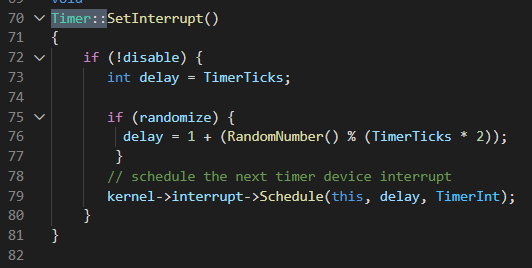


L2, L3是FCFS還是SJF呢? ( 同一個queue裡面，p的大小還是進去queue的順序影響到process的執行順序？ )



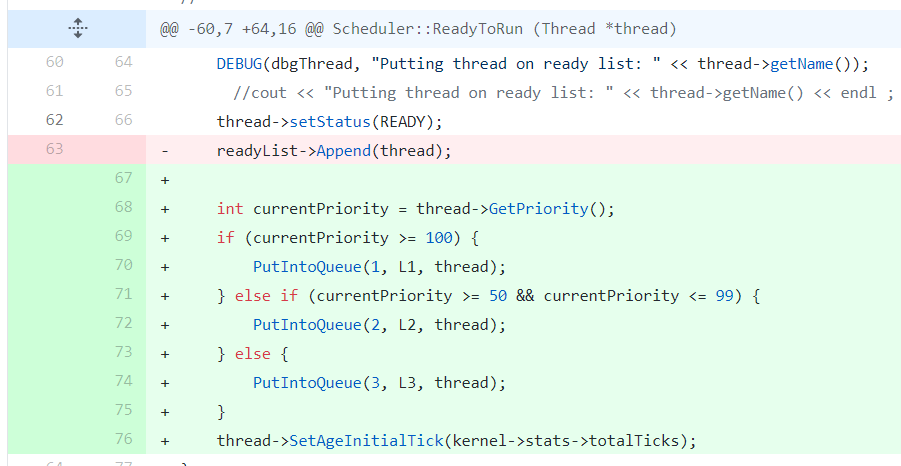
TODO: 12/16 23:00

把那張圖的流程跟code的概念結合起來，接下來對照作業要求還有學長的code，把作業要求要填的code填上。

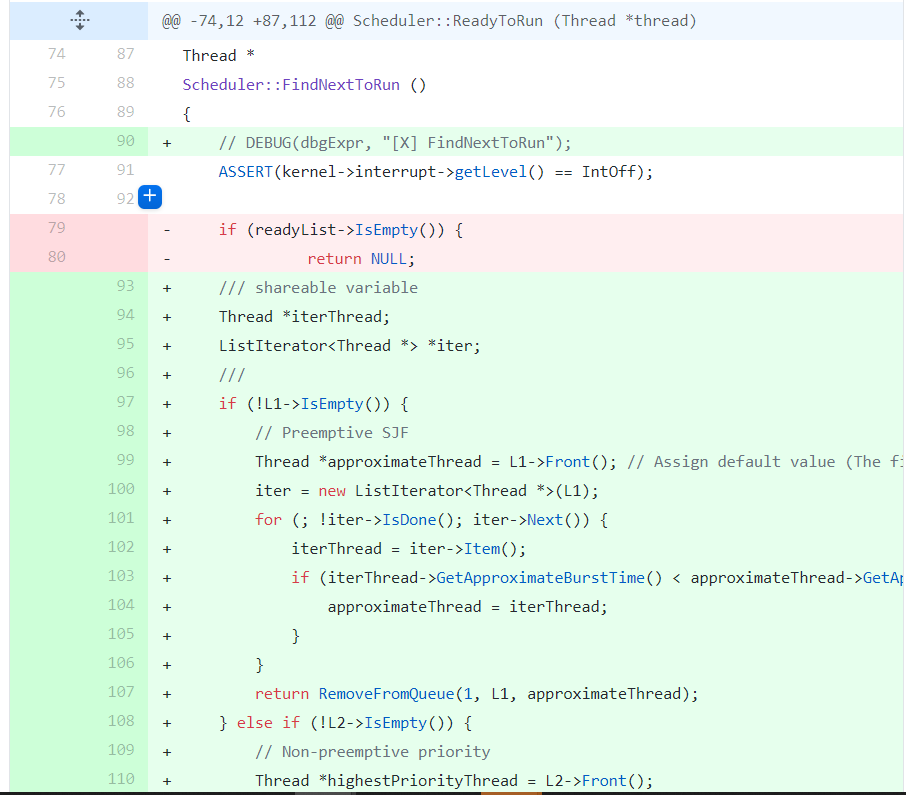
Sjf, preemptive是每個tick都會檢查嗎？

12/18 TODO：

ReadyToRun，把thread放進不同的ready queue裡面。

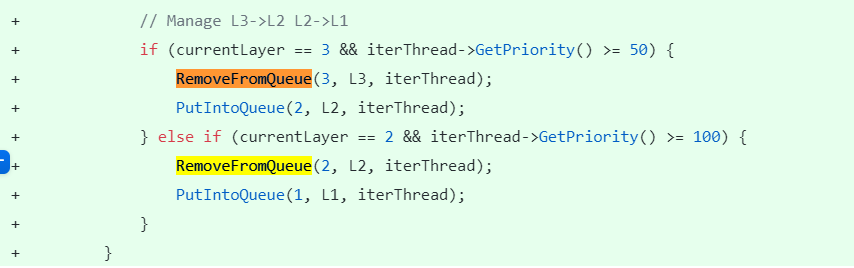


FindNextToRun，從ready queue拉出應該要接著執行的thread。



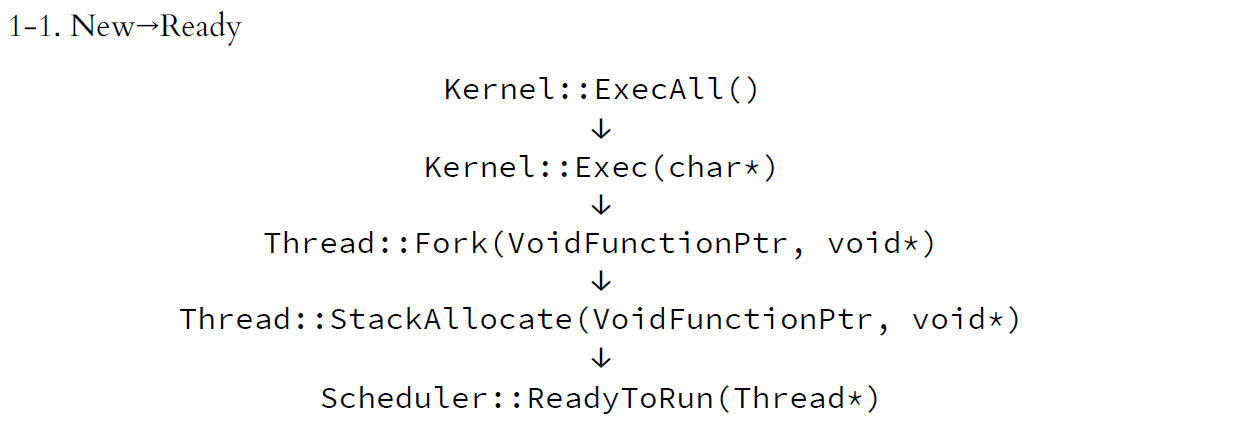
3. DEBUG signal

4. 要不要把token換成priority

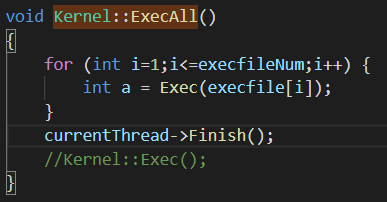


TODO: CheckifDue是幹嘛的?

1. Predicted burst time variable and method
2. Scheduler aging and queuing(put in and take out) operation



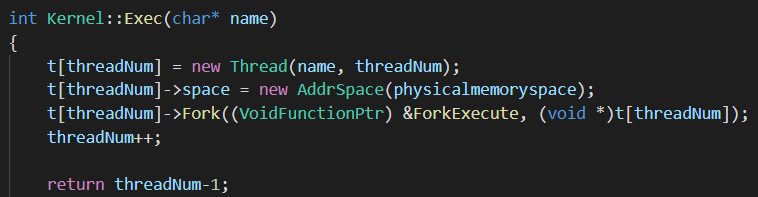
Kernel::ExecAll()



目的：執行每一支程式。

解釋：OS把要執行的數個程式名稱丟到Exec()裡面，有幾個檔案就做幾次，執行完之後結束。

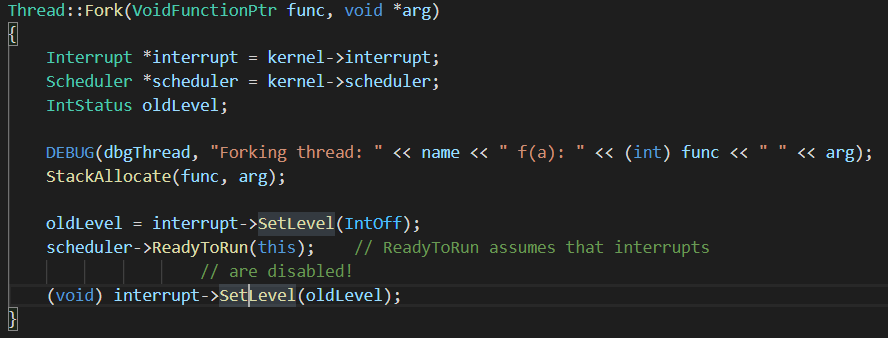
Kernel::Exec(char\* name)



目的：執行一支程式。

解釋：為這支新程式創建一個Thread，給Thread記憶體空間，接著Thread fork。

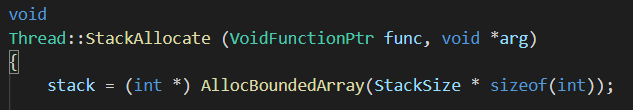
Thread::Fork(VoidFunctionPtr func, void\* arg)

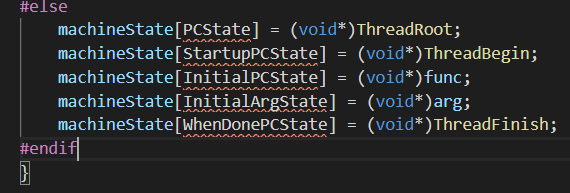


目的：把這個thread變成是可讓CPU執行的thread。

解釋：首先把kernel的interrupt和scheduler抓到當前的thread，再來開創stack的空間給這個thread，都準備好之後，把這個thread放到ready queue裡面。而在scheduler執行ReadyToRun過程中，要把interrupt關閉，結束之後再回到之前的interrupt狀態。

Thread::StackAllocate(VoidFunctionPtr func, void\* arg)

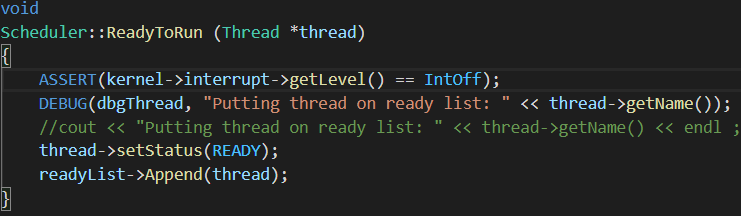




目的：register state and stack initialization for this thread

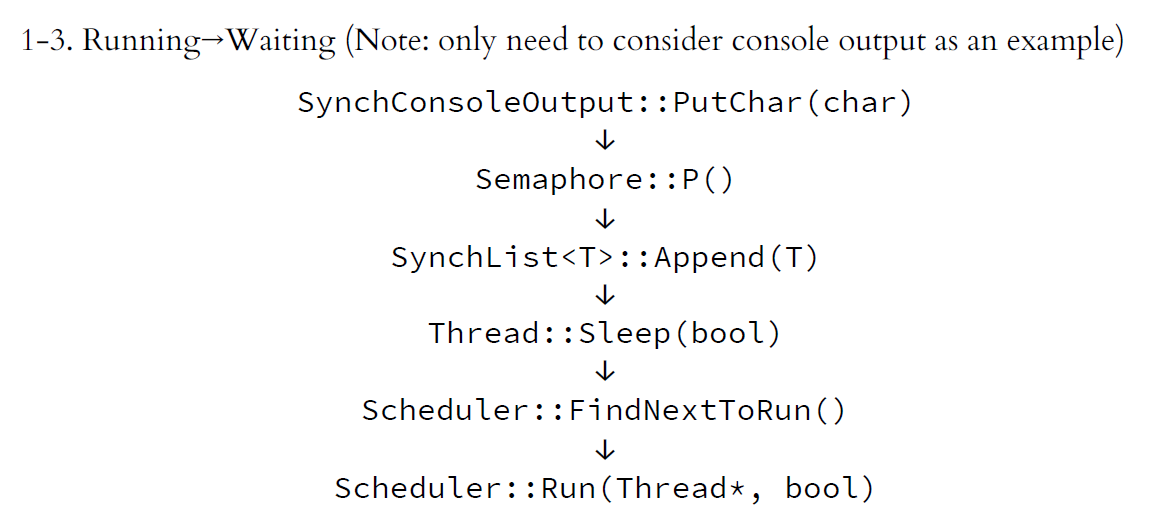
解釋：創建記憶體空間給這個thread，並設定此thread的初始狀態以便此thread進入running state時，能夠讓CPU正常執行。

Scheduler::ReadyToRun(Thread\* thread)

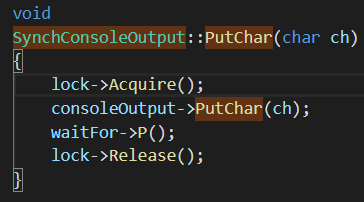


目的：讓thread加到ready queue裡面，才能排程進入running state。

解釋：設定thread的狀態為ready，並把thread加到ready queue裡面。



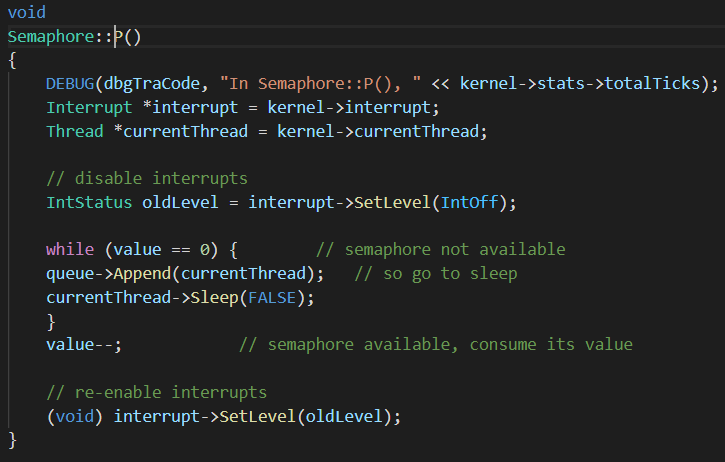
SynchConsoleOutput::PutChar(char)



目的：把一個字元display到console上。

解釋：由於I/O不可同時進行，所以要等拿到lock之後才可以做I/O。另外為了不要同時改到Semaphore的value，所以寫了waitFor->P()，確定沒人用P()之後，再把lock的資源釋放回系統。

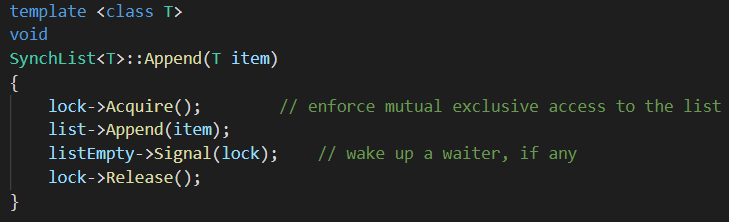
Semaphore::P()



目的：分配資源給需要I/O的Thread。

解釋：首先關閉interrupt，接著判斷現在是否有資源，若沒有則讓thread加到waiting list，等到再次輪到他時，Semaphore再看看有沒有資源可以給他；若有資源則將I/O資源減1，如此一來這個Thread就可以執行I/O。

SynchList<T>::Append(T)



目的：管控I/O資源。

解釋：首先為了不要同時跟其他function使用list，這裡有monitor在管控，所以一開始先拿lock，拿到之後才能append item到list，做完後signal任何想access list的thread，通知那些thread現在可以access list了，接著release lock。

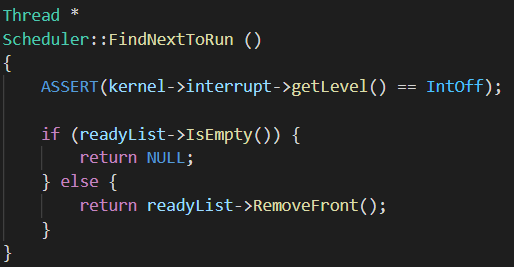
Thread::Sleep(bool)



目的：把目前的thread suspend，讓其他thread先使用CPU。

解釋：首先檢查要sleep的是不是current thread，以及是否已關閉interrupt。接下來把thread state設成waiting，並找ready queue裡面是否有在等待CPU的thread，如果沒有就等到有，有了之後就讓給他CPU。

Scheduler::FindNextToRun()



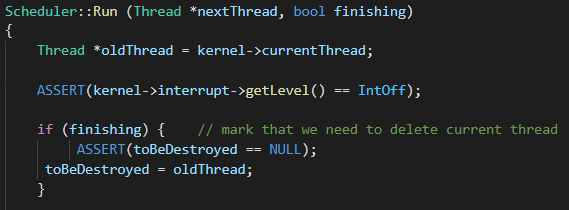
目的：在ready queue找一個可以跑的thread。

解釋：首先檢查是否已關閉interrupt。接下來看看ready queue有沒有thread，有的話回傳那個thread，並將那個thread從queue中移除，不然就回傳NULL。

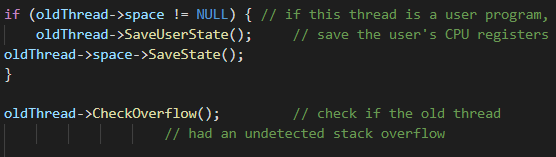
Scheduler::Run(Thread\*, bool)

目的：把目前正在執行的thread（舊的thread），換成另一個準備要使用CPU的thread（新的thread）。

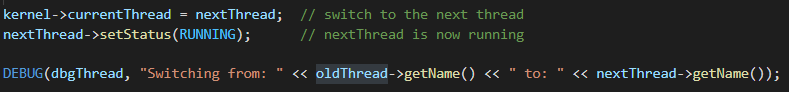
解釋：



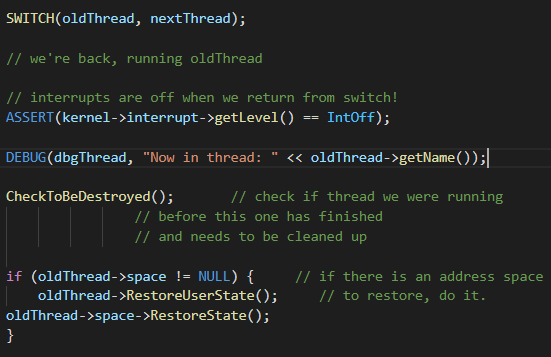
首先看看目前舊的thread是否已做完該做的工作，做完就紀錄給toBeDestroyed，下面的CheckToBeDestroyed會刪除這個thread。



接下來幫舊的thread儲存CPU register狀態，並且看看stack有無overflow。



把currentThread換成新的thread，並且將新的thread的state換成running。



執行context switch，結束後檢查interrupt是否是關閉狀態，接著看看舊的thread是否要被移除，最後把新的thread的CPU register state load回來，使得新的thread可以回到之前做到一半的地方繼續做下去。