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

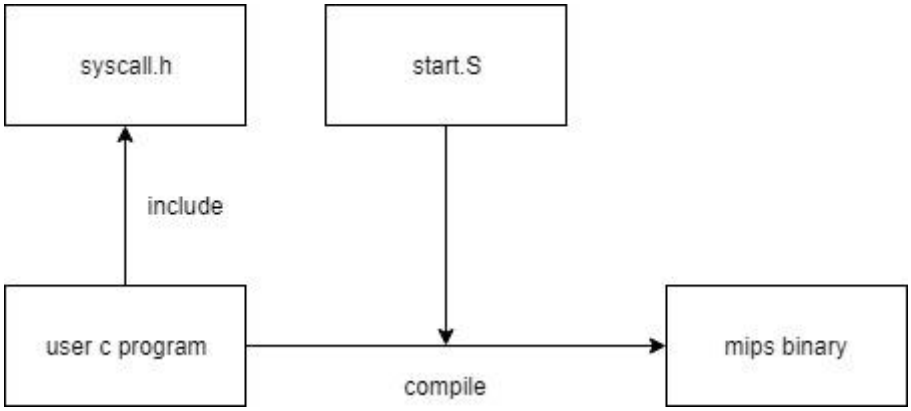
Implement : 蔡悅承

Report : 謝承儒

Trace Code

0. C++ Convert to MIPS

(1) 流程圖



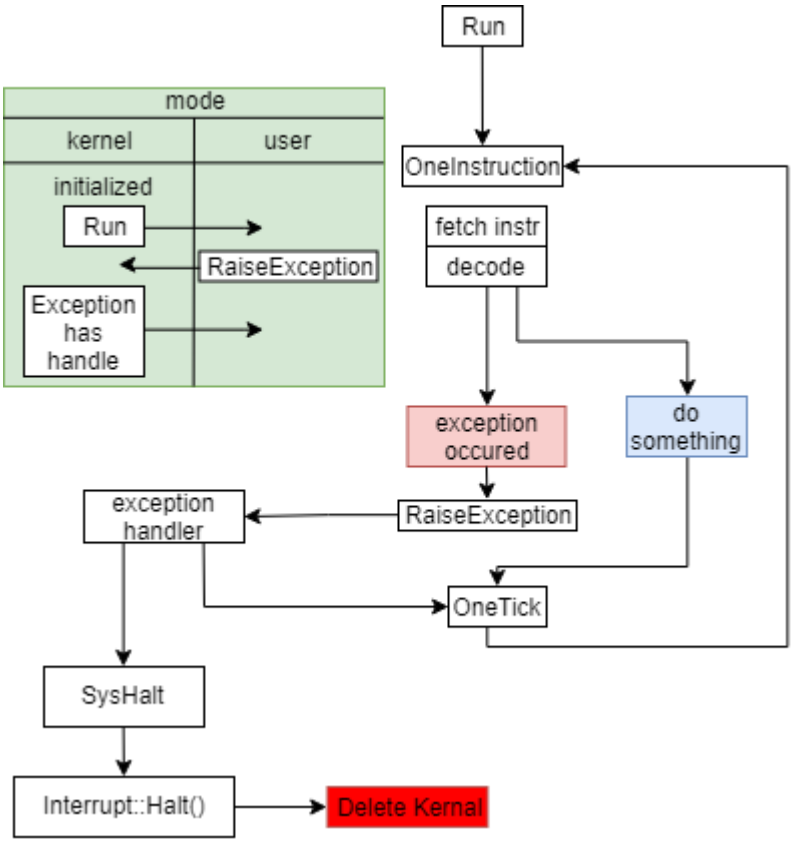
(2) 步驟

a. 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。

b. 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)

1. SC_Halt

(1) 流程圖



(2) 步驟

a. 編譯完 C++後，開始執行 Run()，此時將 mode 切換成 user mode。

b. 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。

c. 進入 RaiseException 後，會將 mode 轉為 kernel mode 以便執行 syscall，接著呼叫 ExceptionHandler()。

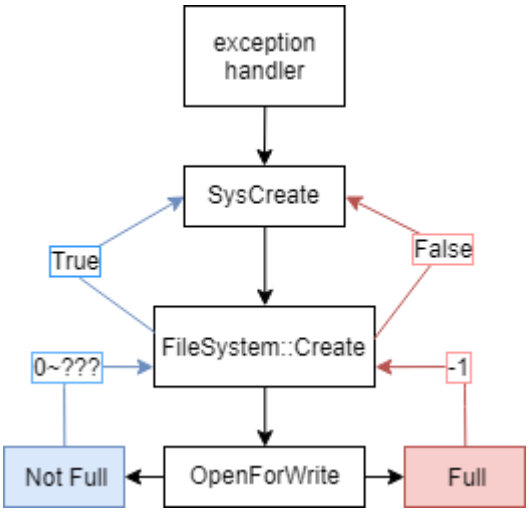
d. ExceptionHandler 會從 r2 知道為何種 syscall，以此做出相對應的動作，若 syscall 為 syshalt，則呼叫 Halt()，把整個 kernel 刪除，相當於關機。倘若不是，則執行 syscall 後接續下面步驟 e。

e. 回到 RaiseException，將 mode 轉為 user mode。

f. 回到 OneInstruction()，執行 OneTick()，然後再執行下一條 instruction。

2. SC_Create

(1) 流程圖



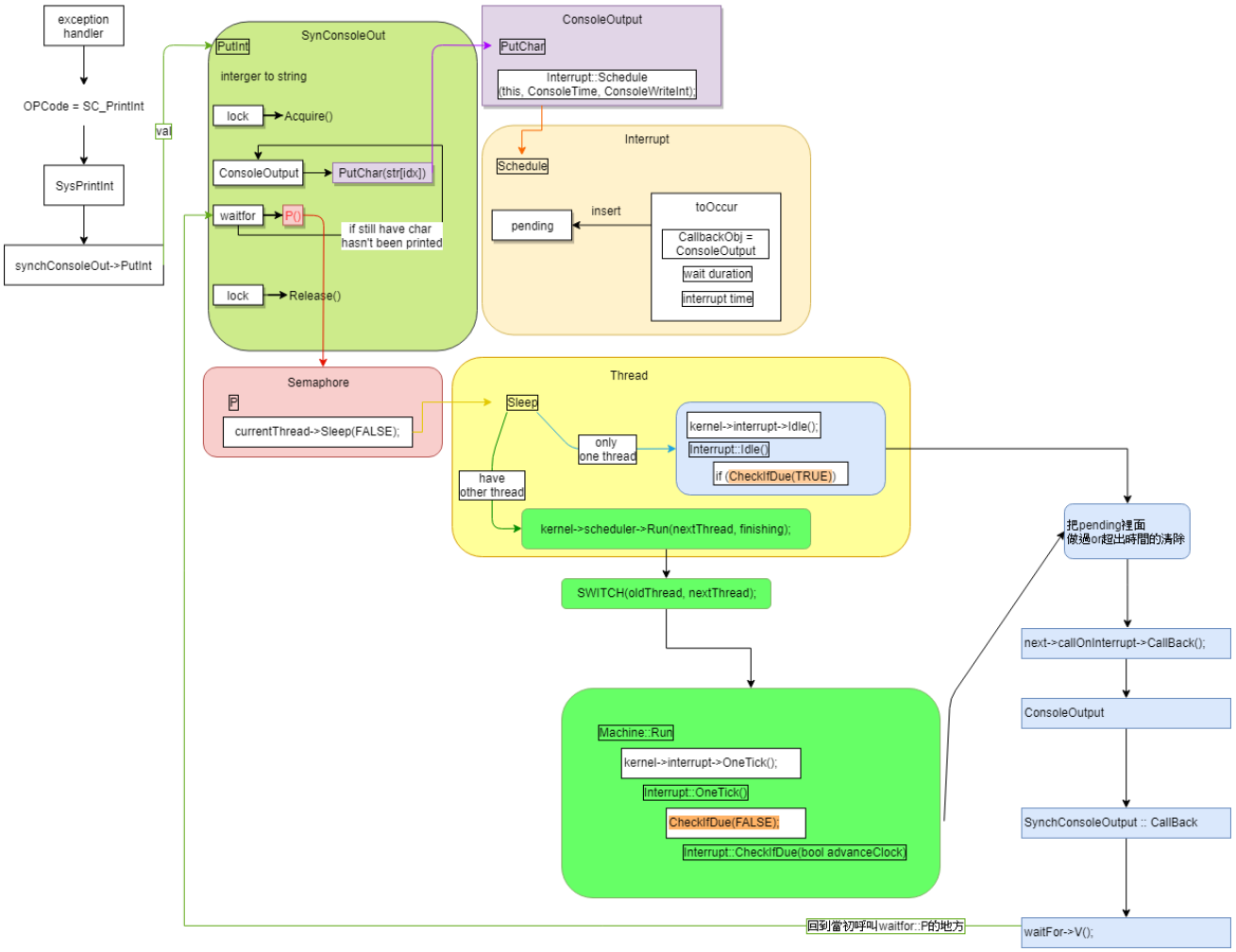
(2) 步驟

a. ExceptionHandler()接收到 SC_Create，呼叫 SysCreate()。

b. 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



(2) 步驟

a. 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。
b. SysPrintInt 裡呼叫 synchConsoleOut 的 PutInt(val)。
c. 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。
d. 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。
e. Console::PutChar 會呼叫 Interrupt::Schedule，把”印出字”這件事(下面稱做 toOccur)放入 pending。
f. 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。
g. 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。

h. 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)。

i. thread B 的執行過程時，會呼叫 OneTick()，裡面會使用 CheckIfDue(FALSE)，接續步驟 j。

j. 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。

k. 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

0. 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

(2) 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。

(3) 在ExceptionHandler中，增加遇到SC_Open、SC_Write、SC_Read、SC_Close時的case，呼叫SysOpen、SysWrite、SysRead、SysClose。

(4) 在ksyscall.h中，加上SysOpen、SysWrite、SysRead、SysClose function。

a. OpenFileId SysOpen(char *filename)
b. int SysWrite(char *buffer, int size, int id)
c. int SysRead(char *buffer, int size, int id)
d. 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*。

(2) 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。

(3) 問題二 : 是否有重複開檔

a. 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。

b. 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。

(4) 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。

2. int Write(char *buffer, int size, OpenFileId id)

(1) 參數意義

a. buffer : 要寫入的內容
b. size : 要寫buffer前多少個字進去
c. id : 用來查fileDescriptorTable，看要寫入哪個開啟的檔案
d. 回傳值 : 成功寫了多少個字

(2) 運行過程

a. 利用id在fileDescriptorTable找到想要的Openfile。
b. 若找不到表示該檔案沒有開啟，回傳-1。
c. 若找到檔案，則呼叫該Openfile的Write(buffer, size)來寫入。

3. int Read(char *buffer, int size, OpenFileId id)

(1) 參數意義

a. buffer : 讀到的內容
b. size : 要讀file前多少個字
c. id : 用來查fileDescriptorTable，看要讀取哪個開啟的檔案
d. 回傳值 : 成功讀了多少個字

(2) 運行過程

a. 利用id在fileDescriptorTable找到想要的Openfile。
b. 若找不到表示該檔案沒有開啟，回傳-1。
c. 若找到檔案，則呼叫該Openfile的Read(buffer, size)來讀取。

4. int Close(OpenFileId id)

(1) 參數意義

a. id : 用來查fileDescriptorTable，看要關閉哪個開啟的檔案
b. 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

a. 利用id在fileDescriptorTable找到想要的Openfile。
b. 若找不到表示該檔案沒有開啟，回傳0。
c. 若找到檔案，則呼叫delete清空fileDescriptorTable[id]，並把該位置重新設回null。