+--------------------+

| CS 140 |

| PROJECT 1: THREADS |

| DESIGN DOCUMENT |

+--------------------+

---- GROUP ----

>> Fill in the names and email addresses of your group members.

FirstName LastName <email@domain.example>

FirstName LastName <email@domain.example>

FirstName LastName <email@domain.example>

---- PRELIMINARIES ----

>> If you have any preliminary comments on your submission, notes for the

>> TAs, or extra credit, please give them here.

>> Please cite any offline or online sources you consulted while

>> preparing your submission, other than the Pintos documentation, course

>> text, lecture notes, and course staff.

ALARM CLOCK

===========

---- DATA STRUCTURES ----

>> A1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

---- ALGORITHMS ----

>> A2: Briefly describe what happens in a call to timer\_sleep(),

>> including the effects of the timer interrupt handler.

>> A3: What steps are taken to minimize the amount of time spent in

>> the timer interrupt handler?

---- SYNCHRONIZATION ----

>> A4: How are race conditions avoided when multiple threads call

>> timer\_sleep() simultaneously?

>> A5: How are race conditions avoided when a timer interrupt occurs

>> during a call to timer\_sleep()?

---- RATIONALE ----

>> A6: Why did you choose this design? In what ways is it superior to

>> another design you considered?

PRIORITY SCHEDULING

===================

---- DATA STRUCTURES ----

>> B1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

>> B2: Explain the data structure used to track priority donation.

>> Use ASCII art to diagram a nested donation. (Alternately, submit a

>> .png file.)

---- ALGORITHMS ----

>> B3: How do you ensure that the highest priority thread waiting for

>> a lock, semaphore, or condition variable wakes up first?

>> B4: Describe the sequence of events when a call to lock\_acquire()

>> causes a priority donation. How is nested donation handled?

>> B5: Describe the sequence of events when lock\_release() is called

>> on a lock that a higher-priority thread is waiting for.

---- SYNCHRONIZATION ----

>> B6: Describe a potential race in thread\_set\_priority() and explain

>> how your implementation avoids it. Can you use a lock to avoid

>> this race?

---- RATIONALE ----

>> B7: Why did you choose this design? In what ways is it superior to

>> another design you considered?

ADVANCED SCHEDULER

==================

---- DATA STRUCTURES ----

>> C1: Copy here the declaration of each new or changed `struct' or

>> `struct' member, global or static variable, `typedef', or

>> enumeration. Identify the purpose of each in 25 words or less.

更改的struct 全局或静态变量 typeof 声明：

**一、thread.h中**

1. 更新了struct thread{

int nice; //每个线程都要有nice值来决定对其他线程CPU影响

int64\_t recent\_cpu;//线程最近使用的CPU时间的估计值，与nice值共同影响线程优先级

}

1. 增加了方法

bool cmp\_priority(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux);

//该方法用在更新全部线程优先级的函数与需要更新线程优先级的函数中，实现线程就绪队列按照优先级大小排序

**二、thread.c中**

1、定义了浮点类型int64\_t fixed\_t,通过将整型左移14位得到17.14格式的浮点型来模拟浮点数运算，以便实现之后对于recent\_cpu与load\_avg两个实数的计算。

typedef int64\_t fixed\_t;

#define SHIFT\_AMOUNT 14

#define INT2FLOAT(n) ((fixed\_t)(n << SHIFT\_AMOUNT))

#define FLOAT2INTPART(x) (x >> SHIFT\_AMOUNT)

#define FLOAT2INTNEAR(x) (x >= 0 ? ((x + (1 << (SHIFT\_AMOUNT - 1))) >> SHIFT\_AMOUNT) : ((x - (1 << SHIFT\_AMOUNT)) >> SHIFT\_AMOUNT))

#define FLOATADDFLOAT(x, y) (x + y)

#define FLOATSUBFLOAT(x, y) (x - y)

#define FLOATADDINT(x, n) (x + (n << SHIFT\_AMOUNT))

#define FLOATSUBINT(x, n) (x - (n << SHIFT\_AMOUNT))

#define FLOATMULFLOAT(x, y) ((((int64\_t)x) \* y) >> SHIFT\_AMOUNT)

#define FLOATMULINT(x, n) (x \* n)

#define FLOATDIVFLOAT(x, y) ((((int64\_t)x) << SHIFT\_AMOUNT) / y)

#define FLOATDIVINT(x, n) (x / n)

1. 声明int64\_t load\_avg 作为系统平均负载，它估计过去一分钟内准备运行的线程的平均数量。

---- ALGORITHMS ----

>> C2: Suppose threads A, B, and C have nice values 0, 1, and 2. Each

>> has a recent\_cpu value of 0. Fill in the table below showing the

>> scheduling decision and the priority and recent\_cpu values for each

>> thread after each given number of timer ticks:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| timer ticks | recent\_cpu | | | priority | | | thread to run |
| A | B | C | A | B | C |
| 0 | 0 | 0 | 0 | 63 | 61 | 59 | A |
| 4 | 4 | 0 | 0 | 62 | 61 | 59 | A |
| 8 | 8 | 0 | 0 | 61 | 61 | 59 | B |
| 12 | 8 | 4 | 0 | 61 | 60 | 59 | A |
| 16 | 12 | 4 | 0 | 60 | 60 | 59 | B |
| 20 | 12 | 8 | 0 | 60 | 59 | 59 | A |
| 24 | 16 | 8 | 0 | 59 | 59 | 59 | C |
| 28 | 16 | 8 | 4 | 59 | 59 | 58 | B |
| 32 | 16 | 12 | 4 | 59 | 58 | 58 | A |
| 36 | 20 | 12 | 4 | 58 | 58 | 58 | A |

>> C3: Did any ambiguities in the scheduler specification make values

>> in the table uncertain? If so, what rule did you use to resolve

>> them? Does this match the behavior of your scheduler?

存在歧义——例如上表timer ticks为28时此时A的优先级降低，优先级最高的有B、C此时应该选择哪个进程运行，对此根据文档对同优先级的线程采用循环调度，实际上也与调度程序相符合。Q循环中排序顺序是不确定的，(到底跟不根据入队顺序确定运行的线程？)

>> C4: How is the way you divided the cost of scheduling between code

>> inside and outside interrupt context likely to affect performance?

中断切换上下文要保证原子操作就是要进行关闭外中断，而如果原子操作过长导致关闭外中断时间过长就会影响到操作系统的性能，同时在中断上下文的优先级是高于进程上下文的，无论对于用户进程还是内核线程，一般情况下如果中断发生，都会造成中断上下文抢占当前进程上下文执行的，一般都会把调度放在中断外进行，即内核中断中不进行调度。

---- RATIONALE ----

>> C5: Briefly critique your design, pointing out advantages and

>> disadvantages in your design choices. If you were to have extra

>> time to work on this part of the project, how might you choose to

>> refine or improve your design?

根据指导手册一步一步实现有关计算nice值、recent\_cpu值、load\_avg值与priority值的函数方法以及细节，另外实现pintos模拟浮点数的运算。

不更改timer.c中的timer\_interrupt()函数而将判断是否是mlfqs调度放到thread\_tick()中

优点：

1、对于nice为负值的线程，其recent\_cpu也可能为负值，如果为负值，优先级更新会超过PRI\_MAX(63)，调度会产生错误，所以在更新优先级函数中对优先级溢出部分做了限定。

/\* 更新单个线程的优先级 \*/

void

renew\_priority(struct thread\* t){

/\* priority = PRI\_MAX - (recent\_cpu / 4) - (nice \* 2) \*/

t->priority = PRI\_MAX - FLOAT2INTPART(FLOATDIVINT(t->recent\_cpu, 4)) - (t->nice \* 2);

if(t->priority > PRI\_MAX)

t->priority = PRI\_MAX;

else if(t->priority < PRI\_MIN)

t->priority = PRI\_MIN;

}

缺点：

对于list\_sort排序使用的是默认归并排序方法，没有处理优先级相同情况。

>> C6: The assignment explains arithmetic for fixed-point math in

>> detail, but it leaves it open to you to implement it. Why did you

>> decide to implement it the way you did? If you created an

>> abstraction layer for fixed-point math, that is, an abstract data

>> type and/or a set of functions or macros to manipulate fixed-point

>> numbers, why did you do so? If not, why not?

通过位运算模拟浮点数运算，同时声明宏与一个int64\_t数据类型来完成浮点数运算

typedef int64\_t fixed\_t;

#define SHIFT\_AMOUNT 14

#define INT2FLOAT(n) ((fixed\_t)(n << SHIFT\_AMOUNT))

#define FLOAT2INTPART(x) (x >> SHIFT\_AMOUNT)

#define FLOAT2INTNEAR(x) (x >= 0 ? ((x + (1 << (SHIFT\_AMOUNT - 1))) >> SHIFT\_AMOUNT) : ((x - (1 << SHIFT\_AMOUNT)) >> SHIFT\_AMOUNT))

#define FLOATADDFLOAT(x, y) (x + y)

#define FLOATSUBFLOAT(x, y) (x - y)

#define FLOATADDINT(x, n) (x + (n << SHIFT\_AMOUNT))

#define FLOATSUBINT(x, n) (x - (n << SHIFT\_AMOUNT))

#define FLOATMULFLOAT(x, y) ((((int64\_t)x) \* y) >> SHIFT\_AMOUNT)

#define FLOATMULINT(x, n) (x \* n)

#define FLOATDIVFLOAT(x, y) ((((int64\_t)x) << SHIFT\_AMOUNT) / y)

#define FLOATDIVINT(x, n) (x / n)

首先定义宏来进行简化指令，方便代码书写，再者由于只涉及加减与位运算，篇幅较小，满足宏定义的要求，并且一定程度上能提高程序运行效率。

SURVEY QUESTIONS

================

Answering these questions is optional, but it will help us improve the

course in future quarters. Feel free to tell us anything you

want--these questions are just to spur your thoughts. You may also

choose to respond anonymously in the course evaluations at the end of

the quarter.

>> In your opinion, was this assignment, or any one of the three problems

>> in it, too easy or too hard? Did it take too long or too little time?

>> Did you find that working on a particular part of the assignment gave

>> you greater insight into some aspect of OS design?

>> Is there some particular fact or hint we should give students in

>> future quarters to help them solve the problems? Conversely, did you

>> find any of our guidance to be misleading?

>> Do you have any suggestions for the TAs to more effectively assist

>> students, either for future quarters or the remaining projects?

>> Any other comments?