ISTRUZIONI PRATICHE

Esame del modulo di laboratorio di “Sistemi Operativi”

Durata: 120’ (2 ore)

● Creare una cartella principale denominata con il proprio numero di matricola e dentro tutti i contenuti richiesti dal compito.

● Questa cartella andrà consegnata “zippandola” (compressione formato “zip”) in modo da creare un file avente per nome il proprio numero di matricola più l’estensione “.zip”. Deve essere compressa l’intera cartella e non solo il suo contenuto.

Se il proprio numero di matricola fosse 123456 questo deve essere anche il nome della cartella e l’archivio compresso da consegnare deve chiamarsi 123456.zip. All’estrazione dovrà essere presente la cartella 123456.

● Consegna:

  ○ Dopo 50’ ed entro 60’ dall’inizio della prova si deve fare una prima consegna (lavoro parziale) con il lavoro compiuto complessivo fino a tale momento ANCHE SE NON FUNZIONANTE utilizzando il modulo nell’homepage del browser. Nominare l’archivio parziale con “**<matricola>_parziale.zip**”

  ○ Dopo 90’ ed entro 120’ dall’inizio della prova si deve fare una seconda consegna (lavoro finale) utilizzando il modulo nell’homepage del browser: questa consegna è l’unica considerata per la valutazione finale.

  ○ Per consegnare, effettuare il login con il proprio account universitario e selezionare il file “zip” da allegare.

  ○ I punteggi [x] sono indicativi dato che la valutazione tiene conto anche di dettagli “trasversali” che non sono riferibili a singoli punti. Un punteggio complessivo maggiore o uguale a 31 porterà alla lode.

NOTA: parte delle verifiche può avvenire con procedure completamente o parzialmente automatizzate per cui le denominazioni e gli output devono essere rigorosamente aderenti alle indicazioni.

Eventuali irregolarità comportano l’esclusione dalla prova oltre a possibili sanzioni disciplinari.

Consegna

Si richiede la creazione di un sorgente di nome “`signal_handler.c`” e di un makefile “`Makefile`”. Il makefile deve creare un eseguibile dal nome “`signal_handler.out`” solamente tramite il comando “`make compila`” ([3]), mentre il sorgente deve svolgere le seguenti funzionalità descritte di seguito:

1. [4] Alla ricezione di un qualsiasi segnale `SIGXCPU`, `SIGTTOU` e `SIGTTIN`, il programma stampa su **stdout** il numero totale dei segnali di quel tipo ricevuti.
   Per esempio, se il programma ha ricevuto 5 segnali `SIGXPU` e 1 segnale `SIGTTOU`, alla ricezione di un nuovo segnale `SIGTTOU` esso dovrà stampare “`SIGTTOU_2`”. Alla ricezione di un nuovo segnale `SIGTTOU` dovrà stampare “`SIGTTOU_3`”, mentre alla ricezione di `SIGXPU` dovrà stampare “`SIGXPU_6`”.
   NB: l’output deve essere esattamente “*`<nomeSegnale>_<numeroRicezioni>`*” con terminatore di riga finale.
2. [8] Alla ricezione di un qualsiasi segnale **`SIGFPE`** e **`SIGUSR1`**, il programma deve mandare un messaggio sulla coda identificata dalla tupla **`('/tmp/file',5)`** contenente il nome del segnale ricevuto (es: “SIGFPE” o “SIGUSR1”) e di tipo uguale al PID del processo mittente.
   NB: il programma deve solo inviare il messaggio senza poi leggerlo!
   NB: se la coda non esiste deve essere creata!
3. [8] Alla ricezione di un qualsiasi segnale **`SIGUSR2`**, il programma deve generare un figlio. Alla ricezione di un qualsiasi segnale **`SIGXFSZ`**, il programma deve uccidere il figlio attivo (non terminato) più anziano. Per esempio, se il programma ha ricevuto 3 segnali **`SIGUSR2`** andando quindi a creare, rispettivamente, i figli dal PID 34, 43 e 50, alla ricezione di un segnale **`SIGXFSZ`** il programma dovrà terminare il figlio dal PID 34. Al successivo **`SIGXFSZ`** dovrà terminare il figlio 43. Come terminare il figlio è a discrezione dell’esaminato, ma è importante che venga liberata ogni risorsa appartenente a quel figlio, così come ogni riferimento nel sistema a quel processo.
   NB: per semplificare, in fase di controllo dell’elaborato verranno generati al massimo 10 figli.
4. [8] Alla ricezione di un qualsiasi segnale **`SIGUSR2`**, oltre alle operazioni del punto 3, il programma deve creare una pipe anonima con il nuovo figlio generato → ogni figlio avrà una pipe anonima con il padre. Alla ricezione di un segnale **`SIGPROF`**, il programma deve inviare solo al figlio più giovane tramite l’apposita pipe anonima il PID del penultimo figlio generato (0 se esiste solo un primo figlio). Alla sua ricezione, il figlio più giovane deve stampare su **stderr** tale PID ricevuto. **NB**: solo l’ultimo figlio generato riceve il messaggio.
   Esempio: il programma avviato riceve **`SIGUSR2`** e genera un primo figlio con il PID 34. Alla ricezione di **`SIGPROF`** il programma invierà al primo figlio tramite pipe anonima “0” (dato che non esiste un penultimo figlio) e quest’ultimo dovrà stamparlo su **stderr** “0”. Alla successiva ricezione di **`SIGUSR2`** il programma genera un nuovo

figlio con PID 43 ed, alla ricezione di **SIGPROF** invierà ad esso tramite pipe anonima "34" (il PID del penultimo figlio), facendo così stampare al processo 43 "34" su **stderr**. Al figlio successivo (es con PID 80) verrà eventualmente inviato "43" e così via.

5. [4] Alla ricezione del segnale **SIGQUIT**, il programma dovrà inviare un segnale di **SIGQUIT** a tutti i figli ancora attivi, aspettare la loro terminazione, e poi terminare. I figli, e solamente i figli, alla ricezione di **SIGQUIT** dovranno invece stampare su **stderr** "QUITTING" con terminatore di linea prima di terminare.

NB:

- **La mancata presenza di opportuni commenti nel codice può portare a delle penalizzazioni.**
- **Se il file sorgente o il makefile si chiamano in maniera differente, ci sarà una penalità di punteggio**
- **La generazione di warnings nella compilazione porterà alla perdità di punti**
- **Errori di compilazione porteranno alla completa invalidazione della prova**

Aiuti alla consegna

Snprintf
Per convertire un intero in stringa potete usare

```
#include <stdio.h>
int snprintf(char *str, size_t size, const char *format, ...);
```

Che, similmente a `printf`, scrive al più `size` bytes (contanto eventuali terminatori di stringa) di una stringa formattata in `str`. Esempio:
`snprintf(&buf, bufSize, "%d", 22);`
per scrivere la stringa "22" in `buf`

TODO:
Prima di consegnare, può essere utili controllare la seguente lista di todo:

- ☐ Il file zip si chiama correttamente e contiene solo i due file?
- ☐ Il sorgente di chiama signal_handler.c?
- ☐ Il makefile si chiama Makefile?
- ☐ Il programma usa correttamente stdout e stderr secondo le indicazioni date?
- ☐ Il programma stampa gli output seguendo il formato specificato?
- ☐ Il programma usa i giusti segnali nelle varie parti?
- ☐ Il programma è opportunamente commentato?
- ☐ La compilazione genera warnings?