Corso di Sistemi Operativi e Reti

Prova scritta 26 NOVEMBRE 2021

ISTRUZIONI PER CHI È IN PRESENZA:

1. **Rinomina** la cartella chiamata "Cognome-Nome-Matricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali e **lasciando i trattini**; se hai un doppio nome oppure un doppio cognome dovrai chiamare la cartella come in questo esempio:
    a. DeLuca-MarcoGiovanni-199999
2. **Carica** tutto il materiale didattico che vorrai usare sul Desktop; puoi farlo solo nei primi 5 minuti della prova;
3. **Svolgi** il compito; lascia tutto il sorgente che hai prodotto nella cartella di cui al punto 1;
4. Quando hai finito lascia la postazione facendo logout,

senza spegnere il PC.

SALVA SPESSO

ISTRUZIONI PER CHI SI TROVA ONLINE:

1. **Questo file contiene il testo che ti è stato dato ieri, incluso il codice;**
2. **Mantieni a tutto schermo** questo file per tutta la durata della prova; puoi scorrere liberamente tra le sue pagine, ma non puoi cambiare applicazione;
3. **Firma** preliminarmente il foglio che userai per la consegna con nome cognome e matricola;
4. **Svolgi** il compito; puoi usare solo carta, penna e il tuo cervello;
5. **Aiutati** con i numeri di linea per indicare le eventuali modifiche che vorresti fare al codice che ti è stato dato.
6. **Alla scadenza** termina *immediatamente* di scrivere, e attendi di essere chiamato, pena l'esclusione dalla prova;
7. **Quando è il tuo turno** mostra il foglio ben visibile in webcam, e poi metti una foto dello stesso foglio in una chat privata Microsoft Teams con il prof.

CI SONO DEI PUNTI AMBIGUI NELLA TRACCIA? COMPLETA TU

È parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo nuove strutture dati, o estendendo quelle preesistenti laddove si ritenga necessario, risolvendo eventuali ambiguità. Si può cambiare il codice dei metodi esistenti dove serve.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI O DI QUELLI ESISTENTI? NO

Non è consentito modificare il prototipo dei metodi se questo è stato fornito. Potete aggiungere qualsivoglia campo e metodo di servizio, e qualsivoglia classe ausiliaria, ma NON variare l'interfaccia dei metodi pubblici già specificati. Analogamente, i metodi esistenti possono essere modificati nel loro codice, ma non se ne deve cambiare il risultato finale o il significato.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X

Il linguaggio da utilizzare per l'implementazione è Python 3. Ricorda che l'operatore di formattazione `f` (esempio, `f"Ciao sono la stringa {testo}"`) è disponibile solo dalla versione 3.6 di Python in poi, ma può essere sostituito con `"Ciao sono la stringa %s" % testo`

MA IL MAIN() LO DEVO AGGIORNARE? E I THREAD DI PROVA? SI

E' obbligatorio implementare esplicitamente del codice di prova oppure modificare il codice di prova pre-esistente, e accertarsi che giri senza errori prima della consegna.

ESERCIZIO 1 - PROGRAMMAZIONE MULTITHREADED

Punto 1:

Si osservi che nel codice fornito, per via di problemi di context switch da individuare, l'ordine delle stampe non sempre corrisponde alla sequenza di gioco che ci si aspetterebbe. Ad esempio, può verificarsi che a video appaia la sequenza

```
RED LIGHT!!!!
GREEN LIGHT!!!!
KILLING PLAYER 016 AAHAHAHAHAHA!!!!
PLEASE PLEASE NO NO NO NO DON'T KILL MEEEE AAAAAAHHHH. [RIP PLAYER 016]!!!!
KILLING PLAYER 198 AAHAHAHAHAHA!!!!
```

Che lascia apparentemente pensare che è stato eliminato un Player quando la luce del semaforo è GREEN. Si corregga opportunamente il codice per ottenere una sequenza delle stampe corretta. Si salvi la versione corretta al punto 1 col nome `SquidGame-Punto1.py`.

Punto 2:

Partendo dal codice originale contenuto in `SquidGame.py`, si modifichi il codice in maniera tale da affidare le eliminazioni dei Player a un Thread `Killer`. Solo il Thread killer può invocare il metodo `UnDueTreStella.kill(num)`. Quando un'altra tipologia di Thread deve effettuare una o più eliminazioni, si deve commissionare l'eliminazione stessa usando il nuovo metodo `UnDueTreStella.ordinaEliminazione(num)`. Tale metodo commissiona una richiesta di eliminazione al Thread killer ed esce immediatamente. Il Thread killer attende invece l'arrivo di richieste di eliminazione. Queste ultime vengono smaltite in ordine di arrivo FIFO dal thread killer stesso, il quale invoca il metodo `UnDueTreStella.kill(num)` in accordo alla corrispondente richiesta di eliminazione. Il Thread killer ha la facoltà di eliminare i Player in qualsiasi momento del gioco (sia con luce RED che con luce GREEN che a tempo scaduto).

Punto 3:

Si fornisca una variante del gioco in cui il timer si avvia, anziché subito, dal momento in cui il primo giocatore si salva. Dal momento in cui si salva il primo giocatore, il gioco deve terminare se:

1. Si sono salvati 50 Player, oppure
2. E' scaduto il tempo del timer

Al termine del gioco, tutti i restanti Player devono essere eliminati utilizzando il killer implementato al punto 2.

Deve essere possibile avviare il gioco a scelta in una delle due modalità, dunque la classe UnDueTreStella dovrà possedere un costruttore nella forma:

```
def __init__(self,specialMode = False)
```

Allorquando *specialMode = True*, il gioco dovrà svolgersi nella modalità speciale di cui al Punto 3.

Si salvino le modifiche apportate al Punto 2 e al Punto 3 (**anche se si è svolto solo un punto o parte di entrambi i punti)**, nel file `SquidGame-Punto2-Punto3.py`

MATERIALE PER LA PROVA SULLA PROGRAMMAZIONE MULTI-THREADED

ESERCIZIO 1 (Programmazione multithread. Punti: 0-20)

Il codice fornito implementa il gioco di “Un, due, tre, Stella!”, nella versione resa famosa dalla serie The Squid Game.

457 diversi Player devono riuscire a compiere un certo numero di passi N entro lo scadere di un certo tempo limite T (`gameDuration` nel codice). Un semaforo che si alterna a intervalli temporali casuali tra i valori RED e GREEN stabilisce i momenti in cui i Player possono muoversi. Se un Player si muove in un momento in cui il semaforo è RED, il Player in questione viene immediatamente eliminato. Un Player può invece compiere liberamente dei passi nei momenti in cui il semaforo è GREEN. Quando un Player compie più degli N passi richiesti esso si può considerare salvo (N=`winningLine` nel codice). Allo scadere del tempo T, tutti i Player che non hanno ancora compiuto i passi richiesti per salvarsi vengono eliminati e infine il gioco termina definitivamente.

Ulteriori dettagli sono disponibili nei commenti inclusi nel codice fornito.

SALVA SPESSO

SALVA SPESSO

ESERCIZIO 2, TURNO 1 - PERL

Si scriva un script dal nome `squid.pl` in grado di monitorare ad intervalli di tempo regolare (ogni 3 secondi) lo stato dei processi attivi sul sistema. Lo script dovrà essere in grado di arrestare tutti i processi che superano una determinata soglia di utilizzo di `CPU` o `MEMORIA` in base alle opzioni passate dall'utente durante l'avvio dello script.

Di seguito è riportata la sinossi dello script da implementare:

```
./squid.pl OPTION
```

dove `OPTION` può essere

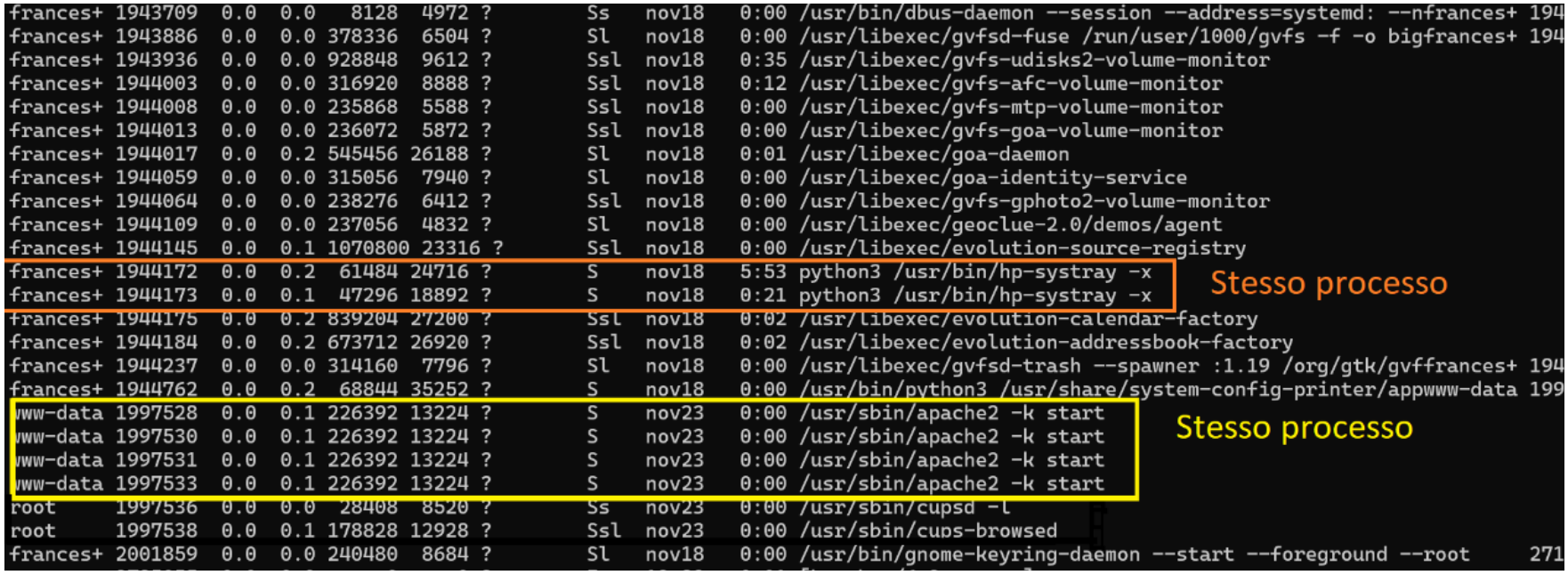
- `CPU=n`
- `MEM=n`

ed `n` è un valore numerico compreso tra `0` e `100`

Più in dettaglio, lo script funzionerà nel seguente modo:

1. una volta avviato lo script attende 3 secondi prima di monitorare lo stato dei processi attivi nel sistema (è possibile usare un qualsiasi comando Unix/Linux a vostra scelta per ottenere la lista completa dei processi attivi sul sistema);
2. ottenuta la lista dei processi, lo script analizza la `CPU` o la `MEMORIA` **cumulativa** (in base all'opzione inseritadall'utente) occupata da ogni singolo processo. Per `CPU` e `MEMORIA` **cumulativa** si intende la somma dei valori di tutti i sottoprocessi relativi allo stesso servizio;

ESEMPIO: Si supponga di voler utilizzare il comando `ps` con le opportune opzioni per ottenere la lista di tutti i processi attivi nel sistema. Un possibile output potrebbe essere il seguente:



L'immagine mostra alcuni dei processi tra loro correlati (stesso identico nome processo). Per il processo `python3 /usr/bin/hp-systray-x` la quantità totale di `CPU` occupata è `0.0` mentre la `MEMORIA` occupata è `0.3`.
N.B. potrebbero essere presenti altri processi dello stesso tipo non evidenziati nell'immagine.

3. Se il valore cumulativo `CPU` o `MEMORIA` supera il valore indicato tramite argomenti in input dall'utente, quel processo verrà killato dallo script (si usi il comando Unix/Linux `kill` seguito dalla lista di tutti i PID di cui effettuare l'arresto o, in alternativa, il comando `pkill` seguito dal nome del processo da terminare);
4. Lo script attende nuovamente 3 secondi per poi ripartire dal punto 1 finchè non sarà terminato manualmente dall'utente.

ATTENZIONE: Non eseguite il comando `kill/pkill` durante l'esame. Potreste effettuare l'arresto di processi di sistema provocando quindi lo spegnimento inatteso del vostro PC, perdendo eventuali dati non salvati (cioè il vostro esame!!!).