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.

  ○ Tutti 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. I punteggi del tipo [x→y] variano in base alla qualità della soluzione, al risultato e alla modalità di esecuzione. Soluzioni che utilizzano tecniche IPC viste a lezione avranno una valutazione maggiore.

NOTA: le denominazioni e gli output devono essere rigorosamente aderenti alle indicazioni.

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

Consegna

NB:

- **La mancata presenza di opportuni commenti nel codice può portare a delle penalizzazioni.**
- **La generazione di warnings nella compilazione porterà alla perdita di punti.**
- **Errori di compilazione porteranno alla completa invalidazione della prova.**
- **L'esame verrà corretto a sezioni (Preparazione, Espionage, Coup d'état, Impostor). Non è necessario che il programma implementi tutte le sezioni.**

Preparazione:

1. [3] Il programma deve accettare esattamente due parametri: un numero `<n>` ed il percorso di un file `<file>`. Parametri errati devono portare alla terminazione del programma con un codice d'errore seguendo le seguenti indicazioni:
   a. Un numero di parametri diverso → codice di errore 10.
   b. Se `<n>` non è compreso tra [1,20] (inclusi) →codice di errore 11.
   c. Se `<file>` non è accessibile → codice di errore 12.
2. [4] Il processo principale deve creare `<n>` figli (tutti figli diretti). Una volta creati, i figli devono **bloccare ogni segnale** possibile ad eccezione di `SIGUSR1`, `SIGUSR2` e `SIGTSTP`. I figli devono dunque rimanere attivi (**senza consumare cicli CPU**).
   Hint: da bash potete usare il comando "`ps`" per controllare la parentela.
3. [4] Il processo principale deve creare una nuova **fifo** `/tmp/fifo`. Inoltre, ogni figlio creato deve essere in comunicazione con il padre attraverso almeno una pipe anonima.
   HINT: Da bash potete verificare se la fifo esiste usando l'operatore unario "`-p`".

Espionage:

4. [4] Il processo riceverà dei messaggi (inviati al momento della valutazione) sulla fifo `/tmp/fifo` sopra creata, contenenti una stringa con il seguente formato "`<PIDchild>-<PIDdst>`". Ogni qualvolta viene ricevuto un tale messaggio, il processo deve stamparlo su **stdout** (eventuale fflush se necessario).
   Es: i messaggi potrebbero essere "`33-55`", "`32-62`", "`30-35`", etc…, ognuno da stampare su stdout (nuova linea opzionale).
5. [5] …In ogni messaggio, la sottostringa `<PIDchild>` contiene il PID di uno dei figli creati, mentre la stringa `<PIDdst>` è un PID qualunque. In aggiunta al punto 4, il processo principale deve dunque inviare un messaggio al figlio `<PIDchild>` contenente la stringa "`<PIDdst>`" usando la pipe anonima creata al punto 3.

Quando il figlio `<PIDchild>` riceve tale messaggio, esso dovrà inviare un segnale `SIGUSR1` al processo con PID=`<PIDdst>`.
Es: riprendendo l'esempio precedente:

a. Processo principale riceve "`33-55`" → processo manda "`55`" sulla pipe anonima al figlio `33` → figlio `33` invia segnale `SIGUSR1` al processo `55`.
b. Processo principale riceve "`30-35`" → processo manda "`35`" sulla pipe anonima al figlio `30` → figlio `30` invia segnale `SIGUSR1` al processo `35`.
c. …
Questo in aggiunta al punto 4!

Coup d'état:

6. [3] Quando un figlio riceve un segnale `SIGUSR1`, esso deve creare un nuovo gruppo di processi e diventare group leader.
7. [4→7] Quando un figlio riceve un segnale `SIGUSR2`, esso si deve unire **all'ultimo** gruppo creato da uno degli altri figli (in fase di valutazione verrà inviato `SIGUSR2` solo dopo aver inviato almeno un segnale `SIGUSR1`, non necessariamente allo stesso processo). Come rendere il processo a conoscenza del gruppo a cui unirsi è a discrezione dell'esaminato.
Es: vengono inviati i seguenti segnali ai figli:
   - `SIGUSR1`→30 →30 diventa leader
   - `SIGUSR1`→34 →34 diventa leader
   - `SIGUSR2`→35 →35 si unisce al gruppo di 34 (non di 30!)
   - `SIGUSR1`→33 →33 diventa leader
   - `SIGUSR2`→34 →34 si unisce al gruppo di 33.

Impostor:

8. [3] Quando un figlio riceve un segnale `SIGTSTP`, esso deve **immediatamente** ed **esclusivamente** sostituire il proprio binario/immagine con il binario `/tmp/impostor.out`, diventando dunque l'impostore. Il destinatario non può fare altre operazioni alla ricezione del segnale! **NB**: il binario per il testing è contenuto nello zip del testo d'esame.
9. [4→7] Quando il processo principale riceve il segnale `SIGTTOU`, il processo deve essere in grado di capire (entro massimo 1s circa) quale figlio sia diventato l'impostore (senza che nessun figlio termini). Il PID dell'impostore, e solo il suo, deve essere scritto sul file `<file>` ricevuto come parametro al lancio del programma.
NB: Viene mandato solo un segnale `SIGTTOU` e solo un segnale `SIGTSTP`! C'è quindi solo un impostore da trovare!
Es: il processo (PID 10) crea inizialmente 6 figli aventi come PID 30→36. Vengono inviati i seguenti segnali:
   - `SIGTSTP`→32 →32 diventa l'impostore.
   - `SIGTTOU`→10 →il processo scrive "32" sul file `<file>`