آزمایش ۷

آشنایی با ریسه‌ها

۱.۷ مقدمه

هدف اصلی این آزمایش، بررسی جنبه‌های مختلف ریسه‌ها و چندپردازشی (و چند ریسه‌ای) است. از اهداف اصلی این آزمایش پیاده‌سازی توابع مدیریت ریسه‌ها است:

- ساخت ریسه‌ها
- پایان بخشیدن به اجرای ریسه
- پاس دادن متغیر به ریسه‌ها
- شناسه‌های ریسه‌ها
- متصل شدن ریسه‌ها

۱.۱.۷ پیش‌نیازها

انتظار می‌رود که دانشجویان با موارد زیر از پیش آشنا باشند:

- برنامه‌نویسی به زبان C/C++
- دستورات پوسته‌ی لینوکس که در جلسات قبل فرا گرفته‌اند.

۲.۷ ریسه چیست؟

یک ریسه، شبه پردازه‌ای است که پشته‌ی خاص خود را در اختیار دارد و کد مربوط به خود را اجرا می‌کند. برخلاف پردازه، یک ریسه، معمولاً حافظه‌ی خود را با دیگر ریسه‌ها به اشتراک می‌گذارد. یک گروه از ریسه‌ها، یک مجموعه از ریسه‌ها است که در یک پردازه‌ی یکسان اجرا می‌شوند. بنابراین آنها یک حافظه‌ی یکسان را به اشتراک می‌گذارند و می‌توانند به متغیرهای عمومی یکسان، حافظه‌ی heap یکسان و ... دسترسی داشته باشند. همه‌ی ریسه‌ها می‌توانند به صورت موازی (استفاده از برش زمانی، یا اگر چندین پردازه وجود داشته باشد، به معنای واقعی موازی) اجرا شوند.

۳.۷ pthread

بر اساس تاریخ، سازندگان سخت‌افزار نسخه‌ی مناسبی از ریسه‌ها را براي خود پیاده‌سازی کردند. از آنجا که این پیاده‌سازی‌ها با هم تفاوت می‌کرد، پس کار را برای برنامه‌نویسان، برای نگارش یک برنامه‌ی قابل حمل دشوار می‌کرد. بنابراین نیاز به داشتن یک واسط یکسان برای بهره بردن از فواید ریسه‌ها احساس می‌شد. برای سیستم‌های Unix این واسط با نام POSIX IEEE ۱۰۰۳.۱c مشخص می‌شد و به پیاده‌سازی مرتبط با آن POSIX THREADS یا pthread گفته می‌شود. اکثر سازندگان سخت‌افزار، علاوه بر نسخه‌ی مناسب با خودشان، استفاده از pthread را نیز پیشنهاد می‌کنند. هاpthread در یک کتابخانه‌ی C تعریف شده‌اند که شما می‌توانید با برنامه‌ی خود link کنید.
توابع موجود در این کتابخانه به صورت غیررسمی به سه دسته تقسیم می‌شوند:

- مدیریت ریسه‌ها: دسته‌ی اول از این توابع به صورت مستقیم با ریسه‌ها کار می‌کنند. همانند ایجاد، متصل کردن و ...
- Mutex: دسته‌ی دوم از این توابع برای کار با هاmutex ایجاد شده‌اند. توابع مربوط به mutex ابزار مناسب برای ایجاد، تخریب، قفل و بازکردن هاmutex را در اختیار قرار می‌دهند.

- متغیرهای شرطی (Condition Variables): این دسته از توابع، برای کار با متغیرهای شرطی و استفاده از مفهوم همزمانی در سطح بالاتر در اختیار قرار می‌گیرند. این دسته از توابع برای ایجاد، تخریب، wait و signal بر اساس مقادیر معین متغیرها استفاده می‌شوند.

نکته ۱ در این جلسه قصد آن را داریم تا با دسته‌ی اول از توابع آشنا شویم. توابع مربوط به این دسته به طور خلاصه در جدول زیر مشاهده می‌شود: برای آشنایی با جزئیات می‌توانید از دستور man و یا اینترنت استفاده کنید.

جدول ۱.۷: توابع مربوط به مدیریت ریسه‌ها

نام تابع	کاربرد
pthread_create	از کتابخانه‌ی pthread، درخواست ساخت یک ریسه‌ی جدید را می‌کند.
pthread_exit	این تابع توسط ریسه استفاده شده تا پایان بپذیرد.
pthread_join	این تابع، برای ریسه‌ی مشخص شده صبر می‌کند تا پایان بپذیرد.
pthread_cancel	درخواست کنسل شدن ریسه‌ی مشخص شده را ارسال می‌کند.
pthread_attr_int	مقدار های attribute پاس داده شده به خود را با مقادیر پیش‌فرض پر می‌کند.
pthread_self	شماره‌ی ریسه را بر می‌گرداند.

۴.۷ شرح آزمایش

۱.۴.۷ آشنایی اولیه

۱. وارد سیستم عامل مجازی ایجاد شده در جلسات قبل شوید.

۲. با استفاده از تکه کد زیر، یک ریسه ایجاد کنید.

```c
#include<pthread.h>

int main()
{
   pthread_t the_thread;
   return 0;
}
```

دقت کنید در هنگام کامپایل، lpthread- را به پرچم‌های linker اضافه کنید:

```
# gcc thread.c -o threads -lpthread
```

دقت کنید در هنگام کامپایل، lpthread- را به پرچم‌های linker اضافه کنید:

۳. با استفاده از توابع pthread_create و pthread_join یک ریسه درست کنید که در آن شماره‌ی پردازه را چاپ کنید. همچنین در پردازه‌ی اصلی نیز شماره‌ی پردازه را چاپ کنید. دقت کنید پردازه‌ی اصلی بعد از پایان یافتن ریسه‌ها تمام شود. آیا شماره پردازه‌های چاپ شده یکسان می‌باشند؟

۴. برنامه‌ی بالا را در یک فایل جدید کپی کنید. حال، متغیر oslab را به این تکه کد به صورت عمومی اضافه کنید. حال، یک بار این متغیر را در ریسه‌ی اصلی و یک بار در ریسه‌ی فرزند تغییر دهید. بعد از تغییر در ریسه‌ی فرزند، بار دیگر در ریسه‌ی اصلی چاپ کنید. آیا ریسه‌ها کپی‌های جداگانه‌ای از متغیر را دارند؟

۵. با استفاده از تابع pthread_attr_init و تنظیم کردن های attribute ریسه به صورت پیش‌فرض، کدی بنویسید که در آن با گرفتن عدد n از ورودی، حاصل‌جمع اعداد ۲ تا n را چاپ کند.

۲.۴.۷ ریسه‌های چندتایی

در این قسمت قصد آن را داریم تا در یک کد، چند ریسه داشته باشیم.

- با استفاده از تابع pthread_create، تعدادی ریسه به تعداد دلخواه ایجاد کنید (حداقل پنج تا) و پیام World Hello را در آن چاپ کنید. سپس ریسه‌ها را با استفاده از تابع pthread_exit خاتمه دهید.

۳.۴.۷ تفاوت بین پردازه‌ها و ریسه‌ها

در این قسمت، قصد آن را داریم تا تفاوت میان پردازه‌ها و ریسه‌ها را بهتر متوجه بشویم.

۱. تکه کد زیر را به عنوان تابع ریسه در فایلی بنویسید:

دقت کنید در هنگام کامپایل، lpthread- را به پرچم‌های linker اضافه کنید

```
1 void kid(void* param)
2 {
3    int local_param;
4    printf("Thread %d, pid %d, addresses: &global: %X, &local: &X \n",
        pthread_self(), getpid(), &global_param, &local_param);
5    global_param++;
6    printf("In Thread %d, incremented global parameter=%d\n",
        pthread_self(), global_param);
7    pthread_exit(0);
8 }
```

دقت کنید در هنگام کامپایل، lpthread- را به پرچم‌های linker اضافه کنید:

۲. حال، در ریسه‌ی اصلی، یک متغیر عمومی به عنوان global_param تعریف کرده، مقداردهی کنید و دو ریسه‌ی فرزند با تابع kid ایجاد و اجرا کنید. در پایان ریسه‌ها نیز مقدار متغیر global_param را چاپ کنید.

۳. حال، مقدار متغیر عمومی را بار دیگر تغییر داده و یک متغیر محلی دیگر در تابع اصلی تعریف کنید. آن را نیز مقداردهی کنید. حال، با استفاده از تابع fork که در جلسات پیش یاد گرفته‌اید، یک پردازه‌ی فرزند ایجاد کرده و متغیرها را در آن دوباره مقداردهی کنید. تغییرات را با استفاده از تابع printf نمایش دهید.

۴.۴.۷ پاس دادن متغیرها به ریسه

تابع pthread_create تنها اجازه می‌دهد که یک متغیر به عنوان ورودی به ریسه داده شود. براي حالاتی که چند پارامتر می‌بایست به ریسه داده شود، این محدودیت به راحتی با استفاده از ساختار (structure) حل می‌شود. تمامی متغیرها می‌بایست به وسیله‌ی reference و تبدیل به void* پاس داده شوند.

- ساختار زیر را در فایلی قرار دهید:

```
1 typedef struct thdata {
2    int thread_no;
3    char message[100];
4 } stdata;
```

حال، در ریسهي اصلی، دو متغیر از ساختار معرفی شده ایجاد کنید و مقادیر آن را به صورت دلخواه تنظیم کنید. سپس، متغیرها را به دو ریسهي جداگانه پاس بدهید. در ریسهها نیز عدد و پیام ذخیره شده در ساختار را نمایش بدهید.