討論與心得

在本次實作中，我們設計了一個程式，模擬執行以下指令的行為：

```
ps aux | grep 41147011S | wc > the_result
```

並且測試了執行結果，對比如下：

1. 使用自行編寫的程式產生的結果：

```
41147011S@telnet:~/SystemSoftware$ gcc -o ps_grep_wc ps_grep_wc.c
41147011S@telnet:~/SystemSoftware$ ./ps_grep_wc
41147011S@telnet:~/SystemSoftware$ cat the_result
      3      37     255
```

2. 使用 shell 指令直接執行產生的結果：

```
41147011S@telnet:~/SystemSoftware$ ps aux | grep 41147011S | wc > the_result1
41147011S@telnet:~/SystemSoftware$ cat the_result1
      3      38     268
```

探討：為什麼兩個結果不同？

結果之所以不同，主要原因可能來自以下幾個方面：

1. `grep` 的行為差異

在直接執行 shell 指令時， `grep 41147011S` 會捕捉到執行該指令時產生的行程資訊。也就是說，

```
ps aux | grep 41147011S
```

的輸出中可能包含了 `grep 41147011S` 本身的行程資訊。

然而，在程式設計中，我們使用了 `execlp` 呼叫，可能未包括 `grep` 本身的執行環境，這使得輸出行數和內容略有不同。

2. 字元計數差異

`wc` 工具會分別計算行數、字數與字元數。雖然行數一致，但字數和字元數的差異可能來自於：

- Shell 指令執行時，可能在管線中添加了額外的空白字元或換行符號。
- 系統呼叫可能在輸出過程中略微調整了結果格式。

3. 輸出重定向影響

在 `./ps_grep_wc` 程式中，我們直接將輸出透過 `dup2` 重定向至檔案 `the_result` 。而 shell 的重定向可能會增加或調整某些輸出格式（例如結尾的換行符號），導致最終計算結果不一致。

學到的東西

透過本次實作，我們深入理解了 UNIX 系統呼叫（如 `pipe` 、 `fork` 、 `dup2` ）的運作原理，尤其是管線的構建與資料流的處理。同時，我們觀察到系統呼叫與 shell 指令執行的細微差異，這些差異強調了操作系統對於進程與輸入輸出管理的細節處理能力。

這次實驗還讓我們體會到程式開發中的可重現性挑戰，特別是在涉及多進程與系統資源競爭時。我們的程式結果與 shell 指令不同雖屬正常，但也提醒我們，在開發與測試時應特別關注這些細節，並嘗試解釋其背後的原因。

結論

兩個結果的不同是正常的，因為程式執行與 shell 指令存在環境與行為上的差異。透過這次實作，我們學會如何分析這些差異，並強化了對系統呼叫與 shell 操作的理解。未來在開發中，我們應更加留意輸出的一致性以及各工具之間的行為差異，從而提升程式的可預測性與穩定性。