Content

[I. Motivation 2](#_Toc52895291)

[II. Implementation 4](#_Toc52895292)

[III. Result 8](#_Toc52895293)

Reference

1. coffeered Nachos (解決dependency 問題)

https://github.com/coffeered/Nachos

1. Nachos Project1 Report

<https://github.com/taldehyde/os-project-1/blob/master/report.md>

2. OS::NachOS::HW1

<http://blog.terrynini.tw/tw/OS-NachOS-HW1/>

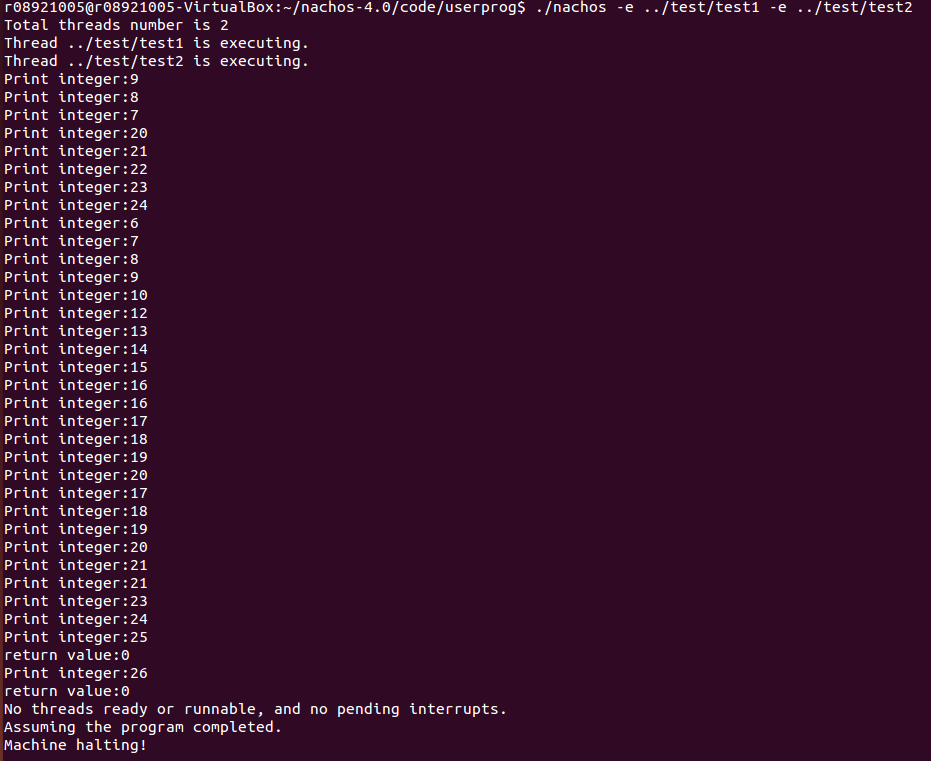
3. 向 NachOS 4.0 作業進發 (1)

<http://morris821028.github.io/2014/05/24/lesson/hw-nachos4/>

# Motivation

* + Motivation and the problem analysis
  + What’s your plan to deal with the problem?

The result is not congruent with expected.



首先我先檢視了test1.c與test2.c兩個檔案，分別為下：

|  |  |
| --- | --- |
| test1.c  應是遞減輸出從9~6 | test2.c  應是遞增輸出20~25 |
| 輸出的結果是錯的 | |

原本的 NachOS並沒有為多個程式做記憶體的管理，雖然有開啟虛擬記憶體，但是基本上沒有作為，所以當多份程式同時執行時就會重疊到其他程式正在使用的 pageTable，然後發生錯誤。

先備知識:

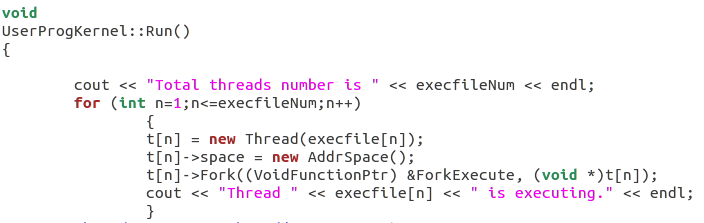
將「 Virtual Address 」對應到「 Physical Address 」。

Virtual Address -> process 執行的過程中存取資料的地方 / process在看的記憶體

Physical Address -> 電腦實際儲放資料的地方 / 真正的記憶體位置

Trace the following files and find out why the result is wrong

userkernel.cc



理論上兩個thread應該要使用不同的pageTable，但原先NachOS的程式碼並沒有做此設定，因而導致兩個thread用到的是同一個pageTable，才會發生此錯誤。

為此要修改nachos-4.0/code/userprog/addrspace.cc, nachos-4.0/code/userprog/addr-space.h，使程式的虛擬記憶體映射到沒有人使用的實體記憶體，而不是互相糾纏。

Problem Formulation

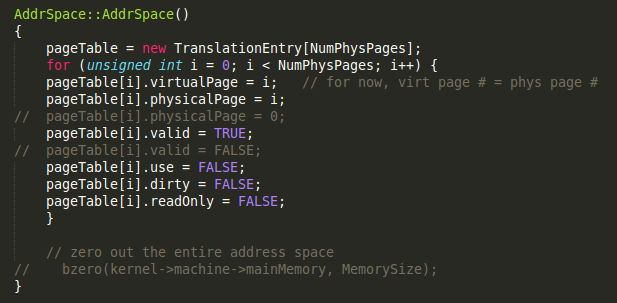
1. 記憶體浪費 :

當建立一個thread時，pageTable直接宣告了最大的Physical Page (NumPhysPages)

1. 不當thread管理 :

當建立多個thread時，每個thread的程式碼可能對應到了相同的physical page

addrspace.cc



# Implementation

* + How do you implement to solve the problem in Nachos?
  + You can include some (not all) important code segments and comments

輸出 n 的結果錯誤，猜測為context-switch上發生問題。

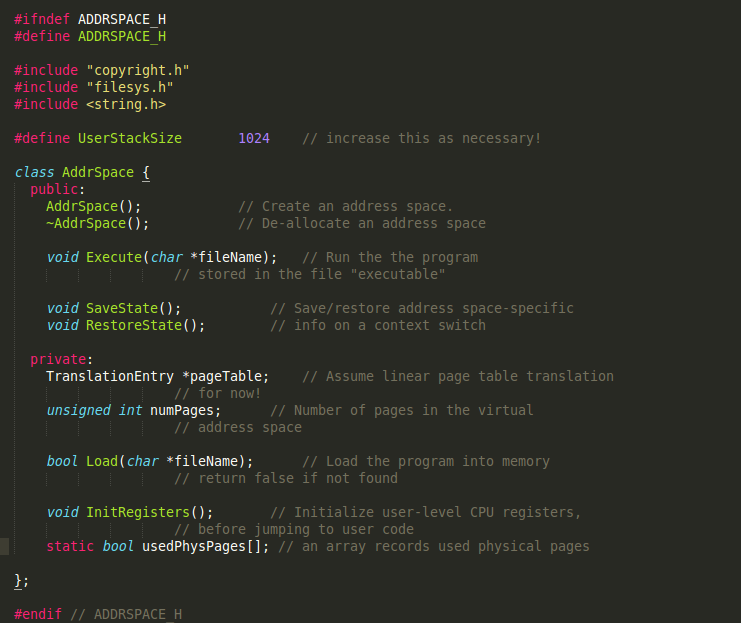
因此，需要為physical pages做標記。因此預計在addrspace.h新增static 變數

∵static是屬於class共享的

1. 用來記錄所有physical pages的使用狀況
2. 知道當前還有多少physical pages是可以用的

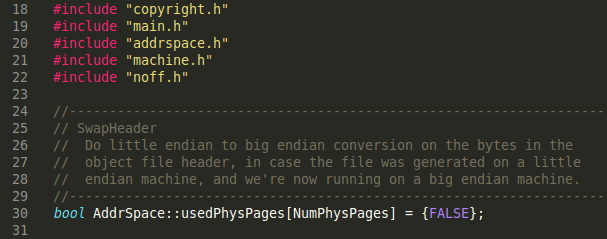
首先先在addrspace.h新增變數static bool usedPhysPages紀錄哪些是被使用過的

addrspace.h



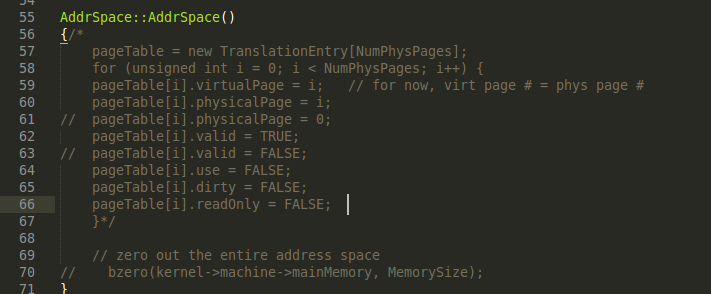
而static變數一定要在addrspace.cc中做過一次初始化

addrspace.cc



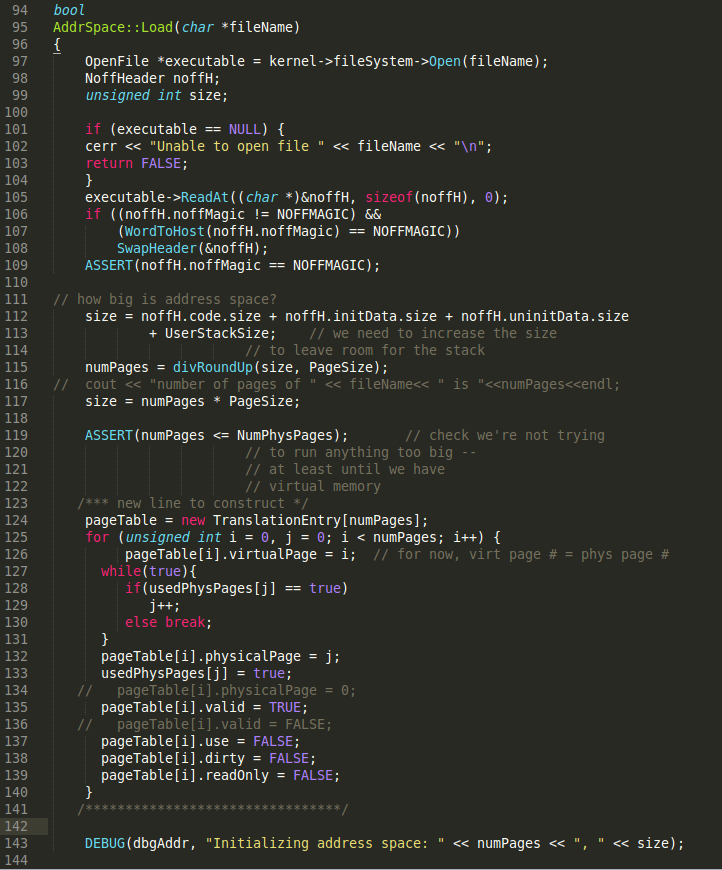
在addrspace.cc中，constructor直宣告一個跟實體記憶體一樣大的 pageTable，但是其實程式沒這麼大，只要分配到跟程式一樣大就好了，然而程式大小必須讀了檔案格式才會知道，所以可以把這整段 code 註解掉，把映射記憶體這件事移到 AddrSpace::Load 中。

addrspace.cc



到addrspace.cc的AddrSpace::Load去重新分配physicalPage，同時維護剛剛新增的變數，每次要assign physicalPage前，先去新增的變數檢查看該位置是否已經被別的physical address所佔據，如果是，則程式會去找下一個未被使用的physicalPage，並把它assign給特定的virtual page。

addrspace.cc

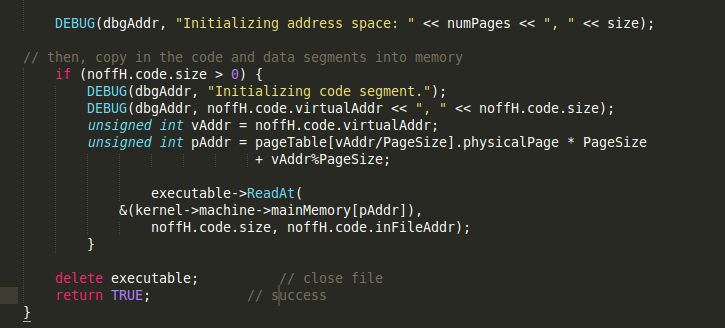


之後將指令複製到memory上，physicalpage的address要用pageTable來換算。使用linear time去搜尋第一個未使用的page進行填入。載入確定後，便要開始執行，

執行時，要去算程序進入點，進入點的位置即是main memory address。

1. 算出第幾頁，乘上 PageSize 就是 page base
2. page offset = code.address%PageSize
3. page base + page offset 就是所需要的程序進入點。

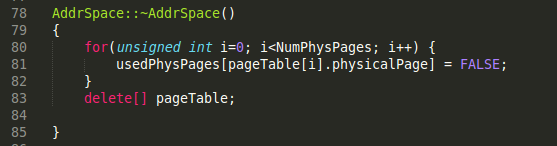
addrspace.cc



最後在process執行完後，要把記憶體歸回給CPU，釋放資源。

把標記為使用中的page設回未使用

addrspace.cc



# Result

* + Experiment result and some discussion
  + Extra effort or observation

做完以上動作之後，執行./nachos -e ../test/test1 -e ../test/test2 的結果如下：

|  |
| --- |
| 最終執行結果 |

程式正確的執行了，並且是用Round-Robin(Nachos預設的scheduling)的方式輪流執行。

但這樣的程式會產生一個問題。當執行多個thread時，會因為physical page的數量不夠而導致程式出現segmentation fault。如下面結果，因此在這邊應該隨著執行程式的thread數量增加physical page的數目，以維持足夠的數量供給thread使用

執行./nachos -e ../test/test1 -e ../test/test2 -e ../test/test1的結果如下

