|  |
| --- |
| Operating System |
| Hw02 MP2: Multi-Programming |
| Report |

|  |
| --- |
| Team 20 member : 105060011蔡悅承 105060016 謝承儒  Team member contribution : Trace Code : 蔡悅承 謝承儒 Implement : 蔡悅承 謝承儒 Report : 蔡悅承 謝承儒 |

Trace Code

1. 流程圖

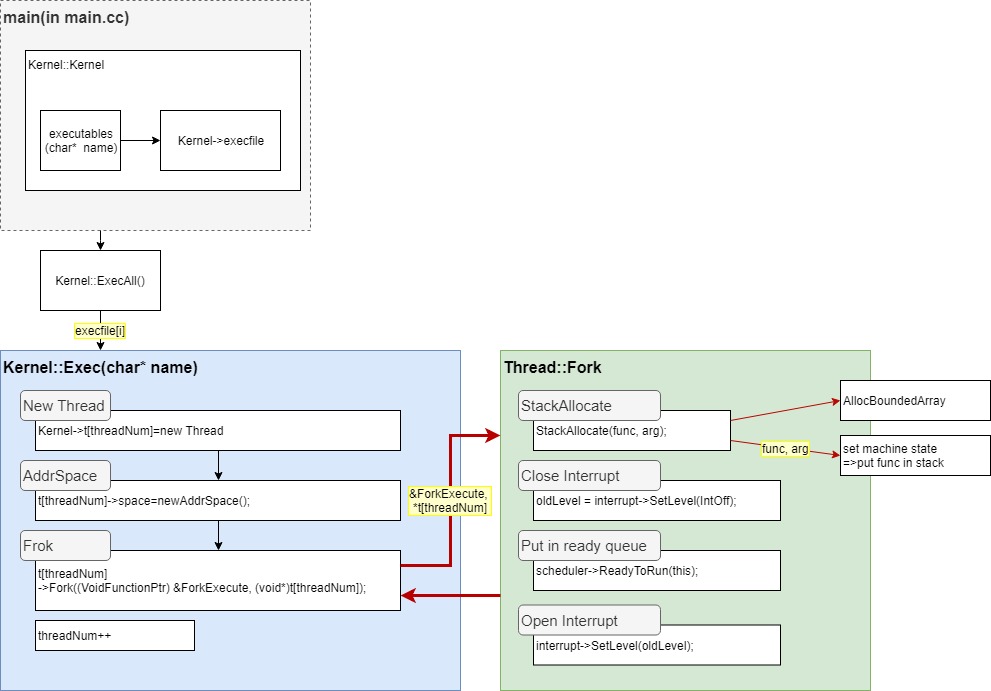


圖1 Kernel::Exec流程圖

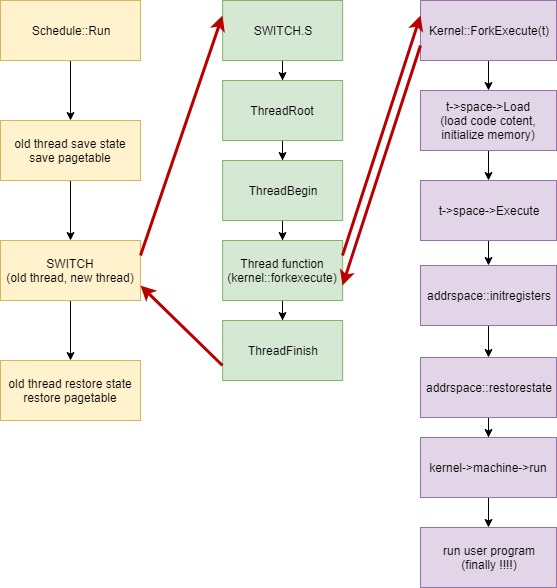


圖2 Machine::Run流程圖

1. 步驟
   1. Kernel::Exec
      1. 在Kernel::Kernel時，將要執行的檔名放進execfile[]裡。
      2. 接著呼叫Kernel::ExecAll()，會把execfile裡的element一個個丟進Kernel::Exec()
      3. 為進來的execfile[i]，依序做:
         1. new Thread(name, threadNum)

創建新thread

* + - 1. AddrSpace:: AddrSpace

給予該新thread address space

* + - 1. Thread::Fork( (VoidFunctionPtr) &ForkExecute, (void \*)t[threadNum] )(接續步驟d)
      2. threadNum++

計算thread的數量，為下一個thread做準備

* + 1. 在Thread::Fork裡依序做
       1. StackAllocate

給予stack相對應的空間，並把func放入stack

* + - 1. 把interrupt關閉，避免待會Schedular::ReadyToRun被打斷
      2. 將thread放入ready queue
      3. 打開interruput
  1. Machine::Run
     1. Shcedule::Run開始執行ready queue裡的Theread
     2. 發生Context switch，準備改成執行new thread
     3. 在Switch.s裡，會依序呼叫 :
        1. ThreadRoot
        2. ThreadBegin
        3. Thread function，以上面為例，這個就是Kernel::ForkExcute(t)(接續步驟d)
        4. ThreadFinish
     4. Kernel::ForkExcute中會執行:
        1. 該thread的addrspace->Load，將該thread的東西load進memory
        2. 呼叫addrspace->Execute，將Registers清空、page table也load進來
     5. Machine::Run開始，program開始執行
     6. 當program執行結束後，回到步驟c-(d)，執行ThreadFinish，將thread刪除
     7. Context switch回old thread，繼續執行old thread

Questions

* 1. How Nachos allocates the memory space for new thread(process)?

Addrspace::Addrspace給了，把它的vail設成true(代表已經y在memory)

* 1. How Nachos initializes the memory content of a thread(process), including loading the user

binary code in the memory?

Addrspcae::Load裡的ReadAt，利用virtual address轉成physical address後，將它load進kernel->machine->mainMemory

* 1. How Nachos creates and manages the page table?

Ceate :　Addrspace::Addrspace

Manage : 每次load new thread就assign全部的memory給它，也就是說，預設只有single thread執行。

* 1. How Nachos translates address?
     1. PageNumber = virtual address / PageSize
     2. Check page number is valid
     3. FrameNumber = 用Page number在page table查表後的結果
     4. Check frame number is valid
     5. PageOffset = virtual address % size
     6. Physical address = Frame number\*PageSize + PageOffset
  2. How Nachos initializes the machine status (registers, etc) before running a thread(process)

Kernel::ForkExec

=>Addrspace::Execute=>InitialRegsiter=>RestoreState

=>Machine::Run

* 1. Which object in Nachos acts the role of process control block

Thread class

* 1. When and how does a thread get added into the ReadyToRun queue of Nachos CPU scheduler?

Thread::Fork=>Scheduler::ReadyToRun

Implement page table in NachOS

1. 修改想法

本來每次thread要allocate addrspace時，會建立自己的page table，同時也會將整個memory初始化，如此memory永遠都只會儲存單個thread的東西。

我們建立Frame table來代表memory，每次thread要allocate addrspace時 :

* 1. 建立自己的page table
  2. 在frame table(=Memory)找出空位，和page相對應
  3. 當thread要被移除時，把和pages相對應的frame格子，從frame table釋放。

藉由(2)、(3)來做到同時儲存多個thread的資料。

1. 實作內容

修改了Kernel.h、Kernel.cc、Addrspace.cc三個code

* 1. Kernel.h

1. 建立Frame table

bool FrameTable[NumPhysPages];

如果FrameTable[i]是true，代表該frame有儲存page；反之，如果是false，代表該frame是空的，可以被使用。

* 1. Kernel.cc

1. Initialize frame table

for (int i=0; i<NumPhysPages; i++)

FrameTable[i] = false; //all frames free

在Kernel開啟時，將FrameTable裡所有的entry都設為false，代表memory是全空的。

* 1. Addrspace.cc

1. Addrspace():
   1. 建立page table，並初始化page table
2. Load
   1. Page和Frame配對

尋找FrameTable上的空位(=FrameTable[k])，找到後將該page的physicalPage便等於k，同時FrameTable[k]也設為true，表示該位置被使用。

此外把第j個frame所在的memory清除成0。

* 1. 讀取code、data、readonlyData

本來是儲存的位置是連續的，但現在是不連續的，所以需要計算自料開始位置在哪，才能往下讀取到，計算範例:

code\_start\_page = pageTable[noffH.code.virtualAddr / PageSize].physicalPage;

code\_start\_offset = noffH.code.virtualAddr % PageSize;

code\_start\_physical = code\_start\_page \* PageSize + code\_start\_offset;

start page :

先找出自己的virtual address在哪個page，接著利用pageTable知道在FrameTable的哪一個。

start offset :

算出在自己該page的哪個位置。

最後

* 1. ~AddrSpace()

for (int i=0; i<NumPhysPages; i++) {

kernel->FrameTable[pageTable[i].physicalPage] = false;

}

把有和該thread的page相連的frame都設成false，代表這些frame是可以再被其他thread使用的。