3-5

# 處理程序的衍生

# 處理程序的衍生

- 學習完本單元，您將可以：
  - 了解處理程序如何進行衍生、執行、等待

# 處理程序之運作（Process Operating）

- 處理程序之運作功能包含
  - 衍生（fork）
  - 執行（execve）
  - 等待（wait）
  - 離開（exit）
  - 放棄（abort）
- 均為系統呼叫。

# 衍生（fork）(1)

- 任何處理程序呼叫fork()，使得系統衍生一個新的處理程序。
- 被生出的處理程序稱為子處理程序（Children Process）。
- 生出處理程序的處理程序稱為父處理程序（Parent Process）。
- 子處理程序與父處理程序長相一樣，有相同的程式碼並繼承父處理程序的資源。(Code, Data, Resource, Stack, Program Counter, Register Set)

# 衍生（fork）(2)

```
parent( )
{ pid = fork( );
   if( pid = 0)
          { play c( );
            exit(0);
           }
  play p( );
  exit(0);
}
```

Parent Process

```
parent( )
{ pid = fork( );
If (pid = 0)
     { play c( );
       exit(0);
     }
    play p( );
     exit(0);
   }
```

Children Process

# 衍生（fork）(3)

在系統內被fork( )的處理程序之pid與父處理程序之pid均不為0，但子處理程序繼承父處理程序為呼叫fork( )前的pid，故看到自己的pid為0，因此子處理程序與父處理程序會執行不同程式碼。

```
parent( )
{ pid = fork( );
   if( pid = 0)
         { play c( );
           exit(0);
          }
  play p( );
  exit(0);
}
```

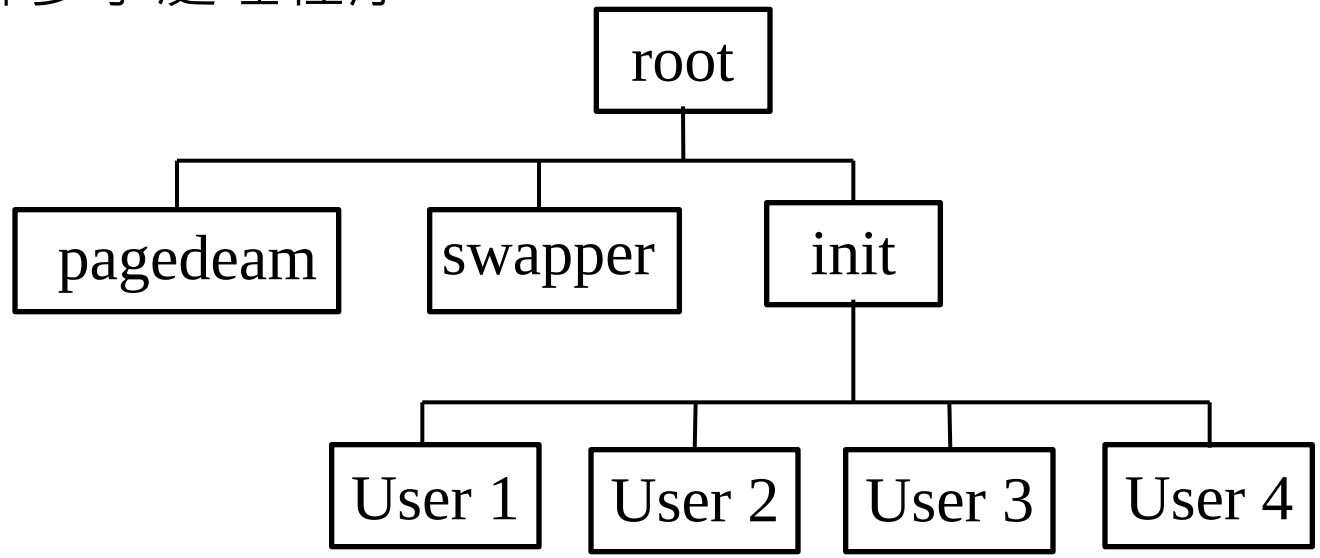
Parent Process

```
parent( )
{ pid = fork( );
If (pid = 0)
    { play c( );
      exit(0);
    }
  play p( );
   exit(0);
}
```

Children Process

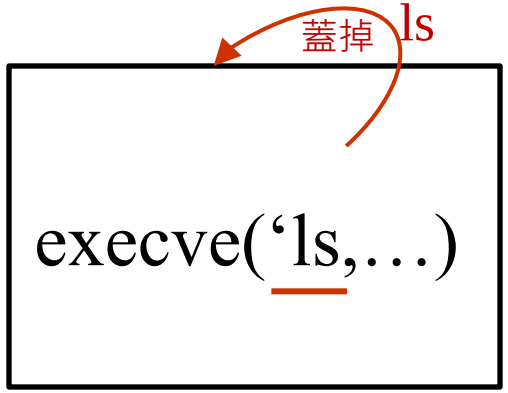
# 衍生（fork）(4)

- 父處理程序可以衍生許多子處理程序，而子處理程序亦可再衍生許多子處理程序。



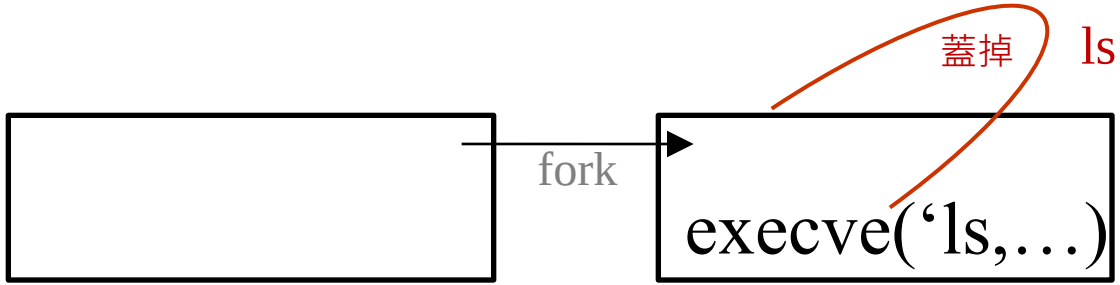
# 執行（execve）

- 當一個處理程序呼叫execve(“ls,null,null”)時，系統會衍生ls這個處理程序，並把ls這個處理程序之程式碼載入主記憶體內，且蓋掉呼叫execve( )的這個處理程序的地址空間；亦即執行呼叫execve( )這個處理程序會自行結束執行，並將它的控制權交至ls這個被呼叫的處理程序。



# 執行（execve）

- execve( )系統呼叫與fork( )系統呼叫最大不同在於，執行fork( )系統呼叫後，父處理程序與子處理程序均存在系統內，有各自的地址空間。而執行execve( )系統呼叫後，僅有被生出之處理程序存在，原處理程序結束執行。



# 等待（wait）及離開（exit）

- 等待（wait）
  - 父處理程序執行wait( )呼叫，進入懸置狀態，當它的所有子處理程序結束後，父處理程序才由懸置狀態進入備妥狀態。

- 離開（exit）
  - 處理程序欲正常結束，必須呼叫exit( )以通知作業系統正常結束，並將控制權交回作業系統。

# 放棄（abort）

- 作業系統或處理程序可以呼叫abort( )，結束處理程序之執行。

- 通常在作業系統內，若父處理程序結束工作，則不論子處理程序是否繼續工作，作業系統會放棄(abort)子處理程序，這就是連帶終止（Cascading Termination）。