Laboratorio di Sistemi Operativi, A.A. 2022–2023

Simulazione di esercitazione

Lunedì 29 maggio 2023

Esercizio 1

Scrivere un programma C `xo1.c` consistente in:

- un processo "genitore" che
  - crea una pipe anonima e ne utilizza il descrittore di output come stdout;
  - crea un figlio;
  - scrive la frase `Ciao, mondo!` nello standard output una volta al secondo.
- un processo "figlio" che
  - stampa il proprio process ID sul terminale;
  - utilizza il descrittore di input della pipe anonima creata dal genitore come standard input;
  - stampa a terminale quanto legge dallo standard input trasformando tutte le "o" in "x".

L'output del programma (che dovrà chiamarsi `xo1`, senza estensioni) sarà dunque:

```
# ./xo1
12345
Ciax, mxndx!
Ciax, mxndx!
Ciax, mxndx!
...
```

Esercizio 2

Scrivere il programma C `xo2.c` in cui il processo figlio è modificato in modo che:

- alla ricezione del segnale `SIGUSR1` cessi di convertire le "o" in "x" e stampi le stringhe lette senza più convertirle;
- alla ricezione del segnale `SIGUSR2` ripristini il comportamento precedente.

L'output del programma (che dovrà chiamarsi `xo2`) dovrà essere:

```
# ./xo2
12345
Ciax, mxndx!
Ciax, mxndx!
Ciao, mondo!
Ciao, mondo!
Ciao, mondo!
Ciax, mxndx!
...
```

dove il processo figlio avrà ricevuto (ad esempio da un altro terminale) il segnale `SIGUSR1` dopo due secondi dalla sua entrata in funzione e `SIGUSR2` dopo cinque secondi.

Esercizio 3

Realizzare uno script bash `xo.sh` che riceva come parametro il PID del processo figlio e ogni tre secondi ne modifichi il comportamento inviando i segnali opportuni.
Ad esempio, se il processo figlio ha stampato il PID `12345` come negli esempi percedenti, una volta dato (in un secondo terminale) il comando

```
# bash xo.sh 12345
```

il processo `xo2` dovrà stampare ripetutamente tre righe `Ciao, mondo!` e tre righe `Ciax, mxndx!`.
Bonus se lo script potrà essere invocato come `./xo.sh 12345` senza menzionare esplicitamente la shell.

Esercizio 4

Realizzare un Makefile con i seguenti quattro obiettivi:

- `xo` crea l'eseguibile dell'esercizio 1;
- `xo2` crea l'eseguibile dell'esercizio 2;
- `all` (obiettivo di default) entrambi gli eseguibili;
- `clean` cancella tutti i file eseguibili, lasciando solo i file sorgenti scritti dallo studente.

Indicazioni utili

Le seguenti informazioni servono solo a facilitare l stesura del codice, ma non sono in alcun modo obbligatorie.

- Comandi bash utili (manuale accessibile con `man 1` *nomecomando*): `sleep`, `kill`, `chmod`, `true`.
- Comandi bash built-in utili (manuale accessibile con `help` *nomecomando*): `while`.
- Chiamate di sistema utili (manuale accessibile con `man 2` *nomefunzione*): `fork`, `pipe`, `close`, `dup2`, `signal`, `getpid`.
- Altre funzioni utili (manuale accessibile con `man 3` *nomefunzione*): `printf`, `puts`, `fflush`, `getchar`, `putchar`, `sleep`.
- Per la scrittura della stringa sullo standard output, il processo genitore può usare la chiamata

  ```
  puts("Ciao, mondo!");
  fflush(stdout);
  ```

  La funzione `puts()` agguinge automaticamente un carattere di a capo; per assicurarsi che l'invio sia immediato, la `fflush()` svuota il buffer di output.
- Dopo la stampa del proprio PID con una `printf()` e la ridirezione dello standard input dalla pipe, il figlio può travasare un carattere alla volta dallo standard input allo standard output con le funzioni

  ```
  char c = getchar();
  putchar(c);
  ```

  Ovviamente, per gli scopi dell'esercizio sarà necessario trasformare le `'o'` in `'x'`.
- I seguenti file header dovrebbero essere sufficienti per tutte le funzioni utilizzate nell'esercitazione:

  ```
  #include <stdio.h>
  #include <sys/types.h>
  #include <unistd.h>
  #include <signal.h>
  ```