HW11 signal

408410019 資工二 徐怡娟

1. signal 的意義

   (1) SIGCHLD：子程序結束後會傳給父程序訊號

   (2) SIGSTOP：暫停程序的訊號（不可被捕獲）

   (3) SIGKILL：強制終止程序的訊號（不可被捕獲）

   (4) SIGINT：當按下 ctrl-c 時終止程式

   (5) SIGALRM：設定一個鬧鐘，用 alarm( )決定幾秒後要發出訊號（鬧鐘響）

2. 攔截 SIGINT，並印出『按下 ctrl-c，但殺不死我』

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ./happyRon
本task的學號是 5749
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
```

3. 承上題，如果快速的按下十次 ctrl+c，會出現多少次『按下 ctrl-c，但殺不死我』

   signalhandler 只需要做 printf 的動作，花費時間極少（下一個 ctrl-c pending 馬上就會被處理），因此每次按 ctrl-c 都能及時捕捉到

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ./happyRon
本task的學號是 3830
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
^C按下ctr-c，但殺不死我
```

4. 承上題，在 SIGINT 的處理函數中，加入 sleep(10)，此時如果快速按下十次 ctrl+c，會出現幾次『按下 ctrl-c，但殺不死我』

兩次，每次按下 ctrl-c 需處理十秒鐘，十秒內再次接收到 ctrl-c 只會有一個留下來 peding，其他都直接被丟掉

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ./happyRon
本task的學號是 3953
^C按下ctr-c，但殺不死我
^C^C^C^C^C^C^C^C^C按下ctr-c，但殺不死我
```

5. 承上題，此時如果使用 fork+execv 讓 child 執行『ls-alh /-R』。當『ls-alh /-R』時，按下 ctrl+c 時，是否會出現『按下 ctrl-c，但殺不死我』

觀察：在程式碼為下圖時會出現『按下 ctrl-c，但殺不死我』且 ls 會被殺掉

```
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c3000-7f0d998c5000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-apply.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c5000-7f0d998c6000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-apply.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c6000-7f0d998c7000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-apply.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c7000-7f0d998c8000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-heuristics.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c8000-7f0d998c9000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-heuristics.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998c9000-7f0d998ca000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-heuristics.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998ca000-7f0d998cb000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-heuristics.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998cb000-7f0d998cc000 -> /usr/lib/pulse-13.99.1/modules/modul
e-filter-heuristics.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998cc000-7f0d998ce000 -> /usr/lib/pulse-13.99.1/modules/modul
e-snap-policy.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998ce000-7f0d998d0000 -> /usr/lib/pulse-13.99.1/modules/modul
e-snap-policy.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998d0000-7f0d998d1000 -> /usr/lib/pulse-13.99.1/modules/modul
e-snap-policy.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998d1000-7f0d998d2000 -> /usr/lib/pulse-13.99.1/modules/modul
e-snap-policy.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998d2000-7f0d998d3000 -> /usr/lib/pulse-13.99.1/modules/modul
e-snap-policy.so
lr-------- 1 accepted accepted 64 May 26 00:53 7f0d998d3000-7f0d998d5000 -> /usr/lib/pulse-13.99.1/modules/modul
e-role-cork.so
^C按下ctr-c，但殺不死我
```

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ps -a
    PID TTY          TIME CMD
   1633 tty2     00:00:27 Xorg
   1656 tty2     00:00:00 gnome-session-b
   8520 pts/0    00:00:08 happyRon
   8521 pts/0    00:00:00 ls <defunct>
   8523 pts/1    00:00:00 ps
```

程式碼：

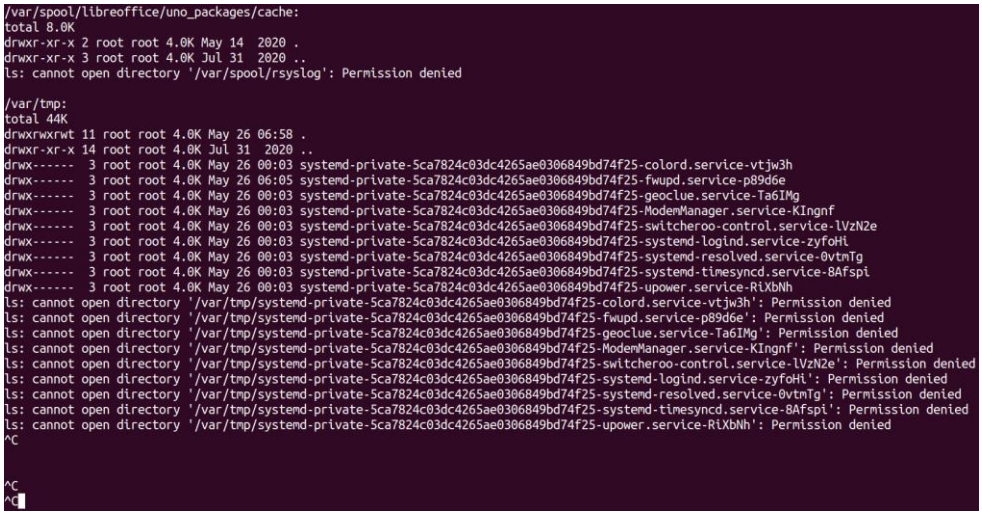
```
signal(SIGINT, sighandler);
int child_pid = fork();
if (child_pid ==0) {
    execlp("ls", "ls","/","-alhR",NULL);
} else {    //parent
    printf("chld的pid是%d\n", child_pid);
    while(1) {
        ;
    }
}
```

討論：

ctrl-c 後 parent 和 child 都收到 SIGINT，parent 觸發 signalhandler，child 則是維持 ctrl-c 預設的動作（terminate process），理論上 child 應該要繼承 parent 所註冊的 signal，child 卻維持預設動作?

6+7. 請使用範例程式加以修改，忽略 SIGINT（SIG_IGN），此時如果使 fork+execv 讓 child 執行『ls』。當『ls』時，按下 ctr+c 時，是否可以終止『ls』的執行

ls 執行的中途按 ctrl-c，signal 被忽略因此無法終止 child，只能等 child 執行完後停止，接下來繼續按 ctrl-c 也無法終止 parent（如 ps -a 結果，child 執行結束後變為 zombie）



```
accepted@ubuntu:~/system-programming/ch10/HW11$ ps -a
    PID TTY          TIME CMD
   1633 tty2     00:01:44 Xorg
   1656 tty2     00:00:00 gnome-session-b
  12875 pts/0    00:00:55 happyRon
  12876 pts/0    00:00:22 ls <defunct>
  12880 pts/1    00:00:00 ps
```

程式碼：

```
signal(SIGINT, SIG_IGN);
int child_pid = fork();
if (child_pid ==0) {
    execlp("ls", "ls","/","-alhR",NULL);
} else {    //parent
    printf("chld的pid是%d\n", child_pid);
    while(1) {
        ;
    }
}
```

討論：

在 5.的討論中提到『child 應該要繼承 parent 所註冊的 signal，child 卻維持預設動作?』，而在這裡 child 的確繼承 parent 所註冊的 signal，問題是否出在 execlp?

驗證：不要使用 execve 系列函數再跑跑看

(1) child 執行一般的程式碼，signal 註冊自定義 signalhandler

程式碼：

```
signal(SIGINT, sighandler);
int child_pid = fork();
if (child_pid ==0) {
    while(1) {
        printf("child\n");
        sleep(1);
    }
} else {    //parent
    printf("chld的pid是%d\n", child_pid);
    while(1) {
        ;
    }
}
```

結果：

parent 和 child 都觸發 signalhandler，child 沒有被 terminate，sleep(10) 後繼續執行，總共等待十秒（猜測是 parent 和 child 同時一起 sleep(10)，沒有輪流 sleep(10)）

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ./happyRon
本task的學號是 12997
chld的pid是12998
child
child
child
child
child
child
^C按下ctr-c，但殺不死我
按下ctr-c，但殺不死我
child
child
child
child
child
```

```
accepted@ubuntu:~/system-programming/ch10/HW11$ ps -a
    PID TTY          TIME CMD
   1633 tty2     00:01:49 Xorg
   1656 tty2     00:00:00 gnome-session-b
  12997 pts/0    00:00:20 happyRon
  12998 pts/0    00:00:00 happyRon
  12999 pts/1    00:00:00 ps
```

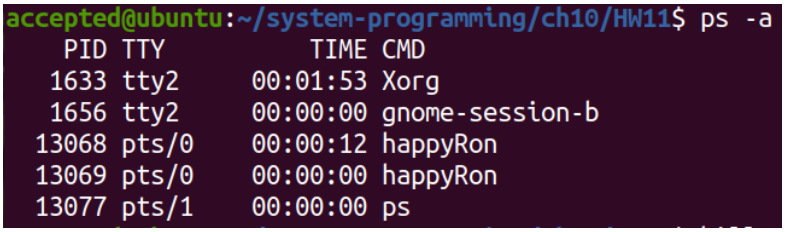
(2) child 執行一般的程式碼，signal 使用預設選項（SIG_IGN）

程式碼：

```c
signal(SIGINT, SIG_IGN);
int child_pid = fork();
if (child_pid ==0) {
    while(1) {
        printf("child\n");
        sleep(1);
    }
} else {    //parent
    printf("chld的pid是%d\n", child_pid);
    while(1) {
        ;
    }
}
```

結果：

parent 和 child 都忽略 ctrl-c 的 signal，child 繼續執行



結論：

主程式註冊的 signal 會被 fork 出來的子程序繼承，但若子程序中有使用 execve 系列函數，只有註冊為預設選項（SIG_ DFL / SIG_IGN）才會被執行的應用程式採納，若為自定義的 handler 則會執行預設的行為（ctrl-c -> terminate process）

8. 我認為 OS 提供對組合鍵統一的預設的詮釋方式是有必要的，像是 ctrl-c 對於大部分的應用程式來說都希望能終止程式執行，如果 OS 沒有提供統一的介面，每個應用程式都必須另外自己寫一段程式碼來處理同一件事有點沒必要，加上若程式設計師沒把組合鍵處理的 code 寫好（漏洞?惡意?），可能會造成一些無法預期的結果。