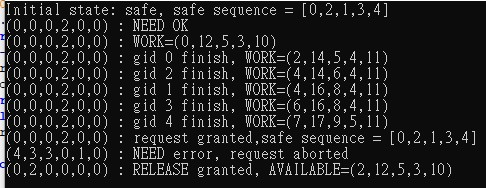
作業系統作業四報告

1081447-李明昕

編譯過程 :



基本功能 :

a. 使用getline讀資料，在用trans把string轉成vector。

b. 讀到request時，代表allocation已經完成了，此時先計算need，然後再判斷現在的state安不安全。

c. 在讀完request之後，進到while迴圈直到所有的request跑完，a-(分別check need/avalible/safe state，若是都OK，就把need/available/allocation都更新，若不是就依照規定aborted/skip。)r-(只需要判斷該gid可不可以release那麼多，若是可以就修改該gid的need/allocation和avalible，不行則skip)

function/variable說明 :

trans(string, int start) : string是讀進來的字串，start表示從哪裡開始轉換，因為max/allocation都有第幾個gid這個input，不過我用vector存，存在vector的第幾格就表示他是第幾個gid了，所以在一開始就先把它改淘汰掉。alu是string轉int的暫存器，轉換完畢後就可以將其放進compute。(compute專門用來存input)

print() : 因為我們會一直印request，所以我把印request這個動作放進這個function。

testneed(int j = -1) : 將這個function分成兩個case，當沒有傳參數進來的時候表示是判斷該gid還可不可以request；當有傳參數進來的時候代表現在在進行safe state的判斷，此時判斷可不可以將該gid放回avalible。

testava() : 判斷系統有沒有足夠的avalible給該gid執行。

testsafe() : fisish就是banker中的FINISH，sequence則是存回收的順序。work就是banker的WORK，依照FCFS的規則回收，若是遇到可以回收的就將其回收，當全部都回收成功時，才修改該gid的need/allocation和avalible，若是不行就回報unsafe。(work專門在這邊使用)

testrelease() : 判斷該gid有沒有足夠的allocation可以釋放，若是不行就skip，可以就修改該gid的allocation/need和avalible。

ava就是banker中的AVALIBLE、Max、allocation、need亦同。

request比較不一樣，因為他的第一格存gid，所以對應的時候要多+1，並且request的最後一格存a/r(這部分使用if-else判斷)。