408410007鄭雅云

學習重點:pipe+signal()+setpgid()

2.//結果相當於執行: ls -R /home | wc -l

```
#include <assert.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int pid1;
void sighandler(int signum) {
        kill(-1*pid1, signum);
        exit(0);
}
int main(int argc, char **argv) {
   int pipefd[2];
   int ret, wstat, pid2;
   //char **param={"EXENAME", NULL};
   pipe(pipefd);
   pid1 = fork();   //產生第一個child
   if (pid1==0) {
        setpgid(0,0);
      close(1);   //關閉stdout
      dup(pipefd[1]); //將pipefd[1]複製到stdout
      close(pipefd[1]);   //將沒用到的關閉
      close(pipefd[0]);   //將沒用到的關閉
      execlp("ls", "ls", "-R", "/home", NULL);   //執行ls, ls會將東西藉由stdout輸出到pipefd[1]
   } else printf("1st child's pid = %d\n", pid1);
   if (pid1>0) {
      pid2 = fork();//產生第二個child
      if (pid2==0) {
         setpgid(0,pid1);
         close(0);   //關閉stdin
         dup(pipefd[0]); //將pipefd[0]複製到stdin
         close(pipefd[1]);   //將沒用到的關閉
         close(pipefd[0]);   //將沒用到的關閉
         execlp("wc","wc","-l", NULL);   //執行wc, wc將透過stdin從pipefd[0]讀入資料
      } else printf("2nd child's pid = %d\n", pid2);
   }
close(pipefd[0]); close(pipefd[1]);  //parent一定要記得關掉pipe不然wc不會結束(因為沒有接到EOF)
signal(SIGINT, sighandler);
printf("child %d\n",wait(&wstat));
printf("child %d\n",wait(&wstat));
}
```

4.完成1及2及3, 能做到2.3

能做到2:

加入:“signal(SIGINT, sighandler);” 偵測ctr+c

用kill(-1*pid1, signum);　　//殺掉process group, 讓『ls -R /home | wc -l』立即中止

能做到3:

(1)將signal刪掉

(2)將setpgid(0, 0); 及 setpgid(0, pid1); 這兩行刪除, 在終端機送出sigint後, 因為他們隸屬同一個foreground process group, 所以parent和child會一起死掉(若不刪, 第一個

child第二個child被設定成新的process group，終端機送出sigint後，就只有在foregound process 的 parent會死掉，child不會）