Threads

محمد مهدی رسول امینی	4001830235	آزمایشگاه سیستم عامل	تمرین 2	1403/09/25

خلاصه اهداف، روشها و نتایج:

هدف

1. ایجاد چند رشته: ایجاد 10 رشته (thread) در یک برنامه C.
2. عملکرد موازی: هر رشته باید یک تابع مشابه را اجرا کند.
3. ارسال شناسه منحصر به فرد: ارسال یک عدد منحصر به فرد به هر رشته برای شناسایی آن.
4. چاپ خروجی خاص: هر رشته باید سه بار عبارت Hello, World (thread n) را چاپ کند که در آن n شماره رشته است.
5. همگام‌سازی: اطمینان از اینکه برنامه تا زمانی که همه رشته‌ها کار خود را تمام نکرده‌اند، خاتمه نمی‌یابد.

روش‌ها

1. ایجاد رشته‌ها:

- از آرایه‌ای از نوع pthread_t برای نگهداری شناسه‌های رشته‌ها استفاده شده است. ا
- ز حلقه‌ای برای ایجاد 10 رشته استفاده شده است که هر رشته تابع printHello را اجرا می‌کند.

2. ارسال شناسه رشته:

- آرایه‌ای از اعداد (thread_ids) به هر رشته اختصاص داده شده و از طریق اشاره‌گر به آن ارسال می‌شود.

3. تابع printHello:

- این تابع شناسه دریافت شده را چاپ کرده و برای هر رشته سه بار عبارت موردنظر را نمایش می‌دهد.
- از تابع sleep(1) برای تأخیر بین چاپ‌ها استفاده شده است.

4. همگام‌سازی با pthread_join:

- از pthread_join برای اطمینان از اتمام تمام رشته‌ها قبل از پایان برنامه استفاده شده است.

نتایج

اجرای همزمان رشته‌ها:

- 10 رشته به طور موازی اجرا می‌شوند. ن

مایش خروجی:

- هر رشته 3 بار خروجی Hello, World (thread n) را با شناسه منحصر به فرد خود چاپ می‌کند.

همگام‌سازی موفق:

- برنامه تا زمانی که همه رشته‌ها اجرای خود را تمام نکنند، خاتمه نمی‌یابد.

اطلاع از پایان اجرا:

- پیام All threads have finished. بعد از پایان تمام رشته‌ها نمایش داده می‌شود.

اطلاعات پیش‌زمینه‌ای و معرفی موضوع آزمایش:

این آزمایش به بررسی اصول اولیه برنامه‌نویسی چندرشته‌ای (Multithreading) در زبان C با استفاده از کتابخانه pthread می‌پردازد. برنامه‌نویسی چندرشته‌ای یکی از تکنیک‌های مهم در بهینه‌سازی عملکرد نرم‌افزارهاست که امکان اجرای همزمان وظایف مختلف را فراهم می‌کند. این تکنیک به طور گسترده در سیستم‌های چندوظیفه‌ای، پردازش داده‌ها و کاربردهایی که نیاز به استفاده بهینه از منابع پردازشی دارند، استفاده می‌شود. هدف این آزمایش، آشنایی با نحوه ایجاد، مدیریت و همگام‌سازی رشته‌ها در زبان C است. در این برنامه، 10 رشته ایجاد می‌شود و هر یک وظیفه دارند سه بار عبارتی شامل شناسه منحصربه‌فرد خود را چاپ کنند. همچنین از تکنیک‌هایی مانند تخصیص شناسه به رشته‌ها و همگام‌سازی با استفاده از pthread_join استفاده می‌شود تا اطمینان حاصل شود که برنامه فقط پس از تکمیل تمام رشته‌ها خاتمه می‌یابد. این تمرین یک نمونه ساده از اجرای عملیات مستقل توسط چندین رشته و هماهنگی بین آن‌ها را نشان می‌دهد.

روش‌ها، ابزارها و نرم‌افزارهای مورد استفاده:

روش‌ها:

1. **ایجاد چندین رشته:**
   - از تابع pthread_createبرای ایجاد 10 رشته استفاده شده است. هر رشته وظیفه دارد تابع مشخصی به نام printHelloرا اجرا کند.
   - برای هر رشته، یک شناسه منحصربه‌فرد از طریق آرایه‌ای از اعداد ارسال می‌شود.
2. **اجرای وظایف مستقل در رشته‌ها:**
   - تابع printHelloبرای هر رشته سه بار یک پیام شامل شناسه آن رشته را چاپ می‌کند. از حلقه forو تأخیر زمانی با تابع sleepبرای ایجاد فاصله بین چاپ‌ها استفاده شده است.
3. **همگام‌سازی رشته‌ها:**
   - از تابع pthread_joinبرای اطمینان از اینکه برنامه اصلی (main thread) منتظر اتمام تمام رشته‌ها بماند، استفاده شده است.
4. **مدیریت خطا:**
   - در صورت بروز خطا در ایجاد رشته‌ها، پیام خطا نمایش داده شده و اجرای برنامه متوقف می‌شود.

ابزارها و نرم‌افزارهای مورد استفاده:

1. **زبان برنامه‌نویسی:C**
   - زبان پایه‌ای برای کار با کتابخانه‌های سیستمی و مفاهیم پایین‌سطحی مانند رشته‌ها.
2. **کتابخانه :pthread**
   - کتابخانه‌ای برای مدیریت رشته‌ها و همگام‌سازی آن‌ها. این کتابخانه توابعی مانند pthread_create، pthread_joinو pthread_exitرا فراهم می‌کند.
3. **کامپایلر:GCC**
   - کامپایلر استاندارد برای زبان C که قابلیت پشتیبانی از برنامه‌های چندرشته‌ای را دارد.
   - دستور کامپایل:

```
Gcc quera.c -o quera
```

4. **محیط توسعه:**
   - ویرایشگر متنی nano

5. **سیستم‌عامل:**
   - سیستم‌عامل Kali Linux.

توضیح مراحل انجام کار با جزئیات کافی:

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h> // برای تابع sleep

#define NUM_THREADS 10

void* printHello(void* thread_id) {
    int tid = *(int*)thread_id;
    for (int i = 0; i < 3; i++) {
        printf("Hello, World (thread %d)\n", tid);
        sleep(1); //delay
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    int thread_ids[NUM_THREADS];

    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i + 1;
        int rc = pthread_create(&threads[i], NULL, printHello,
(void*)&thread_ids[i]);
        if (rc) {
            printf("Error: unable to create thread %d\n", i + 1);
            return 1;
        }
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("All threads have finished.\n");
    return 0;
}
```

1.آماده‌سازی و تعریف مقادیر اولیه:

- **تعریف تعداد رشته‌ها :**یک ثابت NUM_THREADSبا مقدار 10 تعریف می‌شود تا نشان‌دهنده تعداد رشته‌های موردنیاز باشد.
- **تعریف آرایه‌های موردنیاز:**
  - یک آرایه از نوع pthread_tبرای ذخیره شناسه رشته‌ها.(threads)
  - یک آرایه از نوع intبرای ذخیره مقادیر شناسه هر رشته.(thread_ids)

2.نوشتن تابع وظیفه رشته‌ها:(printHello**)**

- **ورودی تابع :**این تابع یک اشاره‌گر دریافت می‌کند که به شناسه رشته اختصاص یافته است.
- **عملیات داخل تابع:**
  - شناسه رشته از طریق ورودی دریافت و به متغیر محلی تبدیل می‌شود.
  - یک حلقه forسه بار عبارت Hello, World (thread n)را چاپ می‌کند که در آن nشناسه رشته است.
  - از تابع sleep(1)برای ایجاد تأخیر 1 ثانیه‌ای بین هر بار چاپ استفاده می‌شود.
- **پایان رشته :**تابع با pthread_exitخاتمه می‌یابد.

3.**ایجاد رشته‌ها:**

- **حلقه ایجاد رشته‌ها :**یک حلقه forبرای ایجاد 10 رشته اجرا می‌شود:
  - شناسه منحصربه‌فرد هر رشته (thread_ids[i]) مقداردهی می‌شود.
  - با استفاده از تابع pthread_createیک رشته جدید ایجاد شده و تابع printHelloبه آن اختصاص می‌یابد.
  - اگر ایجاد رشته موفقیت‌آمیز نباشد، پیام خطای مربوط به آن نمایش داده می‌شود و برنامه خاتمه می‌یابد.

4.هم‌گام سازی رشته‌ها:

- **انتظار برای پایان کار رشته‌ها:**
  - یک حلقه forدیگر اجرا می‌شود تا با استفاده از تابع pthread_joinاطمینان حاصل شود که برنامه اصلی تا پایان کار هر 10 رشته منتظر بماند.
  - این کار تضمین می‌کند که هیچ رشته‌ای در حین اجرای برنامه ناقص باقی نماند.

5.اتمام برنامه:

- پس از اتمام کار تمام رشته‌ها، برنامه پیام .All threads have finishedرا چاپ می‌کند.
- تابع mainبا مقدار ;0 returnخاتمه می‌یابد.

مراحل اجرا روی سیستم:

1. **نوشتن کد :**کد را در یک فایل با نامی مانند threads_program.cذخیره کنید.
2. **کامپایل کد:**
   - از دستور زیر برای کامپایل برنامه استفاده کنید:

```
gcc quera.c -o quera
```

3. **اجرای برنامه:**
   - برنامه کامپایل‌شده را اجرا کنید:

```
./quera
```

خروجی اسکریپت:

```
Hello, World (thread 1)
Hello, World (thread 4)
Hello, World (thread 2)
Hello, World (thread 3)
Hello, World (thread 7)
Hello, World (thread 5)
Hello, World (thread 8)
Hello, World (thread 6)
Hello, World (thread 9)
Hello, World (thread 10)
Hello, World (thread 4)
Hello, World (thread 1)
Hello, World (thread 2)
Hello, World (thread 3)
Hello, World (thread 7)
Hello, World (thread 5)
Hello, World (thread 8)
Hello, World (thread 6)
Hello, World (thread 9)
Hello, World (thread 10)
Hello, World (thread 1)
Hello, World (thread 7)
Hello, World (thread 5)
Hello, World (thread 2)
Hello, World (thread 3)
Hello, World (thread 8)
Hello, World (thread 6)
Hello, World (thread 4)
Hello, World (thread 9)
Hello, World (thread 10)
All threads have finished.
```

نتایج:

ایجاد موفق رشته‌ها 10 **:**رشته به درستی ایجاد شدند و هر کدام وظیفه خود را به صورت مستقل اجرا کردند.
اجرای همزمان :رشته‌ها به صورت موازی اجرا شده و خروجی‌ها به طور همزمان و غیرترتیبی نمایش داده شدند.
چاپ پیام‌ها :هر رشته سه بار پیام Hello, World (thread n)را با شناسه خود چاپ کرد.
همگام‌سازی :برنامه اصلی پس از اتمام تمام رشته‌ها خاتمه یافت و پیام "All threads have finished." نمایش داده شد.
کاربردی بودن :عملکرد صحیح برنامه اهمیت هماهنگی و همگام‌سازی در برنامه‌نویسی چندرشته‌ای را نشان داد.

بحث و نتیجه‌گیری:

این برنامه نمونه‌ای ساده اما کاربردی از اصول برنامه‌نویسی چندرشته‌ای در زبان C را ارائه می‌دهد. استفاده از کتابخانه pthreadنشان می‌دهد چگونه می‌توان وظایف مستقل را به صورت همزمان اجرا کرد و منابع پردازشی را بهینه‌تر به کار گرفت.

در این آزمایش، 10 رشته ایجاد شد که هر کدام شناسه‌ای منحصربه‌فرد دریافت کردند و وظیفه داشتند سه بار پیام مشخصی را چاپ کنند. همزمانی اجرای رشته‌ها منجر به ترتیب نامنظم چاپ خروجی‌ها شد، که یکی از ویژگی‌های ذاتی برنامه‌نویسی چندرشته‌ای است. این رفتار نشان‌دهنده توانایی سیستم در مدیریت موازی وظایف است. با استفاده از تابع pthread_join، برنامه اصلی منتظر اتمام تمام رشته‌ها ماند و از بروز مشکلاتی مانند خاتمه زودهنگام برنامه جلوگیری شد.

نتیجه کلی این است که برنامه به درستی اهداف تعریف‌شده را محقق کرد. این آزمایش اهمیت استفاده از تکنیک‌های هماهنگی و همگام‌سازی مانند pthread_joinرا در برنامه‌های چندرشته‌ای برای جلوگیری از مشکلات احتمالی نشان داد. همچنین، چنین برنامه‌هایی می‌توانند پایه‌ای برای توسعه نرم‌افزارهای پیچیده‌تر در زمینه‌های پردازش موازی، سرورها و سیستم‌های چندوظیفه‌ای باشند.

منابع:

مستندات رسمیPthreads

- POSIX Threads Programming: راهنمای رسمی توابع و ساختارهای Pthreads.
(man7.org)

کتاب مرجع.Kerrisk, Michael :

- *The Linux Programming Interface: A Linux and UNIX System Programming Handbook.*
این کتاب مرجع جامع برای برنامه‌نویسی در محیط‌های لینوکسی و استفاده از Pthreads است.

آموزش آنلاین:GeeksforGeeks

- مقاله "Multithreading in C using Pthreads" برای درک نحوه پیاده‌سازی رشته‌ها و همگام‌سازی آن‌ها.
(geeksforgeeks.org)

مقالات فارسی در وب‌سایت توسینسو:

- آموزش "کامل‌ترین آموزش کار باThread ها در لینوکس" برای آشنایی با مفاهیم اولیه و پیشرفته.
(tosinso.com)

تجربیات کدنویسی و مستندسازی شخصی

ویکی‌پدیا:

- توضیحات عمومی در مورد مفهوم چندرشته‌ای و کاربردهای آن در سیستم‌عامل‌ها.