شما، دارای حالت بوت امن است و امکان غیرفعال‌کردن این حالت نیز برای شما فراهم نیست. اگر نیاز به نصب یک سیستم‌عامل با Bootloader اختصاصی باشید، و این Bootloader توسط UEFI کامپیوتر شما قابل‌اعتماد شناخته نشود، توضیح دهید که چگونه می‌توان این سیستم‌عامل را نصب کرد.

ج) در مورد Mini Bootloader تحقیق کنید و این مفهوم را شرح دهید. سپس بررسی کنید که این مفهوم با بوت امن چه ارتباطی دارد و چه مزیتی به آن اضافه می‌کند.

۱۱- الف) فراناظر ماشین مجازی[16] چیست و چه وظایفی دارد؟

ب) فرا ناظرهای نوع ۱ و نوع ۲ را با یکدیگر مقایسه کرده، مزایا و معایب هریک را بیان کنید.

ج) مزایای امنیتی استفاده از ماشین‌های مجازی در مقابل سیستم‌های غیر مجازی‌سازی شده چیست؟

۱۲- چرا در سیتم‌عامل لینوکس، برای مدیریت فرآیندها از لیست پیوندی دوطرفه[17] استفاده می‌شود و ساختارهایی مانند آرایه یا لیست پیوندی یک‌طرفه مورد استفاده قرار نگرفته‌اند؟

۱۴- الف) چه اطلاعاتی در فیلد mm * mm_struct ذخیره می‌شود و چرا این اطلاعات برای اجرای فرآیند حیاتی هستند؟

ب) چه روش‌های دیگری به جز $list\_head\_children$ برای ردیابی فرایندهای فرزند وجود دارد و این روش چه مزایایی نسبت به سایر روش‌ها دارد؟

[12] Inter-Process Communications
[13] Shared memory
[14] Message passing
[15] Secure Boot
[16] Hypervisor
[17] Doubly Linked List

ج) فیلد $files * files\_struct$ چگونه به مدیریت فایل‌های باز شده توسط فرآیند کمک می‌کند و چه مکانیزم‌هایی برای پاک‌سازی منابع هنگام خاتمه فرآیند وجود دارد؟

د) نقش $task\_struct$ در مدیریت فرآیندها چیست و چگونه به کرنل کمک می‌کند تا فرآیندهای فعال را ردیابی کند؟

۱۵- در یک سیستم مبتنی بر RPC:

الف) اگر چندین کاربر به طور هم‌زمان سعی کنند یک فایل را تغییر دهند، چه مشکلات هم‌روندی[۱۸] ممکن است به وجود بیاید و چگونه می‌توان آن‌ها را مدیریت کرد؟

ب) اگر کاربر یک درخواست RPC ارسال کند؛ اما به دلیل خرابی شبکه پاسخی دریافت نکند، چه مکانیزم‌هایی برای زمان‌بندی مجدد درخواست[۱۹] یا مدیریت Timeout باید پیاده‌سازی شوند؟

موفق باشید

[18] Synchronization
[19] Retry