HW2_report

B10632026 吳苡瑄

B10632014 陳品羚

因為註解及說明較多較繁雜，我放在我的 blog 中 http://www.wiiwu.com/2019/10/29/OS_HW2-2019/

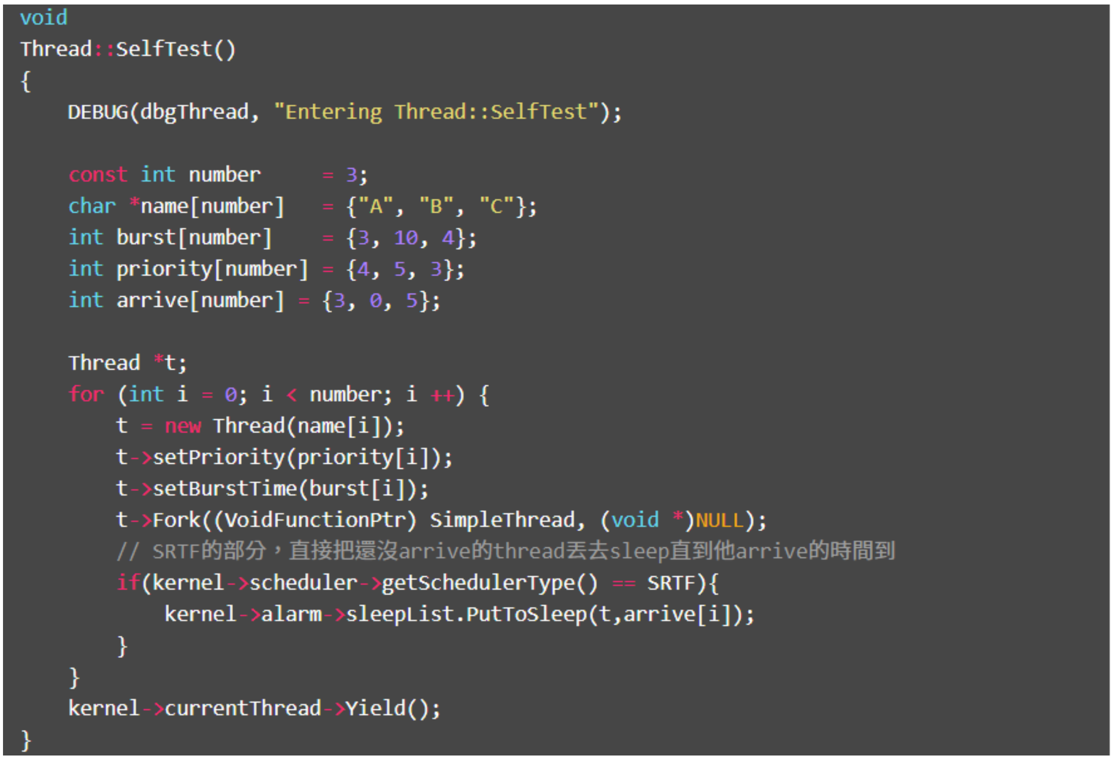
大致說明:

實現 Multi-programming：

在 addrspace.h 和 addrspace.cc 中增加為 physical address 做標記的內容和釋放資源、計算進入點的內容。

SelfTest:

在 thread.cc 中 SelfTest()的地方初始化參數、並讓他去 fork SimpleThread()。



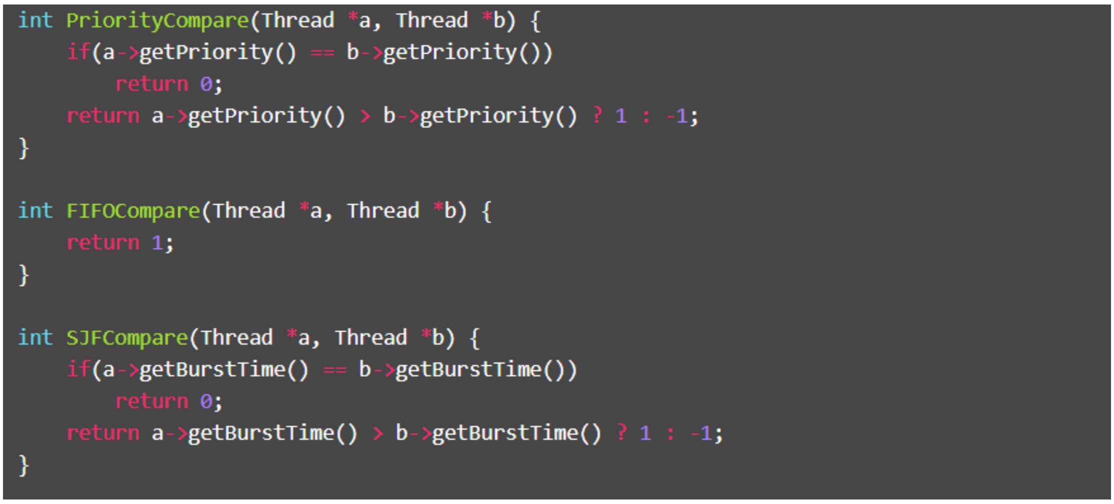
```
void
Thread::SelfTest()
{
    DEBUG(dbgThread, "Entering Thread::SelfTest");

    const int number      = 3;
    char *name[number]    = {"A", "B", "C"};
    int burst[number]     = {3, 10, 4};
    int priority[number] = {4, 5, 3};
    int arrive[number] = {3, 0, 5};

    Thread *t;
    for (int i = 0; i < number; i ++) {
        t = new Thread(name[i]);
        t->setPriority(priority[i]);
        t->setBurstTime(burst[i]);
        t->Fork((VoidFunctionPtr) SimpleThread, (void *)NULL);
        // SRTF的部分，直接把還沒arrive的thread丟去sleep直到他arrive的時間到
        if(kernel->scheduler->getSchedulerType() == SRTF){
            kernel->alarm->sleepList.PutToSleep(t,arrive[i]);
        }
    }
    kernel->currentThread->Yield();
}
```

FIFO、SJF：

kernel.cc 是負責處理參數，我們才能直接在指令裡面決定要用哪個方法排程， scheduler.h、scheduler.cc 就是主要實現排程法的地方，根據不同排程法用不同的 compare 法來 sortlist 就能完成



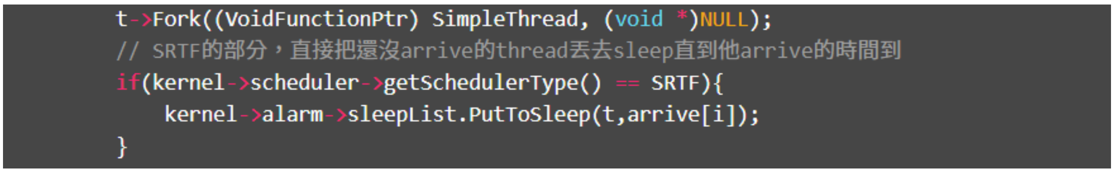
```
int PriorityCompare(Thread *a, Thread *b) {
    if(a->getPriority() == b->getPriority())
        return 0;
    return a->getPriority() > b->getPriority() ? 1 : -1;
}

int FIFOCompare(Thread *a, Thread *b) {
    return 1;
}

int SJFCompare(Thread *a, Thread *b) {
    if(a->getBurstTime() == b->getBurstTime())
        return 0;
    return a->getBurstTime() > b->getBurstTime() ? 1 : -1;
}
```

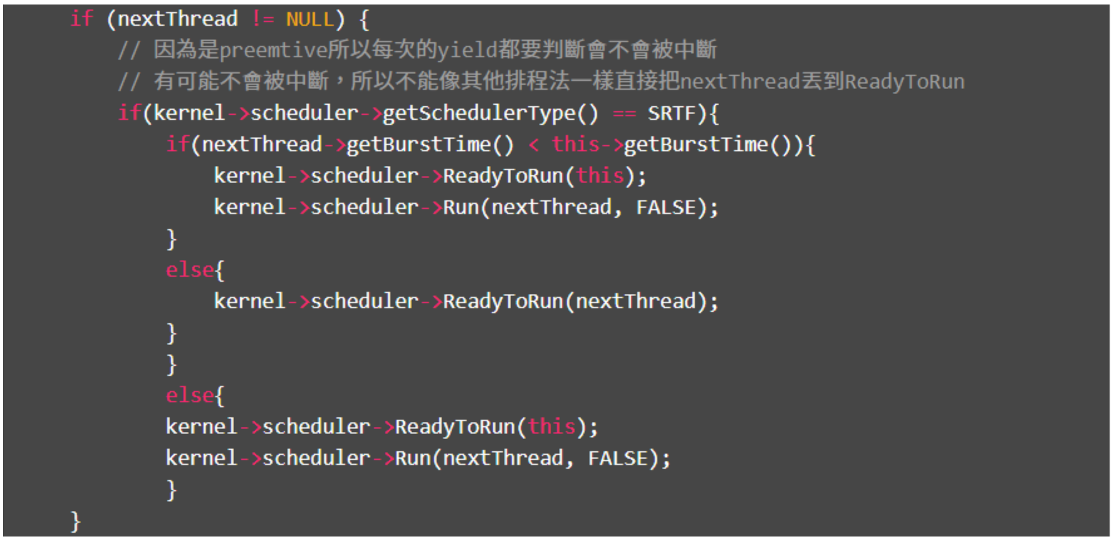
SRTF：

在 SelfTest 中剛 fork 完的時候，就依據不同的 arrive 時間讓他去 sleep。



```
        t->Fork((VoidFunctionPtr) SimpleThread, (void *)NULL);
        // SRTF的部分，直接把還沒arrive的thread丟去sleep直到他arrive的時間到
        if(kernel->scheduler->getSchedulerType() == SRTF){
            kernel->alarm->sleepList.PutToSleep(t,arrive[i]);
        }
```

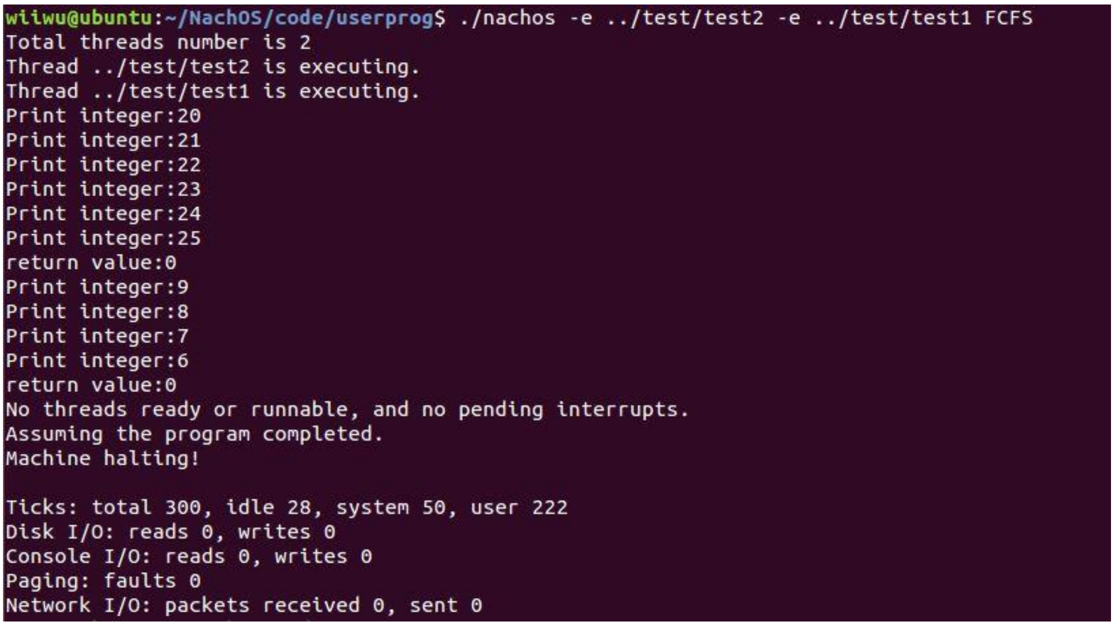
還要對 Yield 做調整，多加判斷是否需要被換掉，因為他每次 preemtive 都會去叫一次 Yield，所以不能用原本只要每次叫 Yield 就換下一個人的方法。



```
    if (nextThread != NULL) {
        // 因為是preemtive所以每次的yield都要判斷會不會被中斷
        // 有可能不會被中斷，所以不能像其他排程法一樣直接把nextThread丟到ReadyToRun
        if(kernel->scheduler->getSchedulerType() == SRTF){
            if(nextThread->getBurstTime() < this->getBurstTime()){
                kernel->scheduler->ReadyToRun(this);
                kernel->scheduler->Run(nextThread, FALSE);
            }
            else{
                kernel->scheduler->ReadyToRun(nextThread);
            }
            }
            else{
            kernel->scheduler->ReadyToRun(this);
            kernel->scheduler->Run(nextThread, FALSE);
            }
    }
```

結果

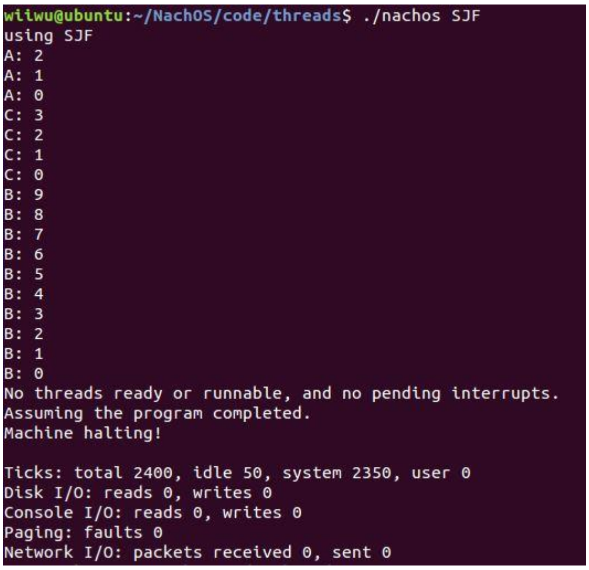
FCFS



```
wiiwu@ubuntu:~/NachOS/code/userprog$ ./nachos -e ../test/test2 -e ../test/test1 FCFS
Total threads number is 2
Thread ../test/test2 is executing.
Thread ../test/test1 is executing.
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 300, idle 28, system 50, user 222
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
```

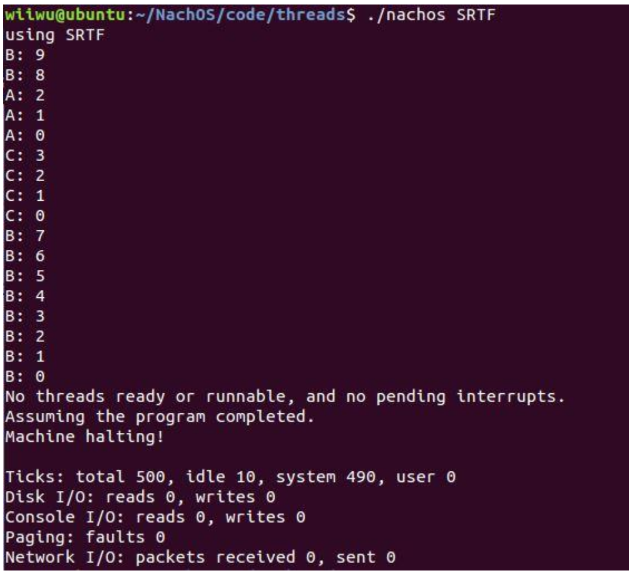
SJF



```
wiiwu@ubuntu:~/NachOS/code/threads$ ./nachos SJF
using SJF
A: 2
A: 1
A: 0
C: 3
C: 2
C: 1
C: 0
B: 9
B: 8
B: 7
B: 6
B: 5
B: 4
B: 3
B: 2
B: 1
B: 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 2400, idle 50, system 2350, 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
```

SRTF



```
wiiwu@ubuntu:~/NachOS/code/threads$ ./nachos SRTF
using SRTF
B: 9
B: 8
A: 2
A: 1
A: 0
C: 3
C: 2
C: 1
C: 0
B: 7
B: 6
B: 5
B: 4
B: 3
B: 2
B: 1
B: 0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 500, idle 10, system 490, 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
```