OS_report2_syscall & CPU scheduling

生醫電資所碩一 R07945029 王思敏

Part1. System call Sleep

Sleep function:

```
#include "syscall.h"

main()
        {
                int     n;
                for (n=0;n<=5;n++)
                {
                        Sleep(1000000);
                        PrintInt(n);
                }
                return 0;
        }
```

首先是在 syscall.h 裡宣告 sleep.c 的 test code 裡會用到的 Sleep()函式，由於除了等待(睡眠)的部分都是需要顯示在終端機上的印出部分，所以跟著 PrintInt 一同更改 start.s，宣告名稱為 Sleep 將 SC_Sleep 存至 register2，system call 的參數則會存在 register4, 5, 6, 7，宣告例外操作的 case SC_Sleep，在 nachos 處理例外操作的部分(exception.cc)加上 SC_Sleep 的 case，印出 Sleep Time (val) (ms)，多加上 kernel->alarm->WaitUntil (val)叫出 WaitUntil 讓其等待 val 的時間(val = 1000000 micro second)。

```
  case SC_PrintInt:
          val=kernel->machine->ReadRegister(4);
          cout << "Print integer:" <<val << endl;
          return;
  case SC_Sleep:
          val=kernel->machine->ReadRegister(4);
          cout << "Sleep Time " << val << "(ms) " << endl;
          kernel->alarm->WaitUntil(val);
          return;
```

故當 Sleep()被呼叫時，WaitUntil 亦會被呼叫，並進入睡眠(put to bed)，並根據從 Sleep 被傳給 WaitUntil 的 val 值決定睡眠多久，即是醒來的時間為 _current_interrupt + val。

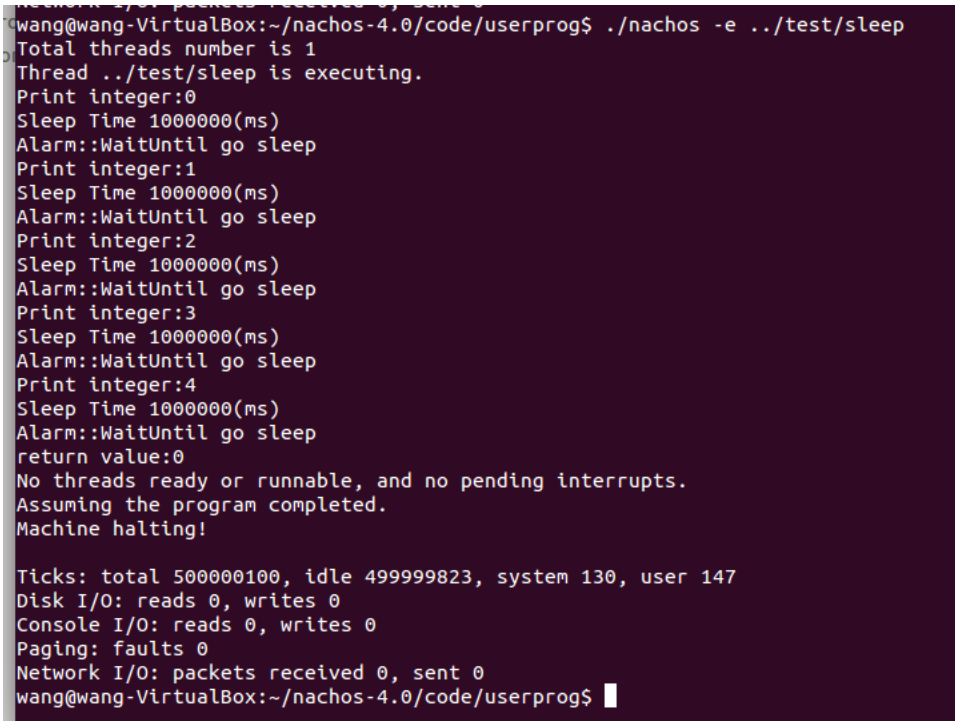
```
void
Alarm::WaitUntil(int x) {
    IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
    Thread* t = kernel->currentThread;

    // burst time
    int worktime = kernel->stats->userTicks - t->getStartTime();
    t->setBurstTime(t->getBurstTime() + worktime);
    t->setStartTime(kernel->stats->userTicks);
    cout << "Alarm::WaitUntil go sleep" << endl;
    _bedroom.PutToBed(t, x);
    kernel->interrupt->SetLevel(oldLevel);
}

bool Bedroom::IsEmpty() {
    return _beds.size() == 0;
}

void Bedroom::PutToBed(Thread*t, int x) {
    ASSERT(kernel->interrupt->getLevel() == IntOff);
    _beds.push_back(Bed(t, _current_interrupt + x));
    t->Sleep(false);
}
```

由於 kernel 裡存有 alarm，所以每當 hardware timer 產生中斷系統便會去 callback()裡，CallBack 中會呼叫 MorningCall 做_current_interrupt 加 1 微秒的動作，如果此時_current_interrupt 達到 current_interrupt 加上 val 的值(時間)則回傳 woken=True 叫醒程序，反之則回傳 woken=False 繼續睡。

```
void
Alarm::CallBack()
{
    Interrupt *interrupt = kernel->interrupt;
    MachineStatus status = interrupt->getStatus();
    bool woken = _bedroom.MorningCall();

    kernel->currentThread->setPriority(kernel->currentThread->getPriority() - 1);

    if (status == IdleMode && !woken && _bedroom.IsEmpty()) {// is it time to quit?
        if (!interrupt->AnyFutureInterrupts()) {
            timer->Disable();   // turn off the timer
        }
    } else {                    // there's someone to preempt
        if(kernel->scheduler->getSchedulerType() == RR ||
                kernel->scheduler->getSchedulerType() == Priority ) {
//              interrupt->YieldOnReturn();
//              cout << "=== interrupt->YieldOnReturn ===" << endl;
                interrupt->YieldOnReturn();
        }
    }
}

bool Bedroom::MorningCall() {
    bool woken = false;

    _current_interrupt ++;

    for(std::list<Bed>::iterator it = _beds.begin();
        it != _beds.end(); ) {
        if(_current_interrupt >= it->when) {
            woken = true;
//            cout << "Bedroom::MorningCall Thread woken" << endl;
            kernel->scheduler->ReadyToRun(it->sleeper);
            it = _beds.erase(it);
        } else {
            it++;
        }
    }
    return woken;
}
```

system call Sleep()執行結果:

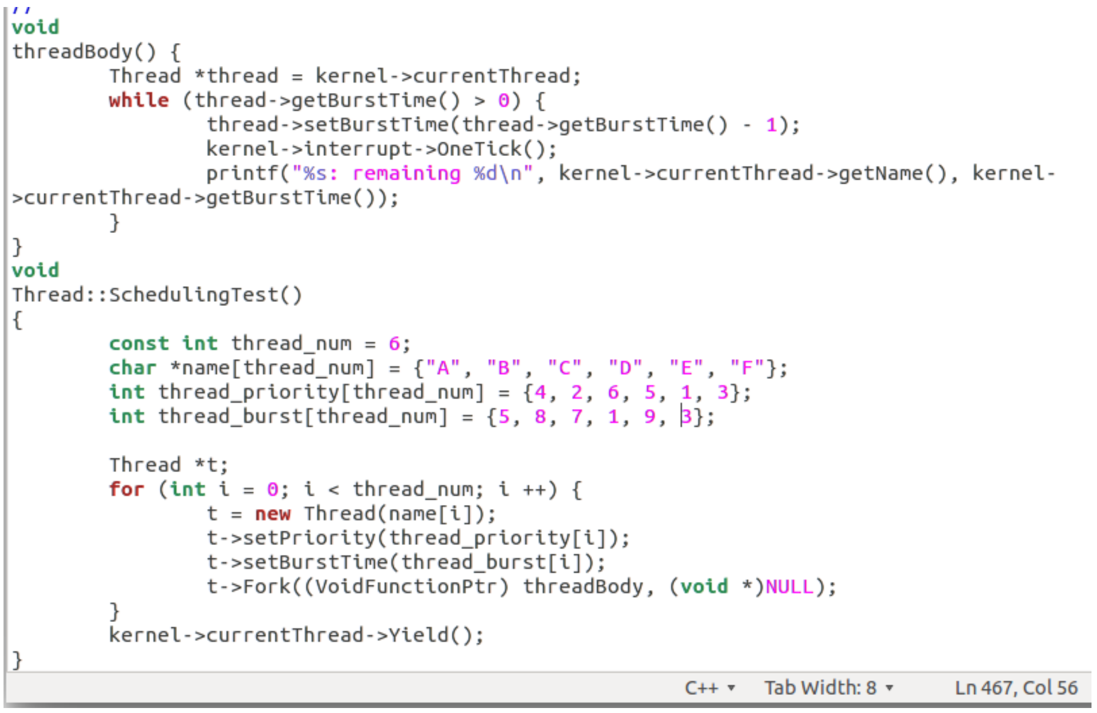
根據前面 sleep 測試程式的架構



```
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos -e ../test/sleep
Total threads number is 1
Thread ../test/sleep is executing.
Print integer:0
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:1
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:2
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:3
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:4
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500000100, idle 499999823, system 130, user 147
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$
```

Part2. CPU scheduling

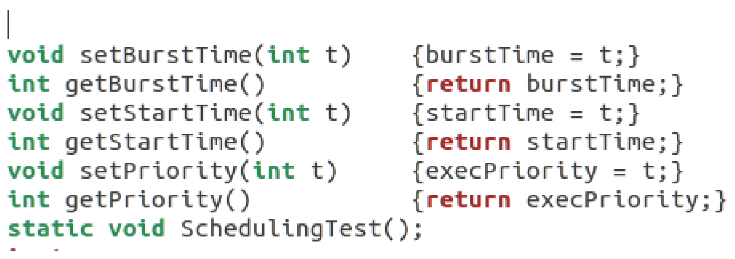
首先在 thread.cc 裡寫 selftest code，設定執行程序名稱、優先權、burst time，thread body 依照排成的優先序執行程序，執行過的程序 burst time-1 並顯示來表示執行順序與過程。



```
//
void
threadBody() {
        Thread *thread = kernel->currentThread;
        while (thread->getBurstTime() > 0) {
                thread->setBurstTime(thread->getBurstTime() - 1);
                kernel->interrupt->OneTick();
                printf("%s: remaining %d\n", kernel->currentThread->getName(), kernel-
>currentThread->getBurstTime());
        }
}
void
Thread::SchedulingTest()
{
        const int thread_num = 6;
        char *name[thread_num] = {"A", "B", "C", "D", "E", "F"};
        int thread_priority[thread_num] = {4, 2, 6, 5, 1, 3};
        int thread_burst[thread_num] = {5, 8, 7, 1, 9, 3};

        Thread *t;
        for (int i = 0; i < thread_num; i ++) {
                t = new Thread(name[i]);
                t->setPriority(thread_priority[i]);
                t->setBurstTime(thread_burst[i]);
                t->Fork((VoidFunctionPtr) threadBody, (void *)NULL);
        }
        kernel->currentThread->Yield();
}
```

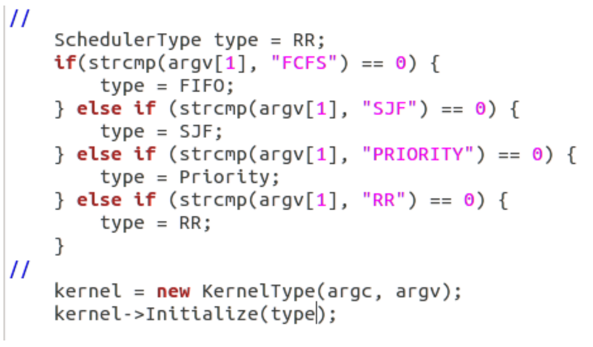
C++ ▾   Tab Width: 8 ▾   Ln 467, Col 56

thread.h 裡宣告 setBurstTime, getBurstTime.....的值及回傳值。



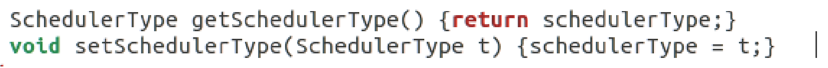
```
void setBurstTime(int t)     {burstTime = t;}
int getBurstTime()           {return burstTime;}
void setStartTime(int t)     {startTime = t;}
int getStartTime()           {return startTime;}
void setPriority(int t)      {execPriority = t;}
int getPriority()            {return execPriority;}
static void SchedulingTest();
```

接著將 threads 裡的 main.cc 檔修改成可以選擇排程方式的形式，預設是 RR，如果有輸入指令且與 FCFS, SJF, PRIORITY, RR 相等則會回傳 0，經判斷確認後將 type 設定為回傳值為 0 的那項(FCFS or SJF or PRIORITY or RR)。



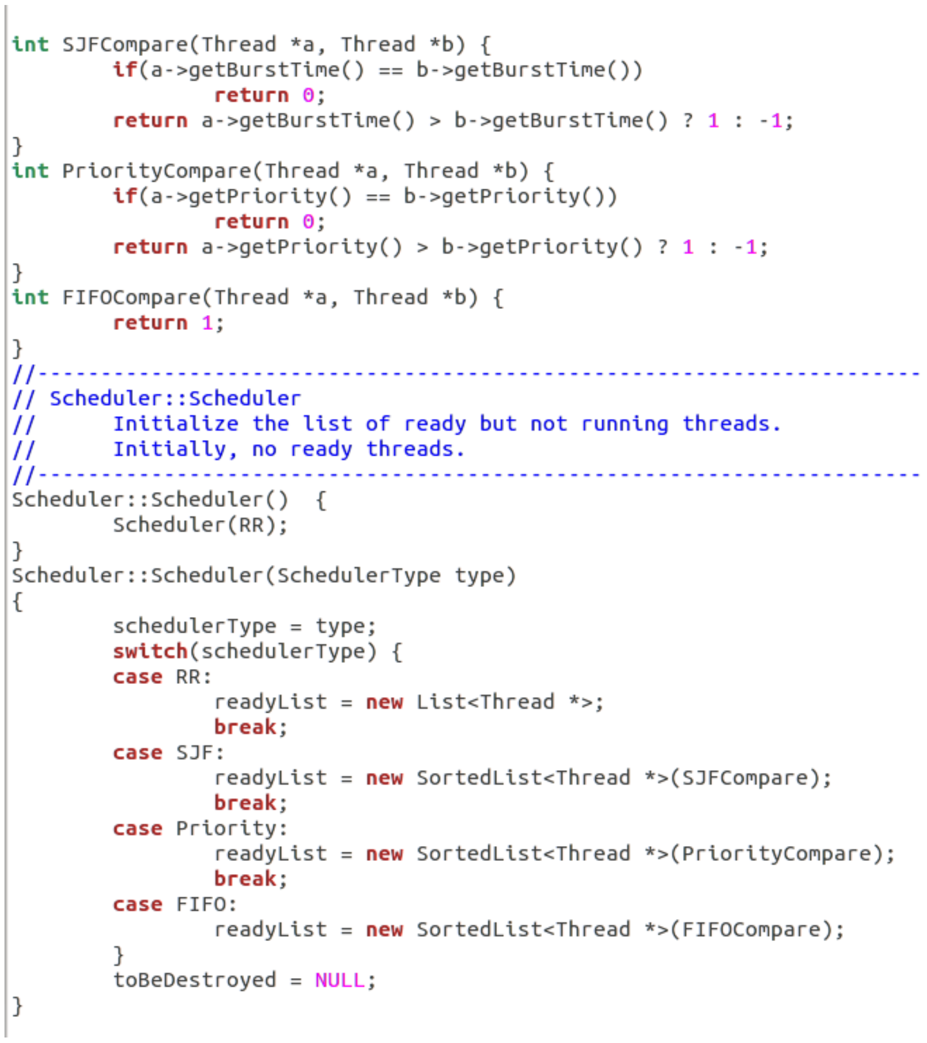
```
//
    SchedulerType type = RR;
    if(strcmp(argv[1], "FCFS") == 0) {
        type = FIFO;
    } else if (strcmp(argv[1], "SJF") == 0) {
        type = SJF;
    } else if (strcmp(argv[1], "PRIORITY") == 0) {
        type = Priority;
    } else if (strcmp(argv[1], "RR") == 0) {
        type = RR;
    }
//
    kernel = new KernelType(argc, argv);
    kernel->Initialize(type);
```

在 scheduler.h 裡宣告抓取讀到的 schedulerType type。



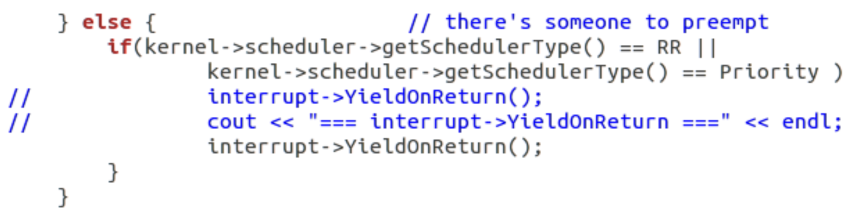
```
SchedulerType getSchedulerType() {return schedulerType;}
void setSchedulerType(SchedulerType t) {schedulerType = t;}
```

根據 type 決定要用哪一種的排程，並宣告相對應的 compare 函式，以決定行程執行方式順序。



```
int SJFCompare(Thread *a, Thread *b) {
        if(a->getBurstTime() == b->getBurstTime())
                return 0;
        return a->getBurstTime() > b->getBurstTime() ? 1 : -1;
}
int PriorityCompare(Thread *a, Thread *b) {
        if(a->getPriority() == b->getPriority())
                return 0;
        return a->getPriority() > b->getPriority() ? 1 : -1;
}
int FIFOCompare(Thread *a, Thread *b) {
        return 1;
}
//----------------------------------------------------------------------
// Scheduler::Scheduler
//      Initialize the list of ready but not running threads.
//      Initially, no ready threads.
//----------------------------------------------------------------------
Scheduler::Scheduler()  {
        Scheduler(RR);
}
Scheduler::Scheduler(SchedulerType type)
{
        schedulerType = type;
        switch(schedulerType) {
        case RR:
                readyList = new List<Thread *>;
                break;
        case SJF:
                readyList = new SortedList<Thread *>(SJFCompare);
                break;
        case Priority:
                readyList = new SortedList<Thread *>(PriorityCompare);
                break;
        case FIFO:
                readyList = new SortedList<Thread *>(FIFOCompare);
        }
        toBeDestroyed = NULL;
}
```

最後，從 alarm 裡的 callback 判斷如果式 RR 或 PRIORITY 的排程則呼叫 interrupt->YieldOnReturn()以查看是否有更需優先執行的行程。

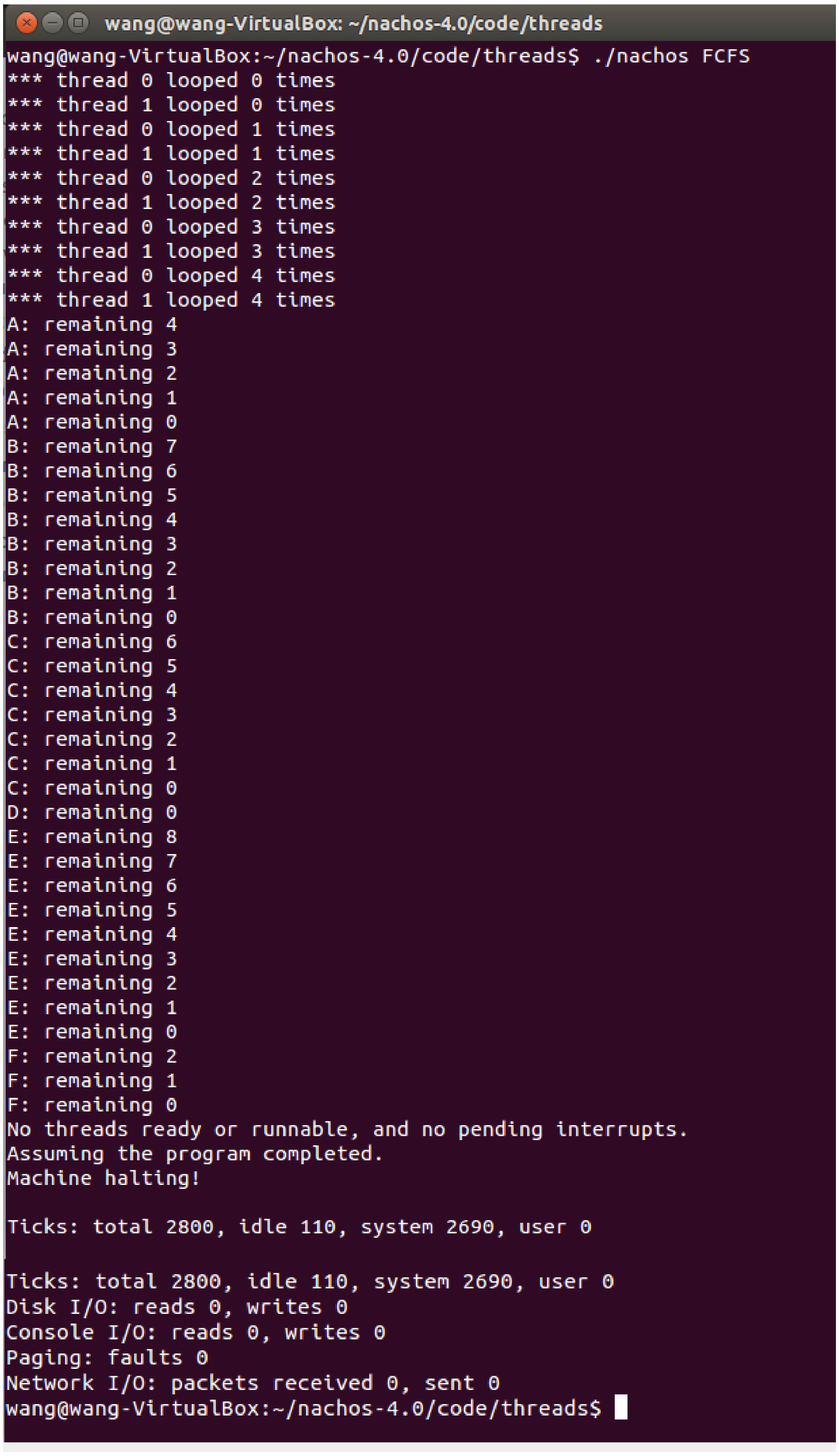


```
    } else {                    // there's someone to preempt
        if(kernel->scheduler->getSchedulerType() == RR ||
                kernel->scheduler->getSchedulerType() == Priority ) {
//              interrupt->YieldOnReturn();
//              cout << "=== interrupt->YieldOnReturn ===" << endl;
                interrupt->YieldOnReturn();
        }
    }
```

CPU scheduling 執行結果:

Selftest—

FCFS　　　於 threads 下執行 ./nachos FCFS

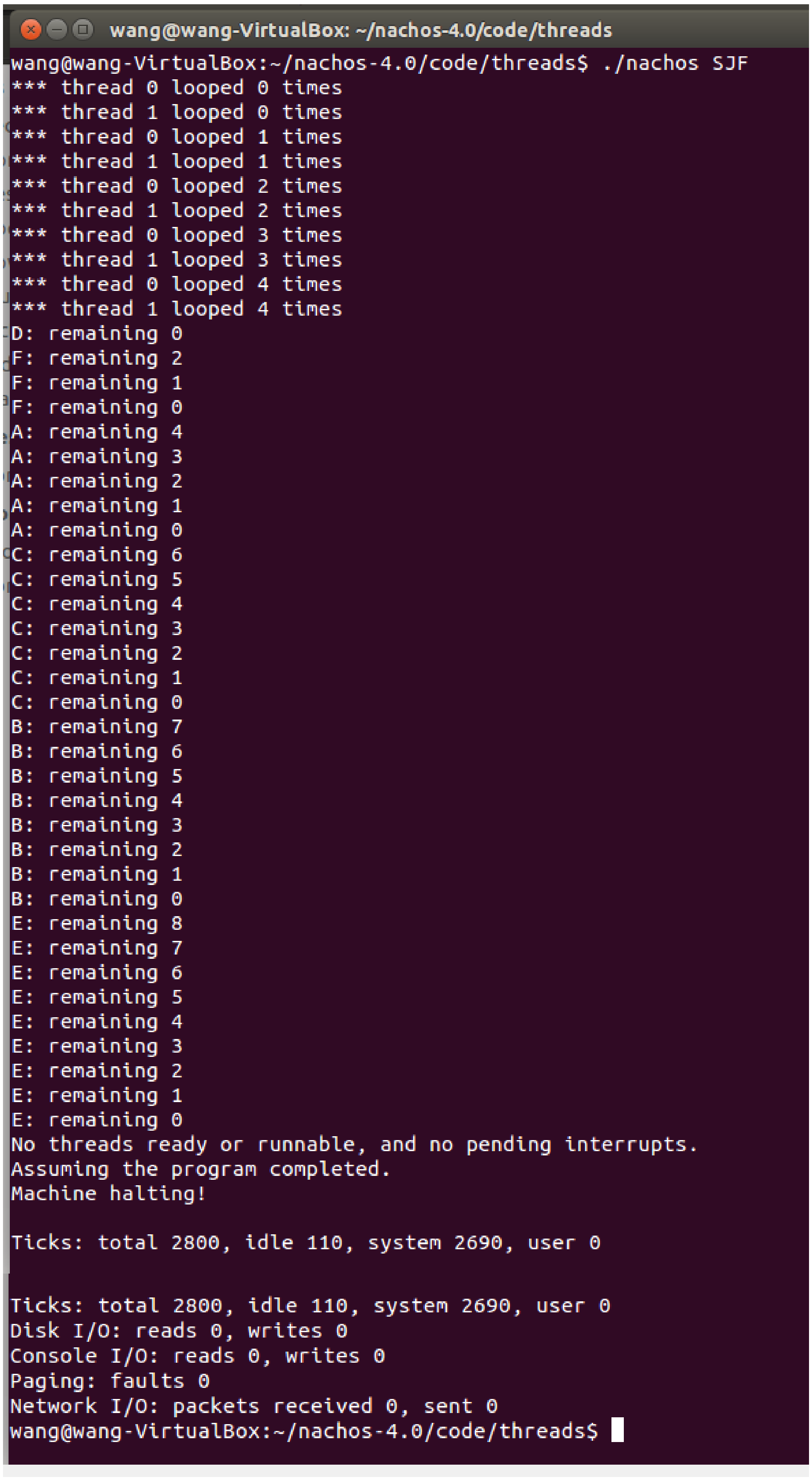


wang@wang-VirtualBox: ~/nachos-4.0/code/threads

```
wang@wang-VirtualBox:~/nachos-4.0/code/threads$ ./nachos FCFS
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
A: remaining 4
A: remaining 3
A: remaining 2
A: remaining 1
A: remaining 0
B: remaining 7
B: remaining 6
B: remaining 5
B: remaining 4
B: remaining 3
B: remaining 2
B: remaining 1
B: remaining 0
C: remaining 6
C: remaining 5
C: remaining 4
C: remaining 3
C: remaining 2
C: remaining 1
C: remaining 0
D: remaining 0
E: remaining 8
E: remaining 7
E: remaining 6
E: remaining 5
E: remaining 4
E: remaining 3
E: remaining 2
E: remaining 1
E: remaining 0
F: remaining 2
F: remaining 1
F: remaining 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 2800, idle 110, system 2690, user 0

Ticks: total 2800, idle 110, system 2690, 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
wang@wang-VirtualBox:~/nachos-4.0/code/threads$
```

由 A 執行到 F，與預期相同

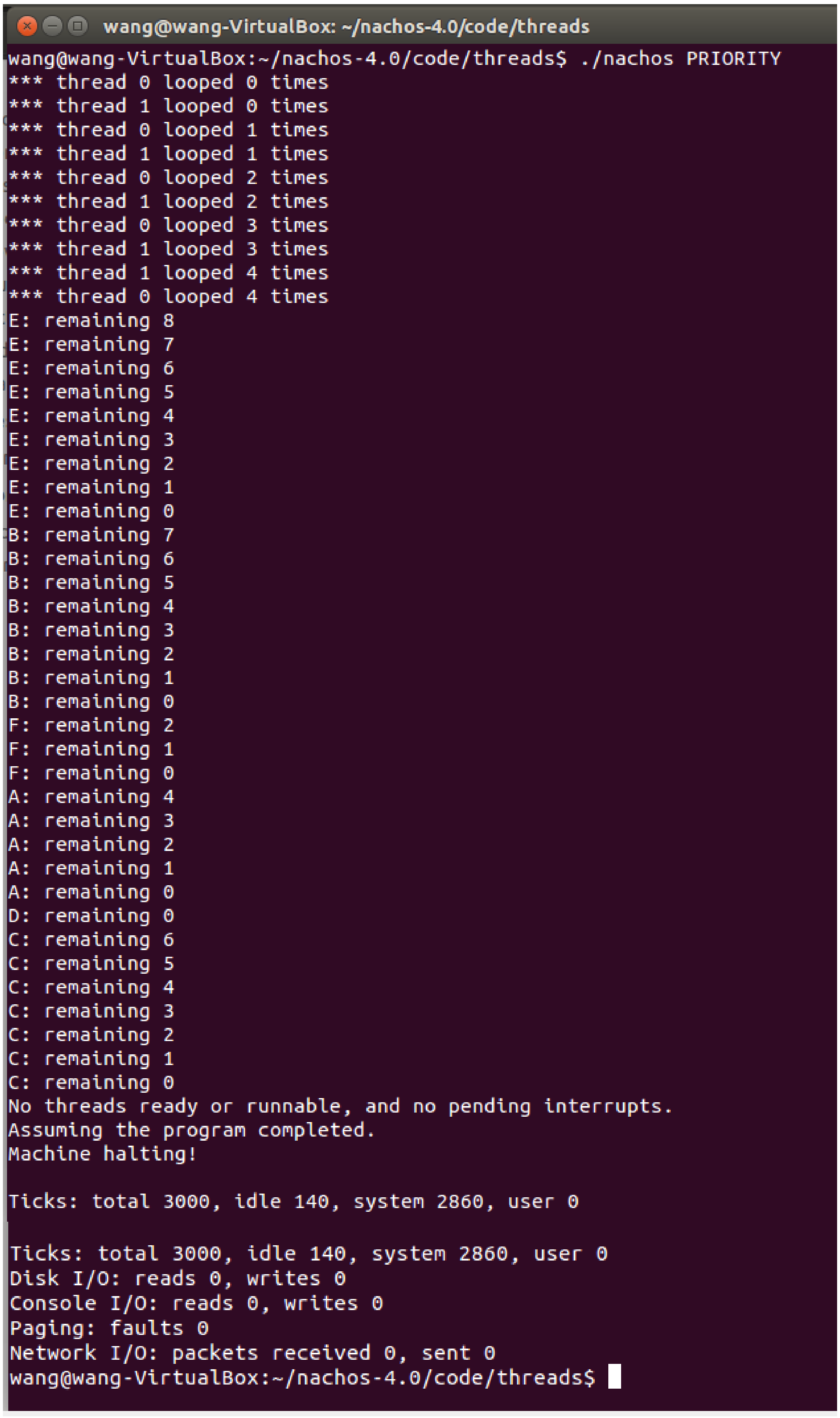
SJF　　　於 threads 下執行 ./nachos SJF


wang@wang-VirtualBox: ~/nachos-4.0/code/threads

```
wang@wang-VirtualBox:~/nachos-4.0/code/threads$ ./nachos SJF
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
D: remaining 0
F: remaining 2
F: remaining 1
F: remaining 0
A: remaining 4
A: remaining 3
A: remaining 2
A: remaining 1
A: remaining 0
C: remaining 6
C: remaining 5
C: remaining 4
C: remaining 3
C: remaining 2
C: remaining 1
C: remaining 0
B: remaining 7
B: remaining 6
B: remaining 5
B: remaining 4
B: remaining 3
B: remaining 2
B: remaining 1
B: remaining 0
E: remaining 8
E: remaining 7
E: remaining 6
E: remaining 5
E: remaining 4
E: remaining 3
E: remaining 2
E: remaining 1
E: remaining 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 2800, idle 110, system 2690, user 0

Ticks: total 2800, idle 110, system 2690, 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
wang@wang-VirtualBox:~/nachos-4.0/code/threads$
```

由工作最短執行到最長與預期相同

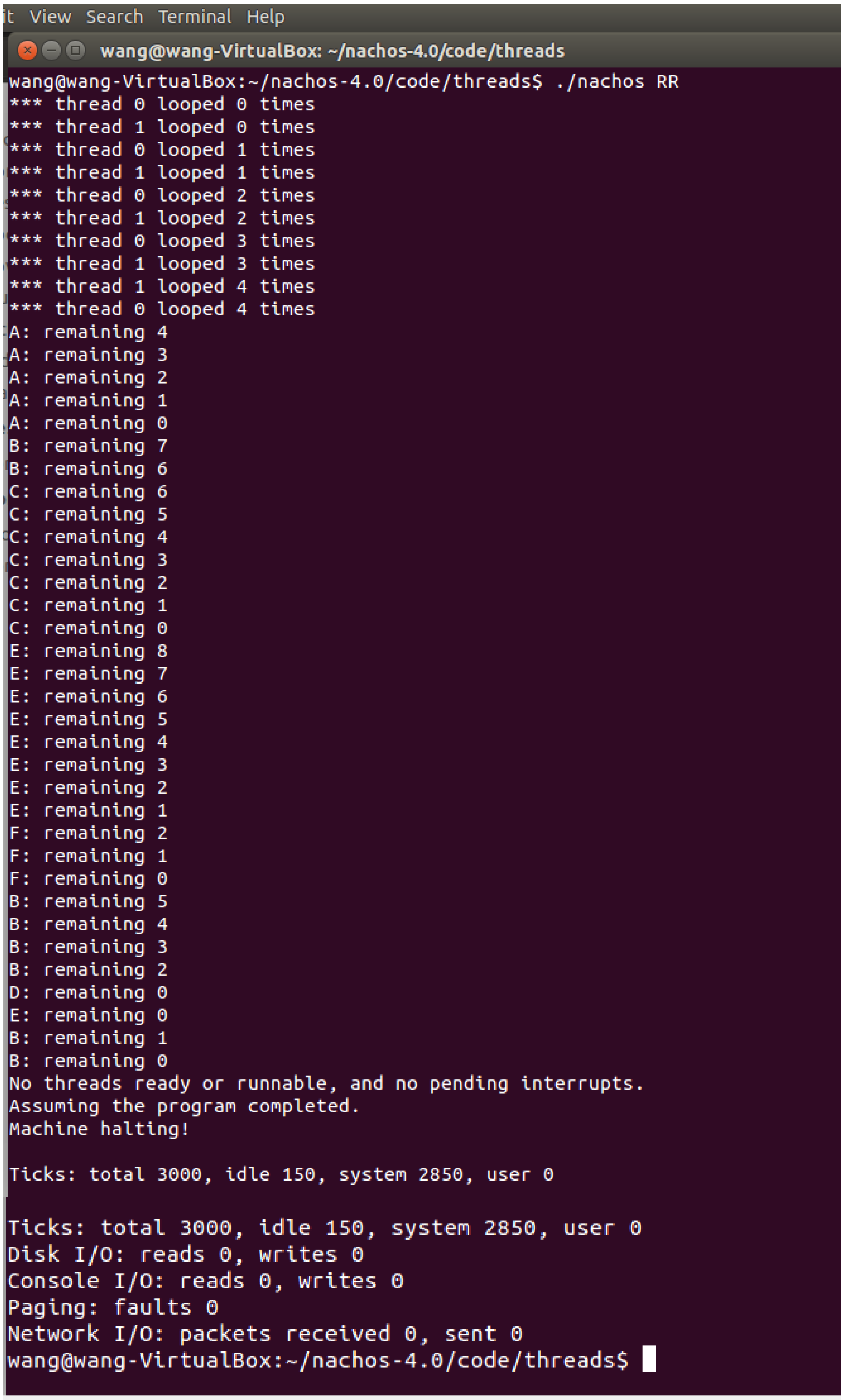
PRIORITY　　於 threads 下執行 ./nachos PRIORITY



```
wang@wang-VirtualBox:~/nachos-4.0/code/threads$ ./nachos PRIORITY
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 1 looped 4 times
*** thread 0 looped 4 times
E: remaining 8
E: remaining 7
E: remaining 6
E: remaining 5
E: remaining 4
E: remaining 3
E: remaining 2
E: remaining 1
E: remaining 0
B: remaining 7
B: remaining 6
B: remaining 5
B: remaining 4
B: remaining 3
B: remaining 2
B: remaining 1
B: remaining 0
F: remaining 2
F: remaining 1
F: remaining 0
A: remaining 4
A: remaining 3
A: remaining 2
A: remaining 1
A: remaining 0
D: remaining 0
C: remaining 6
C: remaining 5
C: remaining 4
C: remaining 3
C: remaining 2
C: remaining 1
C: remaining 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 3000, idle 140, system 2860, user 0

Ticks: total 3000, idle 140, system 2860, 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
wang@wang-VirtualBox:~/nachos-4.0/code/threads$
```

根據優先序執行，與預期相同

RR　　　　於 threads 下執行 ./nachos RR


it View Search Terminal Help
wang@wang-VirtualBox: ~/nachos-4.0/code/threads

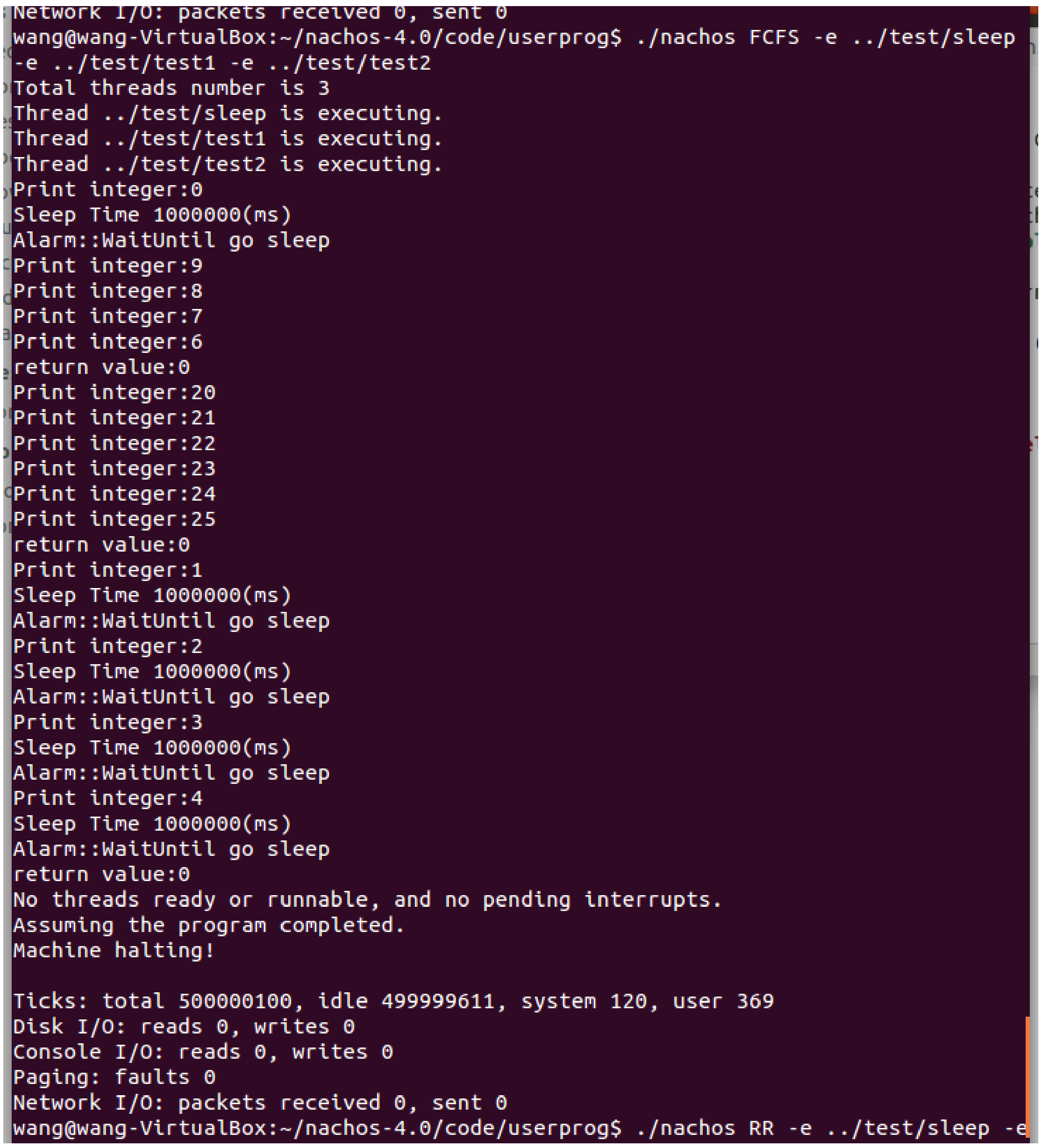
```
wang@wang-VirtualBox:~/nachos-4.0/code/threads$ ./nachos RR
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 1 looped 4 times
*** thread 0 looped 4 times
A: remaining 4
A: remaining 3
A: remaining 2
A: remaining 1
A: remaining 0
B: remaining 7
B: remaining 6
C: remaining 6
C: remaining 5
C: remaining 4
C: remaining 3
C: remaining 2
C: remaining 1
C: remaining 0
E: remaining 8
E: remaining 7
E: remaining 6
E: remaining 5
E: remaining 4
E: remaining 3
E: remaining 2
E: remaining 1
F: remaining 2
F: remaining 1
F: remaining 0
B: remaining 5
B: remaining 4
B: remaining 3
B: remaining 2
D: remaining 0
E: remaining 0
B: remaining 1
B: remaining 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 3000, idle 150, system 2850, user 0

Ticks: total 3000, idle 150, system 2850, 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
wang@wang-VirtualBox:~/nachos-4.0/code/threads$
```

原本預期是 A, B, C, D, E, F, A, B....，但出來不為此，猜測可能是因為執行時間很短一次不會只執行一個。

同時執行多個 test file (test1, test2, sleep)—

FCFS

於 userprog 下執行 ./nachos FCFS –e ../test/sleep –e ../test/test1 –e ../test/test1

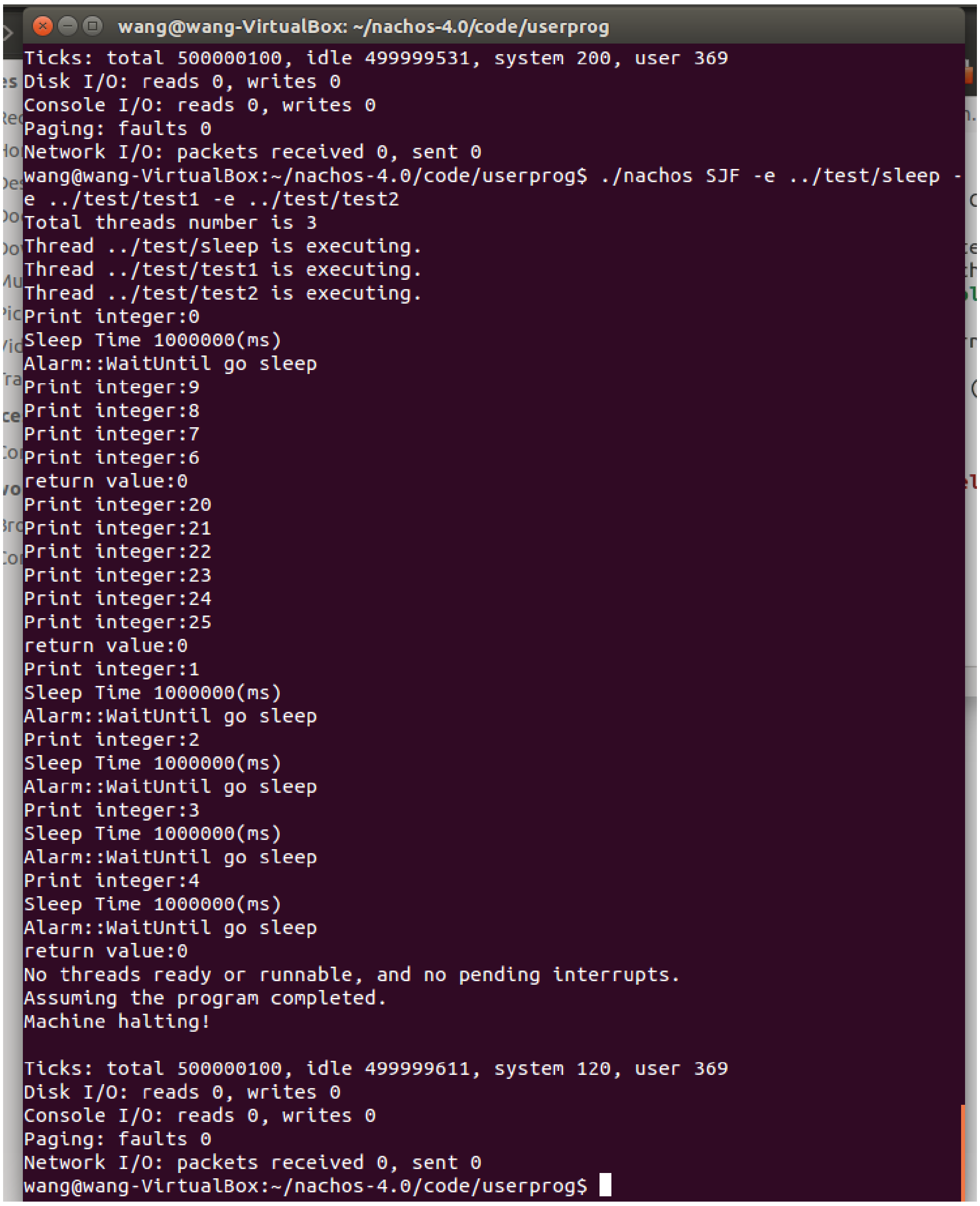


```
Network I/O: packets received 0, sent 0
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos FCFS -e ../test/sleep
-e ../test/test1 -e ../test/test2
Total threads number is 3
Thread ../test/sleep is executing.
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:0
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
Print integer:1
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:2
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:3
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:4
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500000100, idle 499999611, system 120, user 369
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos RR -e ../test/sleep -e
```

預期應該是先執行 sleep，再執行 test1，再 test2，但不是，可能是因為 sleep 執行會太久導致。

SJF

於 userprog 下執行 ./nachos SJF –e ../test/sleep –e ../test/test1 –e ../test/test1

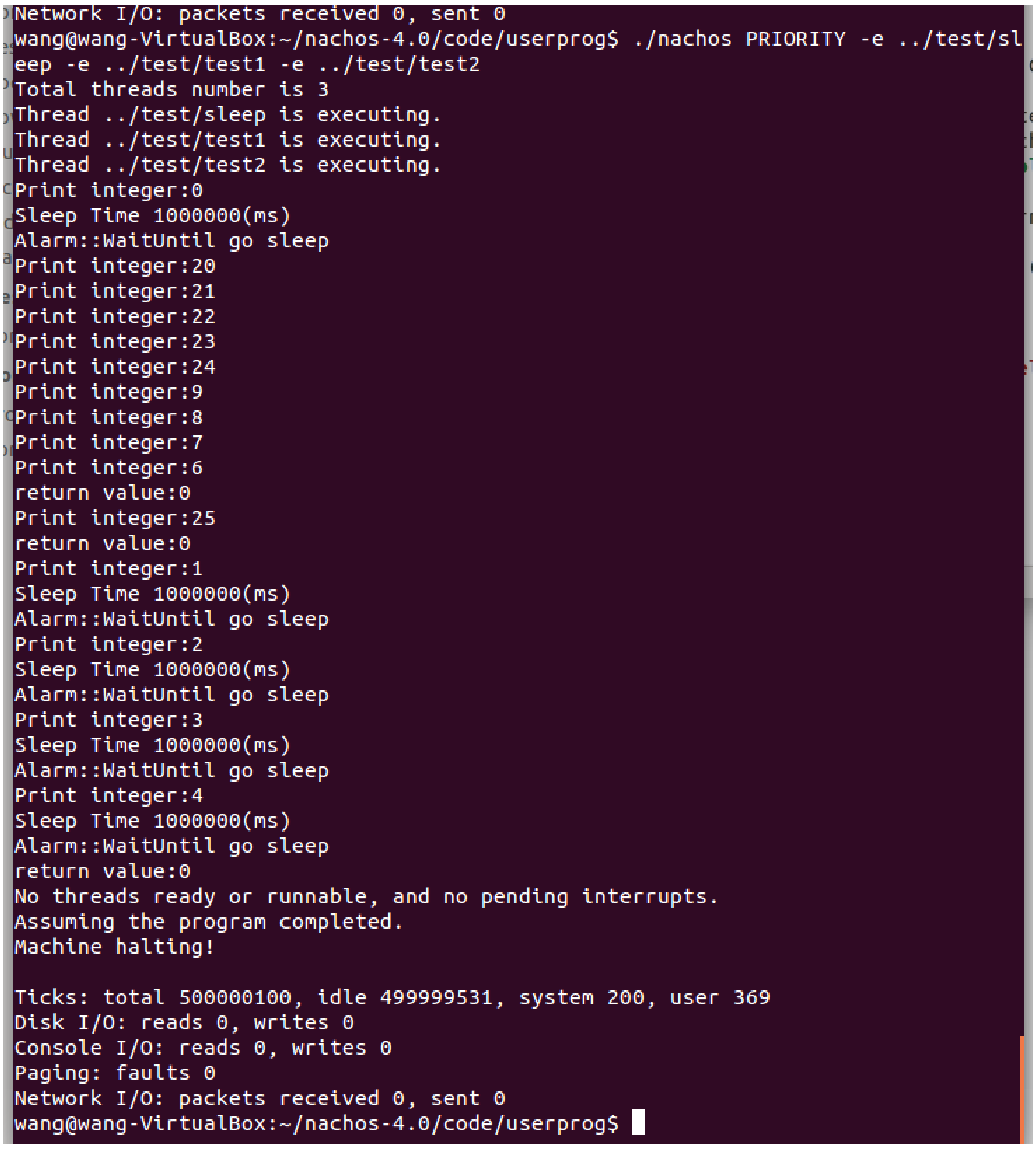

wang@wang-VirtualBox: ~/nachos-4.0/code/userprog

```
Ticks: total 500000100, idle 499999531, system 200, user 369
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos SJF -e ../test/sleep -
e ../test/test1 -e ../test/test2
Total threads number is 3
Thread ../test/sleep is executing.
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:0
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
Print integer:1
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:2
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:3
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:4
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500000100, idle 499999611, system 120, user 369
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$
```

預期 test1 會先於 test2、sleep 執行沒錯，但第一次先執行一次的 sleep 無法解釋。

PRIORITY

於 userprog 下執行 ./nachos PRIORITY –e ../test/sleep –e ../test/test1 –e ../test/test1

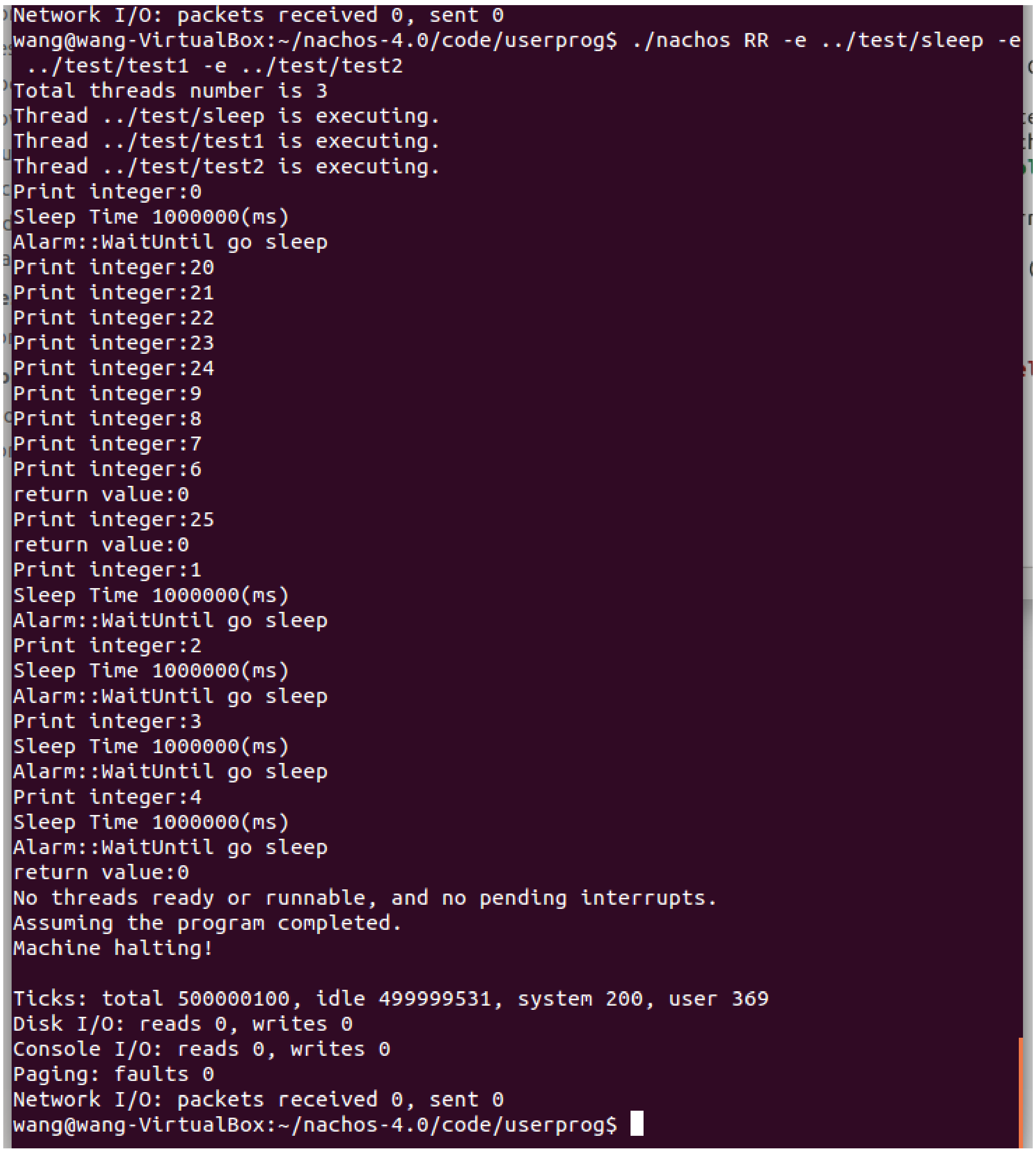


```
Network I/O: packets received 0, sent 0
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos PRIORITY -e ../test/sl
eep -e ../test/test1 -e ../test/test2
Total threads number is 3
Thread ../test/sleep is executing.
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:0
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
Print integer:25
return value:0
Print integer:1
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:2
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:3
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:4
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500000100, idle 499999531, system 200, user 369
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$
```

一開始沒有給定優先序故略過。

RR

於 userprog 下執行 ./nachos RR –e ../test/sleep –e ../test/test1 –e ../test/test1



```
Network I/O: packets received 0, sent 0
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos RR -e ../test/sleep -e
 ../test/test1 -e ../test/test2
Total threads number is 3
Thread ../test/sleep is executing.
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:0
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
Print integer:25
return value:0
Print integer:1
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:2
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:3
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
Print integer:4
Sleep Time 1000000(ms)
Alarm::WaitUntil go sleep
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500000100, idle 499999531, system 200, user 369
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
wang@wang-VirtualBox:~/nachos-4.0/code/userprog$
```

Test1, test2, sleep 應該交錯執行，但 sleep 出現一樣的先只執行一次的狀況，test1、test2 亦只交錯一次，猜測仍是 sleep 執行時間過久、test1,2 執行快的原因。