实验二 用户级线程库

第一部分：热身练习（30 分）

本项目为简单热身任务，旨在帮助回顾系统编程基础并为后续项目做准备。该部分需使用 Linux pthread 库编写多线程程序。

1、pThread 编程回顾（30 分）

1.1 任务描述

这部分练习只需修改 warmup.c 文件。代码框架中定义全局变量 x（初始值为 0）。需使用 pThread 创建 2 个线程，每个线程调用函数 inc_shared_counter 对 x 进行 5 次递增操作，总计递增 10 次。

要求：

1）使用 pthread_create 创建线程，inc_shared_counter 函数内需通过互斥锁（mutex）保证对 x 的互斥访问。

2）每次递增后打印当前 x 的值。

3）主线程需通过 pthread_join 等待子线程结束，最后输出 x 的最终值。

1.2 编译与预期输出

执行 make warmup 命令后，会生成 warmup 可执行文件。执行 ./warmup 后的预期输出如下所示：

```
x is incremented to 1
x is incremented to 2
x is incremented to 3
x is incremented to 4
x is incremented to 5
x is incremented to 6
x is incremented to 7
x is incremented to 8
x is incremented to 9
x is incremented to 10
The final value of x is 10
```

注意：由于使用互斥锁，输出顺序必须与上述完全一致。

参考资料

- pthread_create 手册：http://man7.org/linux/man-pages/man3/pthread_create.3.html
- pthread_join 手册：http://man7.org/linux/man-pages/man3/pthread_join.3.html

第二部分：用户级线程库和调度器（70 分）

在第一部分热身练习中，我们学习了如何在多线程编程中使用 pThread 库。第二部分中，我们深入探索线程库的内部逻辑。该部分要求实现一个与 pThread 接口完全兼容的纯用户级线程库。由于本次实验需要支持多线程的程序运行环境，你的代码中还需实现用于保证临界区互斥访问的 pthread 互斥锁（mutex）。本实验旨在说明操作系统中任务调度的机制与难点。

2、实现用户级线程库和调度器（70 分）

2.1 任务描述

该部分的代码框架结构如下：

```
OSLab2/
|-- my_pthread_t.h
|-- my_pthread.c
|-- Makefile
|-- Benchmark/
    |-- externalCal.c
    |-- genRecord.sh
    |-- Makefile
    |-- parallelCal.c
    |-- README
    |-- vectorMultiply.c
```

其中 my_pthread_t.h 文件包含线程库函数原型和重要数据结构定义。my_pthread.c 包含每个 API 函数的具体实现。Benchmark 文件夹包含用于验证实现和性能分析的基准测试程序。

2.2 线程库 API

本实验需要实现下面列出的所有 API 函数、对应的调度器函数以及其他必要的辅助函数。

```
int my_pthread_create(my_pthread_t *thread, pthread_attr_t *attr, void
*(*function)(void*), void *arg);
```

功能说明：创建并执行线程，可忽略 attr 参数。

```
void my_pthread_yield();
```

功能说明：主动让出 CPU 资源，触发上下文切换：当前线程状态保存后，调度器接管控制权并重新选择运行线程。

```
void my_pthread_exit(void *value_ptr);
```

功能说明：终止当前线程执行，value_ptr 非空时保存线程返回值。

```
int my_pthread_join(my_pthread_t thread, void **value_ptr);
```

功能说明：阻塞调用线程直至目标线程结束，value_ptr 用于接收目标线程返回值。

```
int my_pthread_mutex_init(my_pthread_mutex_t *mutex, const
pthread_mutexattr_t *mutexattr);
```
功能说明：初始化互斥锁，可忽略 mutexattr。

```
int my_pthread_mutex_lock(my_pthread_mutex_t *mutex);
```
功能说明：获取互斥锁，若锁已被占用则阻塞

```
int my_pthread_mutex_unlock(my_pthread_mutex_t *mutex);
```
功能说明：释放持有的互斥锁。

```
int my_pthread_mutex_destroy(my_pthread_mutex_t *mutex);
```
功能说明：销毁互斥锁对象，需确保锁处于释放状态。

2.3 用户级上下文

建议使用系统库 ucontext.h 管理用户级上下文。该库提供创建、切换和获取上下文的函数。上下文一旦开始执行将持续运行直至完成。

2.4 调度器

用户级线程库需自己实现调度器，本实验需实现以下两种调度策略：

1. **抢占式最短作业优先（PSJF）。**我们注意到调度器无法预知线程将运行多长时间。因此，在调度器中，我们可以记录每个线程已经运行的时间片数量，这是基于以下假设：线程已经运行的时间片越多，该作业完成所需的时间就越长。为了实现 PSJF，需要：
    - 在线程控制块（TCB）中维护时间片计数器。
    - 调度时选择计数器值最小的线程。
2. **多级反馈队列（MLFQ）。**在该算法中，需要维护一个具有多级优先级的队列结构。请注意，优先级越高的队列，其对应的时间片就越短。
    - 需要维护多级队列而非单一队列，例如 4 级或 6 级（具体技术可自行决定）。
    - 当线程用完一个时间片后，将其降级到下一优先级队列。
    - 调度器优先选择最高优先级队列中的线程。

需通过 setitimer 和 sigaction 设置时间中断（时间片为 t 毫秒）。默认使用 PSJF 策略，编译时可通过以下命令切换为 MLFQ：

```
$ make SCHED=MLFQ
```

2.5 基准测试

基准测试程序包括 parallelCal、vectorMultiply（CPU 密集型）和 externalCal（IO 密集型）。运行示例：

```
$ make
$ ./parallelCal 4  # 创建 4 个线程
```

在 my_pthread_t.h 中取消注释 #define USE_MY_PTHREAD 1 以使用自定义线程库。

2.6 建议步骤

1. 设计线程控制块（TCB）和队列结构。
2. 实现基础 API（如 my_pthread_create）和简单调度器（如 FCFS）。
3. 实现互斥锁。
4. 扩展调度器支持 PSJF 和 MLFQ。

参考资料

- POSIX 线程教程：链接 1、链接 2
- Linux 线程实现笔记：链接
- MLFQ：https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf

3、实验报告撰写要求

实验报告内容与形式：

1）实验报告中请阐述 my_pthread 相关函数（create，yield，exit，join），my_pthread_mutex 相关函数（init，lock，unlock，destroy）的实现细节，以及你在实验过程中遇到的问题和思考。

2）请解释描述你所实现的 PSJF 和 MLFQ 调度器的实现细节，以及线程库中如何实现调度器的上下文切换。

3）分析不同线程数下的基准测试结果，并与 pThread 库进行比较。

4）在实验报告最后以附录的形式分别粘贴 my_thread.h 和 my_thread.c 代码。**助教会对所有的实验报告进行查重，请各组独立完成编码与实验报告。我们对抄袭零容忍。**

4、实验报告与代码提交要求（会影响最后评分，请务必按格式要求提交）

提交内容，只提交四个文件：

1）**warmup.c** 文件

2）**my_pthread.h** 文件

3）**my_pthread.c** 文件

4）**实验报告 word 文档**，命名方式：**OSLab2-成员 1 学号姓名-成员 2 学号姓名.docx**

例如：OSLab2-B22035678 张三- B22035679 李四.docx

将以上四个文件打成一个压缩包，命名方式：**OSLab2-成员 1 学号姓名-成员 2 学号姓名.zip**。由班长或学委收集，统一发给任课老师。

5、实验评分标准

给分点	分数
第一部分	
提交的代码能够正确编译，程序运行结果正确	10%
正确实现 main 函数与 inc_shared_counter 函数	10%
第二部分	
提交的代码能够正确编译，所有测试程序运行结果正确	10%
my_pthread 相关函数(create, yield, exit, join)功能正确	10%

my_pthread_mutex 相关函数功能正确	10%
PSJF 和 MLFQ 调度器功能正确	20%
代码注释详细、正确，实验报告文档内容详实	30%
总计	**100%**