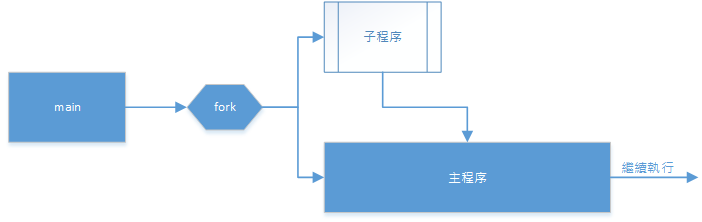
作業系統 Program1 – fork() and shell

電機4A 106501001 顏郁芩

開發環境：Ubuntu下利用C++開發

問題一：fork()

程式邏輯：



函式功能說明：

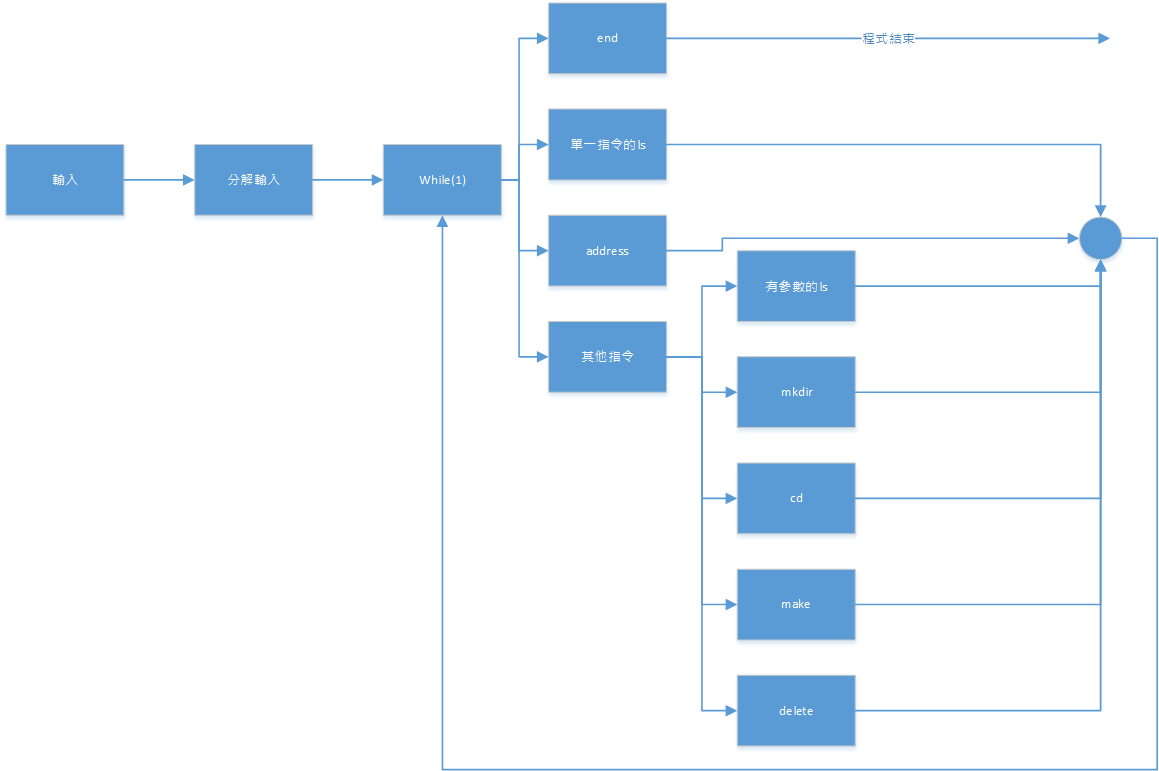
Fork這一個程式可以把主程式複製一次變成子程序，兩個像是叉子一樣併行，因為不共用記憶體空間所以可以平行處理。

自己寫的程式可以輸出程序的執行碼(pid)，藉此來分辨主程序與子程序，子程序的PID會是0。另外利用了wait這一個函數去等待子程序回傳值才結束主程序，可以使子程序先完成。

如果沒有利用wait或是其他的控制方式的話，因為兩個是平行進行所以執行結束的先後順序不太一定，如果主程序先結束的話會導致子程序變成殭屍程序，會關不掉。

問題二：shell

程式邏輯：





程式邏輯方向是用「輸入指令」當作while迴圈的重複執行標準，在輸入之後把指令用linked-list的方式存起來(利用struct的方式)，然後再用一個一個讀取的方式來分析指令。

如上圖所示，能執行的指令分成單一指令的end、ls、address還有複數(兩個)指令的ls+參數、mkdir+檔案名稱、cd+資料夾名稱/..、make+檔案名稱和delete+檔案/資料夾名稱這幾個。

自己並沒有把shell功能寫得非常齊全，只有把自己覺得shell很基本的例如ls和創造/刪除檔案/資料夾等功能寫了進去。

函式功能說明：

|  |  |  |
| --- | --- | --- |
| 輸入 | 功能 | |
| end | 程式結束 | |
| ls | 列出當下的路徑裡面所有的檔案和資料夾 | |
| address | 印出當下的路徑 | |
| ls+參數 | 根據不同參數印出當下路徑裡面所有檔案和資料的詳細內容 | |
| mkdir+資料夾名 | 建立資料夾 | |
| Cd+資料夾名 | 進入資料夾 | 改變執行路徑 |
| cd .. | 返回上一層 |
| make+檔名 | 創造檔案 | |
| Delete+名稱 | 刪除檔案/資料夾 | |

另外在測試過程中發現了一個問題但是不太確定是因為什麼造成的，就是在輸入ls或是ls+參數之後再輸入別的指令，有一定的機率會無法執行，但有時候又正常所以一時之間也找不出fix的方法。不過再輸入一次又會正常執行。

在思考會不會是因為自己使用了fork的方式去寫卻沒有注意到執行順序的問題才導致這個結果，因為ls的實作部分自己用到了execve這一個執行式，如果沒有用fork的話執行下去主程序會被execve取代掉而直接跳出去，沒有辦法重複執行。