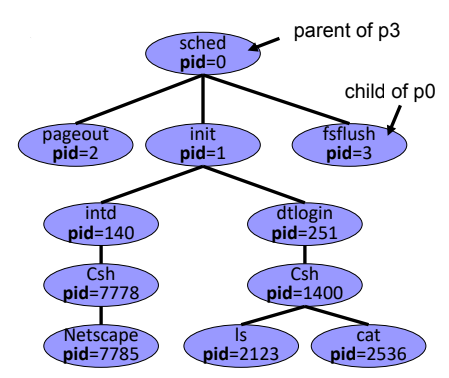
\*process可看作一顆tree

(unix: 使用ps –ael 去看)

每個process都有一個獨特的process ID (PID)

並且每個process是樹的關係(除init之外都有parent)



Process parent-child的關聯性設計理念

1. resource sharing程度

共享資料、部分共享(如靜態var共享，動態不共享)、完全不共享

1. 執行順序

同時(cocurrently)、子程序先(父程序wait)

1. 子程序創造方法

完全複製 (連var、pc都複製) 、 透過messege passing 載入新程式

**以Unix為例: 這只是其中一種實作選擇**

Process creation

呼叫system call fork() -> execlp() -> wait()

1.Fork() (create process)

先要一塊新address空間

**完全**複製一份 parent (連變數”當前狀態”也是 (參考example))

(實作上複製一份會較小，因為有些唯讀狀態會用address傳)

父子cocurrenly的執行

Parent fork() 會return child的PID

Child fork() 會 return 0 (因為parent 使用fork，child一定也有fork那行，然後return 值是0)

2. execlp() (reset new process)

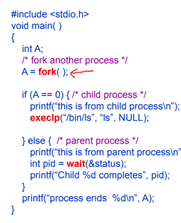
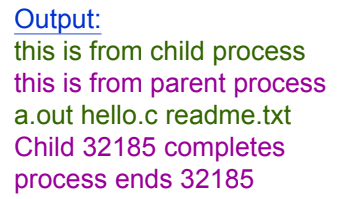
會清空heap、stack，然後載入新的binary file(code)

3 wait()

強迫parent 等待child

Recall system call希望越輕量越好

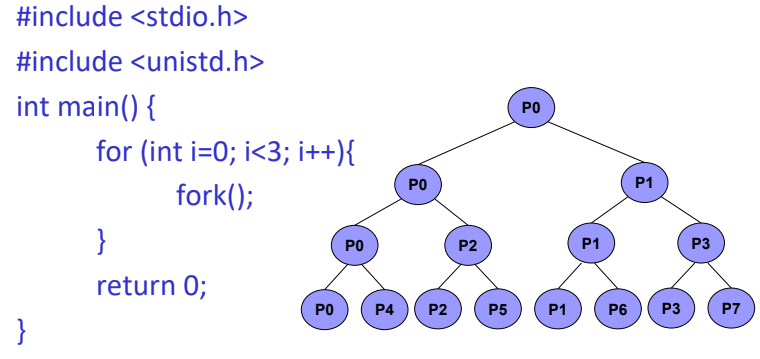
Example1

關鍵，child 得到的A是0，parent得到的A是child PID

關鍵2 child 在 execlp之後已經不會執行最後printf(….)那行了

Example2



關鍵，在fork()之後，PC沒有被reset，因此會i++ (相當於fork()後往下跑喔)

可以自己出Example試試

Process termination

Call exit() (將所有空間、資源還給OS)

Parent 可以 kill child by its PID

Ctrl+C 就是讓當前console (console算是很高級的parent) kill執行程式(child)