四 川 大 学
操作系统实验课程
--Nachos 部分实验报告

学　　院：_____

专　　业：_____

年　　级：_____

组　　员：_____

_____

_____

指导教师评阅意见：_____

_____

_____

指导教师评阅成绩：_____

提交时间　2023 年　11 月　21 日

实验项目二：Nachos 进程管理

2.1 实验目的：

完成 Nachos 的进程管理模块的扩展，掌握操作系统中进程管理和进程调度扩展实现。

2.2 实验环境：

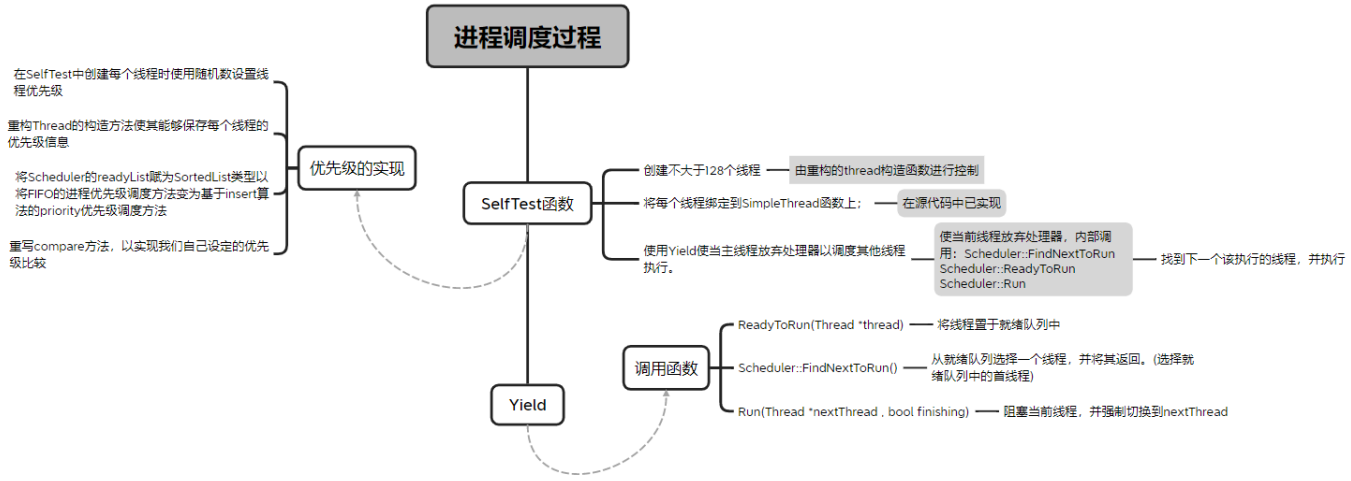
描述所用实验环境，包括计算机硬件和软件资源的情况，如选用的操作系统、机器配置、编译器版本等。

选用的操作系统： VMware 16.0 -Ubuntu Linux 22.04.2 -NachOS-4.1

2.3 实验内容：

对实践过程的详细说明，如对 Nachos 平台的哪些代码进行了什么样的修改，采用了何种算法或思想等。

List 核心的代码，如以文件为单位一一进行描述。



1. **扩展 Nachos 线程管理模式，限制线程的数量为最多 128 个：**

（1） 定义一个宏，标记最大数量：128（MAX_THREAD_NUM）

```
// The maximum number of threads
#define MAX_THREAD_NUM 128
```

（2） 修改 Thread 创建初始化函数，使其最多创建不超过 MAX_THREAD_NUM 个

```
Thread::Thread(char *threadName, int p) {
    // Assert if the {threadNum} has exceeded the max number.
    if (++threadNum > MAX_THREAD_NUM) {
        cout << "最多只能同时创建" << MAX_THREAD_NUM << "个线程！！" << endl;
        ASSERT(threadNum <= MAX_THREAD_NUM);
    }
    cout << "Create Thread " << threadNum << endl;

    // Initialize a new thread.
    name = threadName;
    priority = p;
    stackTop = NULL;
    stack = NULL;
    status = JUST_CREATED;
    setPriority(p);
    for (int i = 0; i < MachineStateSize; i++) {
        machineState[i] = NULL;
    }
    space = NULL;
}
```

2. **修改扩充 Nachos 的线程调度机制，改为“优先级调度”的抢占式调度**

（1） 修改 Scheduler 中使用的数据结构，从 list 修改为 SortedList

```
Scheduler::Scheduler() {
    readyList = new SortedList<Thread *>(Compare);
    toBeDestroyed = NULL;
}
```

（2） 修改 Scheduler 中 SortedList 排序规则：
按照优先级 priority 进行排序

```
static int Scheduler::Compare(Thread *x, Thread *y) {
    if (x->getPriority() > y->getPriority())
        return -1;
    else if (x->getPriority() == y->getPriority())
        return 0;
    else
        return 1;
}
```

（3） 同时将之后加入 List 的方法从 append，修改为 insert，使链表按照规则插入，而非尾插；
使调度策略从原先的 FIFO，变为现在的按照优先级进行排序

```
void
Scheduler::ReadyToRun(Thread *thread) {
    ASSERT(kernel->interrupt->getLevel() == IntOff);
    DEBUG(dbgThread, "Putting thread on ready list: " << thread->getName());

    thread->setStatus( st: READY);
    readyList->Insert(thread);
}
```

（4） 修改 Yield 函数：根据 Scheduler 规则来决定下一个要执行的线程

```
void
Thread::Yield() {
    Thread *nextThread;
    IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);

    ASSERT(this == kernel->currentThread);

    DEBUG(dbgThread, "Yielding thread: " << name);

    // nextThread = kernel->scheduler->FindNextToRun();
    // if (nextThread != NULL) {
    kernel->scheduler->ReadyToRun( thread: this);
    nextThread = kernel->scheduler->FindNextToRun();
    kernel->scheduler->Run(nextThread, FALSE);
    // }
    (void) kernel->interrupt->SetLevel(oldLevel);
}
```

2.4 实验结果：

截图说明运行效果，并且辅助相关描述信息。

1. **限制线程的数量运行结果：**

```
filtee@iZ2vcbcz19gseqovr8jso8Z:~/Desktop/NachOS/code/build.linux$ ./nachos -K


tests summary: ok:0
Create Thread 1
Create Thread 2
Create Thread 3
Create Thread 4
Create Thread 5
Create Thread 6
Create Thread 7
Create Thread 8
Create Thread 9
Create Thread 10
Create Thread 11
Create Thread 12
Create Thread 13
Create Thread 14
Create Thread 15
Create Thread 16
Create Thread 17
Create Thread 18
Create Thread 19
Create Thread 20
Create Thread 21
Create Thread 22
Create Thread 23
Create Thread 24
Create Thread 25
```

```
Create Thread 103
Create Thread 104
Create Thread 105
Create Thread 106
Create Thread 107
Create Thread 108
Create Thread 109
Create Thread 110
Create Thread 111
Create Thread 112
Create Thread 113
Create Thread 114
Create Thread 115
Create Thread 116
Create Thread 117
Create Thread 118
Create Thread 119
Create Thread 120
Create Thread 121
Create Thread 122
Create Thread 123
Create Thread 124
Create Thread 125
Create Thread 126
Create Thread 127
Create Thread 128
最多只能同时创建128个线程！！
Assertion failed: line 57 file ../threads/thread.cc
Aborted (core dumped)
filtee@iZ2vcbcz19gseqovr8jso8Z:~/Desktop/NachOS/code/build.linux$
```

2. 修改扩充 Nachos 的线程调度机制，改为“优先级调度”的抢占式调度运行结果

```
/lib/cpp  -P -I../network -I../filesys -I../userprog -I../threads -I../machine
switch.s > swtch.s
as --32 -o switch.o swtch.s
g++ -m32 bitmap.o debug.o libtest.o sysdep.o interrupt.o stats.o timer.o conso
o network.o disk.o alarm.o kernel.o main.o scheduler.o synch.o thread.o addrsp
rectory.o filehdr.o filesys.o pbitmap.o openfile.o synchdisk.o post.o switch.o
filtee@iZ2vcbcz19gseqovr8jso8Z:~/Desktop/NachOS/code/build.linux$ ./nachos -K


tests summary: ok:0
UserThread:1 priority:2
Create Thread 1
UserThread:2 priority:6
Create Thread 2
UserThread:3 priority:7
Create Thread 3
UserThread:4 priority:7
Create Thread 4
UserThread:5 priority:4
Create Thread 5
UserThread:6 priority:2
Create Thread 6
*** thread 3 looped 0 times
*** thread 4 looped 0 times
*** thread 4 looped 1 times
*** thread 3 looped 1 times
*** thread 4 looped 2 times
*** thread 3 looped 2 times
*** thread 4 looped 3 times
*** thread 3 looped 3 times
*** thread 4 looped 4 times
```

```
*** thread 4 looped 4 times
*** thread 3 looped 4 times
*** thread 2 looped 0 times
*** thread 2 looped 1 times
*** thread 2 looped 2 times
*** thread 2 looped 3 times
*** thread 2 looped 4 times
*** thread 5 looped 0 times
*** thread 5 looped 1 times
*** thread 5 looped 2 times
*** thread 5 looped 3 times
*** thread 5 looped 4 times
*** thread 1 looped 0 times
*** thread 6 looped 0 times
*** thread 1 looped 1 times
*** thread 6 looped 1 times
*** thread 1 looped 2 times
*** thread 1 looped 3 times
*** thread 6 looped 2 times
*** thread 1 looped 4 times
*** thread 6 looped 3 times
*** thread 6 looped 4 times
Machine halting!

Ticks: total 2580, idle 0, system 2580, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
filtee@iZ2vcbcz19gseqovr8jso8Z:~/Desktop/NachOS/code/build.linux$
```

2.5 实验总结：

总结本实验的完成情况，包括代码是否编写完成，是否调试通过，能否正常运行，实现了实验要求中要求的哪些项（对实验要求的满足程度；）；

总结本实验所涉及到的知识点。

代码经过修改已经编写完成，调试通过，成功正常运行。扩展 Nachos 线程管理模式，限制线程数量为最多 128 个

并修改扩充 Nachos 的线程调度机制，改为“优先级调度”的抢占式调度.

完成 Nachos 进程管理模块的扩展需要涉及以下操作系统知识点有：

1. 进程管理：了解操作系统中进程的概念和生命周期，包括进程的创建、销毁和状态转换等。
2. 进程调度：理解操作系统中进程调度的概念和方法，包括调度算法、优先级等。
3. 进程同步：了解操作系统中进程同步的概念和方法，包括临界区、信号量、互斥量等。
4. Nachos 操作系统：熟悉 Nachos 操作系统的体系结构和实现原理，包括线程模型、虚拟内存、文件系统等。
5. C++编程：具备 C++编程语言的基本知识，能够进行面向对象的程序设计和开发。
6. 调试技能：掌握常用的调试技巧和工具，能够快速定位和解决程序中的问题。