Homework 2

Lectures 3, 4

Operating Systems

Dr. Javadi

Spring 2023

۱- به هنگام برنامه‌نویسی، حتی برنامه Hello, World که به هر زبانی بنویسید وارد رزومه خود می‌کنید :) از فراخوانی‌های سیستمی بسیاری استفاده می‌کنید. اغلب سیستم‌عامل‌های به‌روز از چندصد فراخوان سیستمی استفاده می‌کنند (حتی ممکن است برای عملیات‌های خیلی خاص مانند حساب کردن لگاریتم نیز فراخوانی سیستم وجود داشته باشند. با این حال اکثر برنامه نویسان هیچوقت این سطح از جزئیات را نمی‌بینند. به طور معمول، برنامه نویسان با کمک رابط برنامه نویسی یا API به پیاده‌سازی نرم‌افزارهای پیچیده می‌پردازند. این روابط برنامه نویسی خود فراخوانی‌های سیستمی را استفاده می‌کنند.

الف) در ابتدا توضیح دهید چرا برنامه نویسان بجای استفاده مستقیم فراخوانی‌های سیستمی از رابط برنامه نویسی استفاده می‌کنند.

ب) یکی از عوامل مهم در رسیدگی به فراخوانی‌های سیستمی، محیط زمان اجرا یا Runtime Environment است. محیط زمان اجرا یا به طور اختصار RTE مجموعه کامل از نرم‌افزار مورد نیاز برای اجرای برنامه‌های کاربردی نوشته شده در یک زبان برنامه‌نویسی خاص، از جمله کامپایلرها یا مفسرهای آن و همچنین نرم‌افزارهای دیگر مانند کتابخانه‌ها و لودرها است. تحقیق کنید و توضیح دهید، وجود RTE چگونه باعث می‌شود تا استفاده از فراخوانی‌های سیستمی راحت‌تر بشود؟

۲- شبه کد زیر را در نظر بگیرید. با در نظر گرفتن سناریوهای اجرای موفقیت آمیز و اجرایی که با خطا مواجه می‌شود:

```c
int main() {
    char input[100];
    FILE *fp;

    printf("Enter some text: ");
    fgets(input, 100, stdin);

    // Check if input is empty
    if (strlen(input) == 1) {
        printf("Error: Input is empty.\n");
        return 1;
    }

    fp = fopen("example.txt", "w");

    // Check if file was opened successfully
    if (fp == NULL) {
        printf("Error: File not found.\n");
        return 1;
    }

    fprintf(fp, "%s", input);
    fclose(fp);

    printf("Input written to file successfully!\n");

    return 0;
}
```

الف) حداقل ۷ دستور را نام ببرید که نیازمند اجرای فراخوانی سیستمی هستند.

ب) همانطور که می‌دانید توابع مورد استفاده در این کد، فراخوانی سیستمی نیستند بلکه واسطهایی برای این امر هستند. در پیاده‌سازی این توابع از چه روش‌هایی برای پاس دادن نام فایل‌های مورد استفاده به سیستم‌عامل می‌توان استفاده کرد؟

پ) آیا فایل کامپایل شده‌ای از نسخه‌ی کامل این کد را می‌توان در هر سیستم‌عاملی اجرا کرد؟ توضیح دهید.

۳- فراخوانی سیستمی در چه دسته‌بندی‌ای از انواع وقفه قرار می‌گیرد؟ به صورت کلی و خلاصه، مراحل اجرای یک فراخوانی سیستمی به عنوان یک وقفه را توضیح دهید.

۴- خروجی قطعه‌کدهای زیر چیست؟ ضمن رسم درخت پردازه‌های هر برنامه، راه حل خود را توضیح دهید.

الف)

```c
int main() {
   fork() && fork() && fork() && fork();
   printf("+");
   return 0;
}
```

ب)

```c
int main() {
(
  if (fork() && (!fork())) {
    if (fork() || fork()) {
      fork();
    }
  }

  return(0);
}
```

ج)

```c
int main() {
(
  int i = 0;
  while (i<2) {
    printf("%d", i);
    fork();
    i++;
  }

  return(0)
}
```

۵- زمانی که فراخوان سیستمی fork صدا زده می‌شود معمولا یکی از دو فرآیند (والد یا فرزند) یک فراخوان سیستمی دیگر به نام exec را صدا می‌زنند. تحقیق کنید و توضیح دهید که این فراخوان سیستمی چه کاری انجام می‌دهد و چرا یکی از دو فرآیند (والد یا فرزند) پس از اجرای دستور fork آن را صدا می‌زنند؟

سوال عملی

در سیستم عامل لینوکس و به زبان C کدی بنویسید که یک پردازه فرزند ایجاد کرده و آن را تبدیل به zombie کند. این پردازه باید حداقل به مدت ۱ دقیقه در سیستم باقی بماند. شما می‌توانید با استفاده از دستور ps -l وضعیت پردازه‌های خود را مشاهده کنید. اجزای مختلف خروجی دستور ps -l را توضیح دهید و پردازه zombie ایجاد شده را مشخص کنید. از کد خود به همراه خروجی دستور ps -l اسکرین شات بگیرید و همراه با توضیحات مربوط به کد و همچنین توضیحات دستور ps -l ارسال کنید.

به نکات زیر توجه کنید.

- مهلت ارسال تمرین ساعت ۲۳:۵۹ روز پنجشنبه ۱۸ فروردین ماه می‌باشد.
- در صورت کشف تقلب نمره تمرین ۰ در نظر گرفته می‌شود.
- سوالات خود را می‌توانید از طریق تلگرام از تدریسیارهای گروه خود بپرسید.
- فایل پاسخ تمرین را تنها با قالب **HW?_StudentNumber.pdf** در کورسز بارگزاری کنید.
  - نمونه: HW2_9831072