NON SPEGNERE IL PC A FINE ESAME

Corso di Sistemi Operativi e Reti

Prova scritta di APRILE 2019

ISTRUZIONI

1. **Rinomina** la cartella chiamata "CognomeNomeMatricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali;
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 il tuo lavoro

NON SPEGNERE IL PC A FINE ESAME

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

Bisogna implementare un rudimentale sistema di voto elettronico che garantisca l'integrità dei risultati durante e dopo la fase di elezione. Il sistema è implementato attraverso una classe `Elezione`, che gestisce un elenco di candidati e alcuni metodi che devono essere thread-safe poichè eseguibili da remoto attraverso un numero non prefissato di diverse cabine elettorali. Ciascun candidato è modellato con un campo `nome` e un intero che registra il numero di voti acquisiti.

I metodi che una `Elezione` deve obbligatoriamente fornire sono:

`Elezione(candidati : List, elettori : int)`. Costruttore che predispone l'oggetto con un elenco candidati e specifica un certo numero di elettori aventi diritto.

`apriElezione()`. Consente di attivare le votazioni. Questo metodo può essere invocato una e una sola volta durante il ciclo di vita dell'oggetto.

`chiudiElezione()`. Interrompe le votazioni. Questo metodo può essere invocato una e una sola volta durante il ciclo di vita dell'oggetto e solo ad elezione aperta.

`vota(Candidato c)`. Acquisisce un voto per il candidato c. Interrompe immediatamente le elezioni, senza acquisire l'ultimo voto, se i voti acquisiti superano il numero degli elettori aventi diritto. Questo metodo non può essere invocato ad elezione chiusa.

`getVoti() -> int`. Restituisce quante persone hanno già votato. Questo metodo può essere invocato in qualsiasi momento (sia ad elezione appena creata, ad elezione aperta e ad elezione chiusa).

`waitForRisultati() -> List`. Attende che il processo di elezione sia concluso. Appena l'elezione risulta chiusa, restituisce i risultati ordinati per candidato avente il maggior numero di voti. Restituisce un array vuoto se il numero di votanti supera il numero di elettori aventi diritto.

Tutti i metodi richiesti per devono essere implementati garantendo la necessaria thread safety; non sono ammesse situazioni di deadlock; è opportuno migliorare l'accessibilità concorrente alle strutture dati ed evitare, se presenti, situazioni di starvation.

NON SPEGNERE IL PC A FINE ESAME

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 o estendendo tutte le strutture dati laddove si ritenga necessario, e risolvendo eventuali ambiguità.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI? 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.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X; oppure JAVA 7 o successivo

Il linguaggio da utilizzare per l'implementazione è Python 3, o in alternativa, Java. È consentito usare qualsiasi funzione di libreria di Python 3.X o di Java 7 e versioni successive.

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

Sebbene non saranno oggetto di valutazione, è obbligatorio scrivere un `main()` e implementare esplicitamente del codice di prova, che è comunque necessario per testare il proprio codice prima della consegna.

ESERCIZIO 2 (Linguaggi di scripting. Punti 0-10)

Si scriva uno script Perl dal nome psaux.pl in grado di estendere le funzionalità del comando shell ps con i parametri 'a', 'u' e 'x' impartiti contemporaneamente (vedi descrizione comando **`ps aux`** a pagina seguente). In particolare lo script dovrà essere eseguito con i seguenti parametri obbligatori: --kill=true|false e --name=ALL|process_name.

Sinossi del comando:

```
./psaux --kill=true|false --name=ALL|process_name
```

I due parametri obbligatori possono avere i seguenti valori:

- **`--kill`** : può essere impostato a `true` oppure a `false`
- **`--name`** : può essere impostato ad ALL oppure può contenere una qualsiasi stringa che identifichi il nome di un processo scelto dall'utente.

Lo script, a seconda delle opzioni settate in fase di esecuzione, eseguirà una delle seguenti operazioni:

1. Se **`--kill=true`** e **`--name=ALL`** lo script termina segnalando un errore generico.
2. Se **`--kill=true`** e **`--name=process_name`** lo script stampa su STDOUT tutti i processi e sottoprocessi relativi a process_name (la colonna command deve contenere la stringa process_name) e successivamente esegue il comando shell `kill` (vedi descrizione comando **`kill`** a pagina seguente) seguito dal numero di pid di tutti i sottoprocessi relativi a `process_name`.
3. Se **`--kill=false`** e **`--name=ALL`** lo script stampa su STDOUT il risultato dell'esecuzione del comando `ps aux`.
4. Se **`--kill=false`** e **`--name=process_name`** lo script stampa su STDOUT il consumo totale di CPU% e MEM% del processo scelto.

NOTA BENE: VEDERE ESEMPIO A PAGINA SEGUENTE

NON SPEGNERE IL PC A FINE ESAME

DESCRIZIONE COMANDI SHELL DA UTILIZZARE:

- **`ps aux:`** produce un elenco di informazioni sui processi in esecuzione nel sistema. L'output è così composto (le colonne evidenziate in grassetto sono quelle di nostro interesse per il corretto funzionamento del nostro script):

USER	**PID**	**%CPU**	**%MEM**	VSZ	RSS	TTY	STAT	START	TIME	**COMMAND**
frances+	**8958**	**4.2**	**1.3**	1198748	221600	?	SLl	21:08	0:57	**/opt/google/chrome/chrome**
frances+	**8968**	**0.0**	**0.0**	8680	900	?	S	21:08	0:00	**/opt/google/chrome-sandbox**
frances+	**8969**	**0.0**	**0.3**	426756	55828	?	S	21:08	0:00	**/google/chrome--type=zygote**

- **`kill:`** termina uno o più processi in esecuzione nel sistema. Ad esempio è possibile eseguire il comando kill nel seguente modo per terminare tutti i processi e sottoprocessi di google chrome sopra riportati:

```
kill 8958 8968 8969
```

ESEMPIO DI ESECUZIONE: