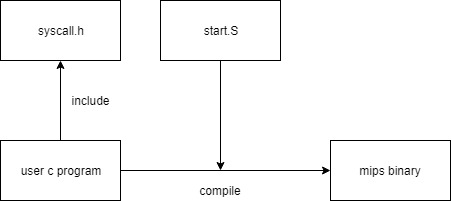
|  |
| --- |
| Operating System |
| Hw01 MP1: System Call |
| Report |

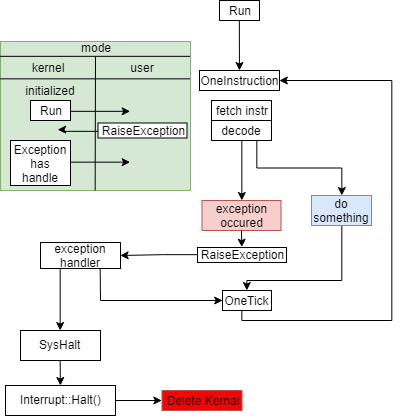
|  |
| --- |
| Team 20 member : 105060011蔡悅承 105060016 謝承儒  Team member contribution : Trace Code : 蔡悅承 謝承儒 Implement : 蔡悅承 Report : 謝承儒 |

Trace Code

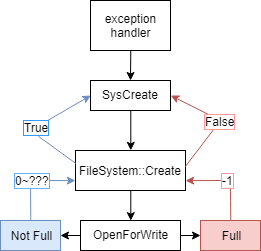
1. C++ Convert to MIPS
2. 流程圖



1. 步驟
2. 若C++有用到syscall，便會將該syscall的值存入Register的第2腳位(以下稱為r2)，這項動作實做於start.s。各項syscall的值定義於syscall.h。
3. Syscall裡放入的參數，會被依序放在r4、r5、r6…。(code中並未解釋如何達成，只有在start.s的comment提到)
4. SC\_Halt
5. 流程圖

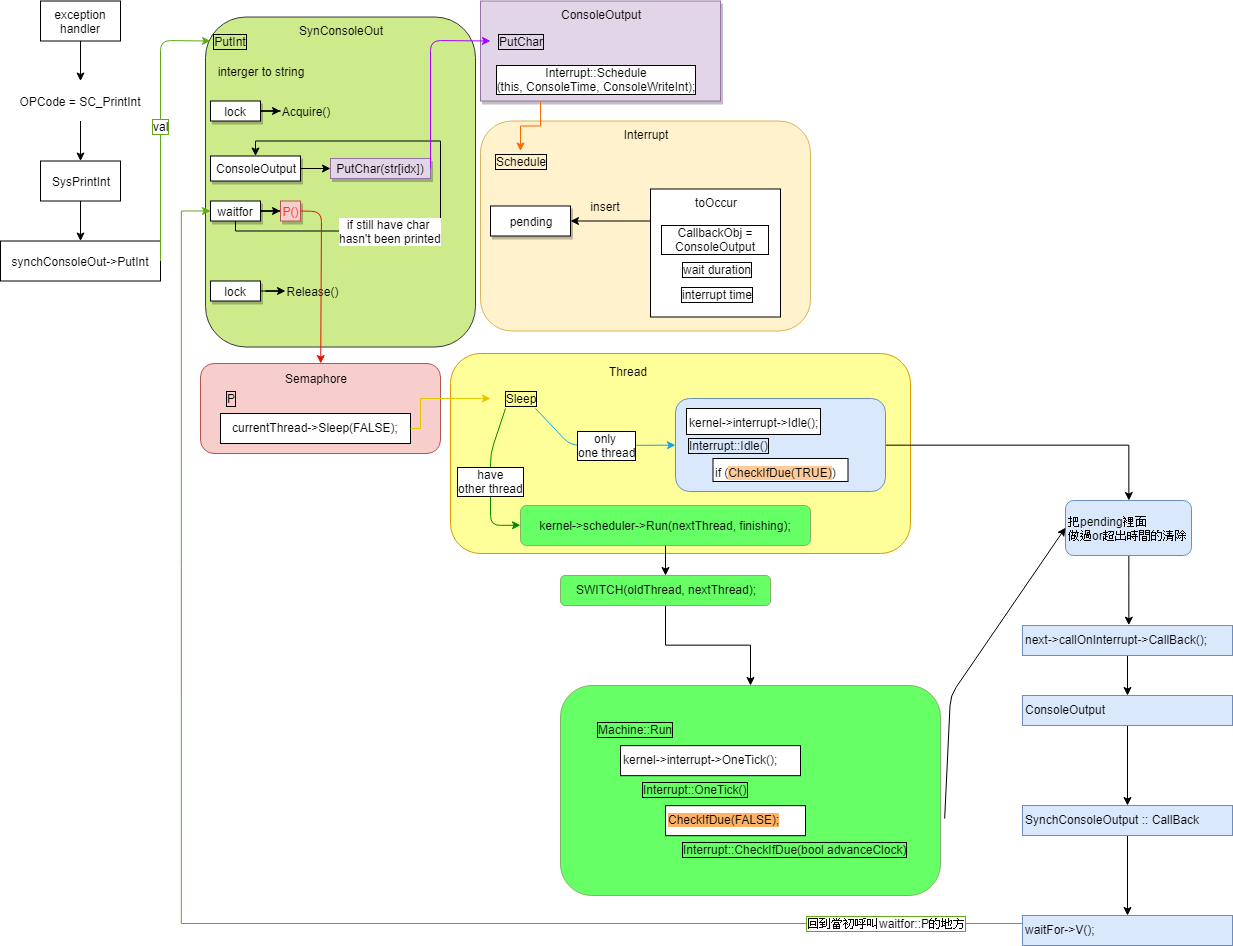


1. 步驟
2. 編譯完C++後，開始執行Run()，此時將mode切換成user mode。
3. 把mips裡的instruction decode後得到instr放入OneInstruction()，根據instr的opCode執行相對應的動作。若opCode為OP\_SysCall或是overflow則進入紅色方塊，呼叫RaiseException()；反之，則進入藍色方塊。
4. 進入RaiseException後，會將mode轉為kernel mode以便執行syscall，接著呼叫ExceptionHandler()。
5. ExceptionHandler會從r2知道為何種syscall，以此做出相對應的動作，若syscall為syshalt，則呼叫Halt()，把整個kernel刪除，相當於關機。倘若不是，則執行syscall後接續下面步驟e。
6. 回到RaiseException，將mode轉為user mode。
7. 回到OneInstruction()，執行OneTick()，然後再執行下一條instruction。
8. SC\_Create
   1. 流程圖



* 1. 步驟

1. ExceptionHandler()接收到SC\_Create，呼叫SysCreate()。
2. SysCreate裡會呼叫OpenForWrite，若空間已滿便會回傳-1；若還有空間可以創建，則找出位置並回傳該文件的id。
3. SC\_PrintInt
   1. 流程圖



* 1. 步驟

1. ExceptionHandler()接收到SC\_PrintInt，呼叫SysPrintInt(val)，要Print的數值為val，該值可以從r4得到。
2. SysPrintInt裡呼叫synchConsoleOut的PutInt(val)。
3. 把val轉成string(val=>str)，並把output的lock鎖住，避免其他thread搶走output的權限。
4. ConsoleOutput的PutChar(str[idx])可以把字印出來，利用while迴圈把str裡的字一個個印出來。
5. Console::PutChar會呼叫Interrupt::Schedule，把”印出字”這件事(下面稱做toOccur)放入pending。
6. toOccur放入pending後，便呼叫waitfor裡的P()讓該thread(下面稱作thread A)休息，等待output完成。
7. 如果thread A是唯一的thread，呼叫kernel的Idel()，裡面的CheckIfDue(TRUE)會檢查pending，接續步驟j。
8. 如果thread A不是唯一的thread，呼叫Switch把CPU讓給其他的thread(下面稱做thread B)。
9. thread B的執行過程時，會呼叫OneTick()，裡面會使用CheckIfDue(FALSE)，接續步驟j。
10. CheckIfDue()會把pending中逾期或是當下時間的element清除，同時執行該element的CallBack()，toOccur的CallBack便是waitfor裡的V()，將thread A喚醒。
11. 如果str裡仍然有字還沒印出，回到步驟e。如果已經全部印出，則結束迴圈並把output的lock解除。

＊步驟g、i雖然都是呼叫CheckIfDue(bool advanceClock)，但步驟g中參數是TRUE，代表可以預先處理，如果檢查pending中的第一個後發現時間未到，則直接加速至該時間；反之，步驟i中參數是FALSE，因為必須處理thread B，不能夠直接加速，如果檢查pending中的第一個後發現時間未到，則直接結束CheckIfDue()。

Implement four file I/O system call in NachOS

1. 前置作業
   1. 在syscall.h中，define SC\_Open、SC\_Write、SC\_Read、SC\_Close的值。

|  |  |  |  |
| --- | --- | --- | --- |
| SC\_Open | SC\_Write | SC\_Read | SC\_Close |
| 6 | 8 | 7 | 10 |

* 1. 在Start.s中，加入當遇到Open、Write、Read、Close，把r2位設為相對應的值(SC\_Open、SC\_Write、SC\_Read、SC\_Close)並呼叫syscall。
  2. 在ExceptionHandler中，增加遇到SC\_Open、SC\_Write、SC\_Read、SC\_Close時的case，呼叫SysOpen、SysWrite、SysRead、SysClose。
  3. 在ksyscall.h中，加上SysOpen、SysWrite、SysRead、SysClose function。
     1. OpenFileId SysOpen(char \*filename)
     2. int SysWrite(char \*buffer, int size, int id)
     3. int SysRead(char \*buffer, int size, int id)
     4. int SysClose(int id)

1. OpenFileId Open(char \*name)

選擇實作OpenFileId回傳型態。

因此，我們多寫了一個OpenFileId HandleOpen(char \*filename)來處理開檔時會遇到的問題。

(-1代表遇到開檔錯誤)

* 1. 資料型態

為了符合資料型態，ksyscall.h裡的SysOpen是呼叫HandleOpen(char \*filename)，而不是Open(char \*name)，前者的型別是OpenFileId，後者是OpenFile\*。

* 1. 問題一 : 沒有檔案的名字是filename

HandleOpen裡會宣告OpenFile\* openfile=Open(filename)，在Open函式中會使用OpenForReadWrite(filename)來尋找該檔名。

若是沒找到同檔名的檔案會回傳null；反之，若是找到該檔名，便會回傳OpenFile(fileDescriptor, name)，裡頭包含檔名和編號。

如果openfile是null，代表沒有找到檔案，回傳-1。

* 1. 問題二 : 是否有重複開檔
     1. 在OpenFile Class新增char\* name。

這樣就可以知道fileDescriptorTable裡的OpenFile element是屬於哪個檔案。

* + 1. Include <string.h>

為了使用strcmp來比較兩字串是否相同。

有了上述a、b兩件事後，就可以利用一個for迴圈，來檢查每個在Table裡OpenFile element的名字是否和filename相同。

如果有相同的名稱就表示重複開檔，回傳-1。

* 1. 問題三 : fileDescriptorTable是否已經填滿

利用一個for迴圈來檢查Table有沒有element是null，若第Table[i]是null表示該格為空，就可以把openfile放入，並回傳i當作用來查fileDescriptorTable的id。

若是已經滿，則會跳出迴圈，便回傳-1。

1. int Write(char \*buffer, int size, OpenFileId id)
   1. 參數意義
      1. buffer : 要寫入的內容
      2. size : 要寫buffer前多少個字進去
      3. id : 用來查fileDescriptorTable，看要寫入哪個開啟的檔案
      4. 回傳值 : 成功寫了多少個字
   2. 運行過程
      1. 利用id在fileDescriptorTable找到想要的Openfile。
      2. 若找不到表示該檔案沒有開啟，回傳-1。
      3. 若找到檔案，則呼叫該Openfile的Write(buffer, size)來寫入。
2. int Read(char \*buffer, int size, OpenFileId id)
   1. 參數意義
      1. buffer : 讀到的內容
      2. size : 要讀file前多少個字
      3. id : 用來查fileDescriptorTable，看要讀取哪個開啟的檔案
      4. 回傳值 : 成功讀了多少個字
   2. 運行過程
      1. 利用id在fileDescriptorTable找到想要的Openfile。
      2. 若找不到表示該檔案沒有開啟，回傳-1。
      3. 若找到檔案，則呼叫該Openfile的Read(buffer, size)來讀取。
3. int Close(OpenFileId id)
   1. 參數意義
      1. id : 用來查fileDescriptorTable，看要關閉哪個開啟的檔案
      2. 回傳值 : 是否成功刪除，0表示無檔案，1表示成功刪除
   2. 運行過程
      1. 利用id在fileDescriptorTable找到想要的Openfile。
      2. 若找不到表示該檔案沒有開啟，回傳0。
      3. 若找到檔案，則呼叫delete清空fileDescriptorTable[id]，並把該位置重新設回null。