

Esercitazione su gestione base dei segnali



Roberto De Virgilio

Sistemi operativi - 8 Gennaio 2018

Esercizio 0

- ◆ *Dato il file es0.c, dire cosa stampa su output:*

```
#include <stdio.h> #include <signal.h>
#include <unistd.h> #include <stdlib.h>

int val = 10;

void handler(int sig){
    val += 8;
}

int main(void){
    pid_t pid;
    signal(SIGCHLD, handler);
    if ((pid = fork()) == 0) {
        val -= 3;
        exit(0);
    }
    waitpid(pid, NULL, 0);
    printf("val = %d\n", val);
    exit(0);
}
```

Esercizio 0

- **Output:**

```
$ gcc es0.c -o es0
```

```
$ ./es0
```

```
val = 18
```

```
$
```

Esercizio 1

- ◆ *Dato il file **es1.c**, dire cosa stampa su output:*

```
#include<stdio.h>
#include<signal.h>
#include <unistd.h>
#include <stdlib.h>

pid_t pid;
int counter = 0;

void handler1(int sig){
    counter++;
    printf("counter = %d\n", counter);
    kill(pid, SIGUSR1);
}

void handler2(int sig){
    counter += 3;
    printf("counter = %d\n", counter);
    exit(0);
}
```

Esercizio 1

- ◆ *Dato il file **es1.c**, dire cosa stampa su output:*

```
int main(){
    pid_t p;
    int status;
    signal(SIGUSR1, handler1);
    if ((pid = fork()) == 0){
        signal(SIGUSR1, handler2);
        kill(getppid(), SIGUSR1);
        while(1) ;
    }
    if ((p = wait(&status)) > 0){
        counter += 4;
        printf("counter = %d\n", counter);
    }
}
```

Esercizio 1

- **Output:**

```
$ gcc es1.c -o es1
```

```
$ ./es1
```

```
counter = 1
```

```
counter = 3
```

```
counter = 5
```

```
$
```

Esercizio 2

- ✿ Scrivere il file **es2.c**, che stampi infinite volte su output il messaggio “**hello world**”. Ogni volta che stampa tale messaggio deve poi mettersi a “dormire” per un secondo;
- ✿ solo quando l’utente premerà la combinazione **Ctrl+C** il programma catturerà il segnale e lo gestirà stampando su output il messaggio “**Terminated**” e quindi poi terminerà l’esecuzione del programma

Esercizio 2

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

void handle_sigint(int sig){
    printf("\nTerminated\n");
    exit(0);
}

int main(void){
    signal(SIGINT, handle_sigint);
    while (1)
    {
        printf("hello world\n");
        sleep(1);
    }
    return 0;
}
```

Esercizio 2

- **Output:**

```
$ gcc es2.c -o es2
```

```
$ ./es2
```

```
hello world
```

```
^C
```

```
Terminated
```

```
$
```

Esercizio 3

- ✿ Scrivere il file **es3.c**, che metta in “pausa” l'esecuzione all'infinito attendendo che l'utente premi la combinazione **Ctrl+C**
- ✿ quando l'utente preme tale combinazione, il programma catturerà il segnale ignorandolo; successivamente presenterà all'utente il seguente messaggio su output

Forse hai premuto per sbaglio Ctrl-C?
Vuoi realmente sospendere l'esecuzione [y/n]?

- ✿ nel caso in cui l'utente digiterà ‘**y**’ o ‘**Y**’ allora l'esecuzione terminerà altrimenti il programma continuerà a restare in pausa attendendo di nuovo che l'utente fornisca la combinazione di tasti

Esercizio 3

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

void INThandler(int sig) {
    char c;
    signal(sig, SIG_IGN);
    printf("\nForse hai premuto per sbaglio Ctrl-C?\nVuoi realmente
sospendere l'esecuzione [y/n]?");
    c = getchar();
    if (c == 'y' || c == 'Y')
        exit(0);
    else
        signal(SIGINT, INThandler);
}

int main(void) {
    signal(SIGINT, INThandler);
    while (1)
        pause();
}
```

Esercizio 3

- **Output:**

```
$ gcc es3.c -o es3
```

```
$ ./es3
```

^C

Forse hai premuto per sbaglio Ctrl-C?

Vuoi realmente sospendere l'esecuzione [y/n]? n

^C

Forse hai premuto per sbaglio Ctrl-C?

Vuoi realmente sospendere l'esecuzione [y/n]? y

\$

Esercizio 4

- ✿ Scrivere il file **es4.c**, che utilizzi la seguente sintassi di invocazione
`./es4 com_1 com_2 ... com_N`
- ✿ dove **com_1** ... **com_N** sono nomi di comandi eseguibili (come **ps**, **ls**, etc...). Si ipotizzi N con valore 10;
- ✿ Il processo iniziale (**P₀**) deve creare N processi figli (**P₁**, **P₂**, .. **P_N**).
- ✿ Il processo figlio **P_i**, una volta creato, deve porsi in attesa di un'eventuale attivazione da parte del padre. In caso di attivazione, **P_i** eseguirà il comando **com_i**.

Esercizio 4

- ◆ Il processo padre (*Po*), una volta creati i figli, definirà il suo comportamento in base al valore del proprio *pid* (*ppid*):
 - ◆ se *ppid* è *pari*, attiverà i figli con *pid pari* e terminerà forzatamente i figli con *pid dispari*;
 - ◆ se *ppid* è *dispari*, attiverà i figli con *pid dispari* e terminerà forzatamente i figli con *pid pari*;
- ◆ Successivamente, dopo aver raccolto e stampato lo stato di terminazione di tutti i figli, il padre terminerà la propria esecuzione.

Esercizio 4

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

#define max 10

void gestore_att(int sig){
    printf("%d: sono stato attivato!\n", getpid());
    return;
}

void figlio(char *com){
    int miopid;
    miopid=getpid();
    if ((miopid%2)==0)
        signal(SIGUSR1, gestore_att);
    else
        signal(SIGUSR2, gestore_att);
    pause();
    execlp(com, com, (char *)0);
}
```

Esercizio 4

```
int main(int argc , char *argv[]){
    int ppid, pid[max], pf;
    int status, i;
    if (argc==1){
        printf("sintassi sbagliata!\n");
        exit(1);
    }
    ppid=getpid();

    for(i=0; i< argc-1; i++){
        pid[i]=fork();
        if (pid[i]==0){
            figlio(argv[i+1]);
            exit(0);
        }
        else
            printf("%d: creato figlio %d\n", ppid, pid[i]);
    }
}
```

Esercizio 4

```
sleep(2); /* solo per ritardare...*/
if ((ppid%2)==0)
/*attivazione dei pari e uccisione dei dispari:
 SIGUSR1 */
    for(i=0; i<argc-1; i++)
        kill(pid[i], SIGUSR1);
else
    for(i=0; i<argc-1; i++)
        kill(pid[i], SIGUSR2);

for (i=0; i<argc-1; i++){
    pf=wait(&status);
    if ((char)status==0)
        printf("terminato %d con stato %d\n", pf, status);
    else
        printf("terminato %d involontariamente (segnale %d)\n",
               pf, status);
}
exit(0);
} /* fine padre */
```

Esercizio 4

- **Output:**

```
$ gcc es4.c -o es4
```

```
$ ./es4 ps ls pwd date
```

```
4318: creato figlio 4319
```

```
4318: creato figlio 4320
```

```
4318: creato figlio 4321
```

```
4318: creato figlio 4322
```

```
4322: sono stato attivato!
```

```
4320: sono stato attivato!
```

```
terminato 4321 involontariamente (segnaletico 30)
```

```
terminato 4319 involontariamente (segnaletico 30)
```

Esercizio 4

- **Output:**

```
$ gcc es4.c -o es4
```

```
$ ./es4 ps ls pwd date
```

...

```
Sab 30 Dec 2017 14:04:08 CET
```

```
terminato 4322 con stato 0
```

```
es0.c es1.c es2.c es3.c es4.c
```

```
terminato 4320 con stato 0
```

```
$
```



```
Current conditions at Pescara, Italy (IBP) 42.26N 014.12E 11M (IBP)
Last updated Feb 10, 2012 - 02:50 PM EST / 2012-02-10 1950 UTC
Temperature: 1 C
Relative Humidity: 80%
Wind: from the W (270 degrees) at 15 MPH (13 KT) gusting to 45 KPH
Weather: light snow grains
Sky conditions: overcast
Su Mo Tu We Th Fr Sa      Su Mo Tu We Th Fr Sa
feb 29 30 31 01 02 03 04    mar 04 05 06 07 08 09 10
  05 06 07 08 09 10 11    11 12 13 14 15 16 17
  12 13 14 15 16 17 18    18 19 20 21 22 23 24
  19 20 21 22 23 24 25    25 26 27 28 29 30 31
mar 26 27 28 29 01 02 03    apr 01 02 03 04 05 06 07
silvio@Stain ~ $ cd Video
silvio@Stain ~ /Video $ movgrab http://vimeo.com/27998081
Formats available for this Movie: flv
Selected format: flv
Progress: 61.47% 15.4M of 25.1M 693.6K/s
```

