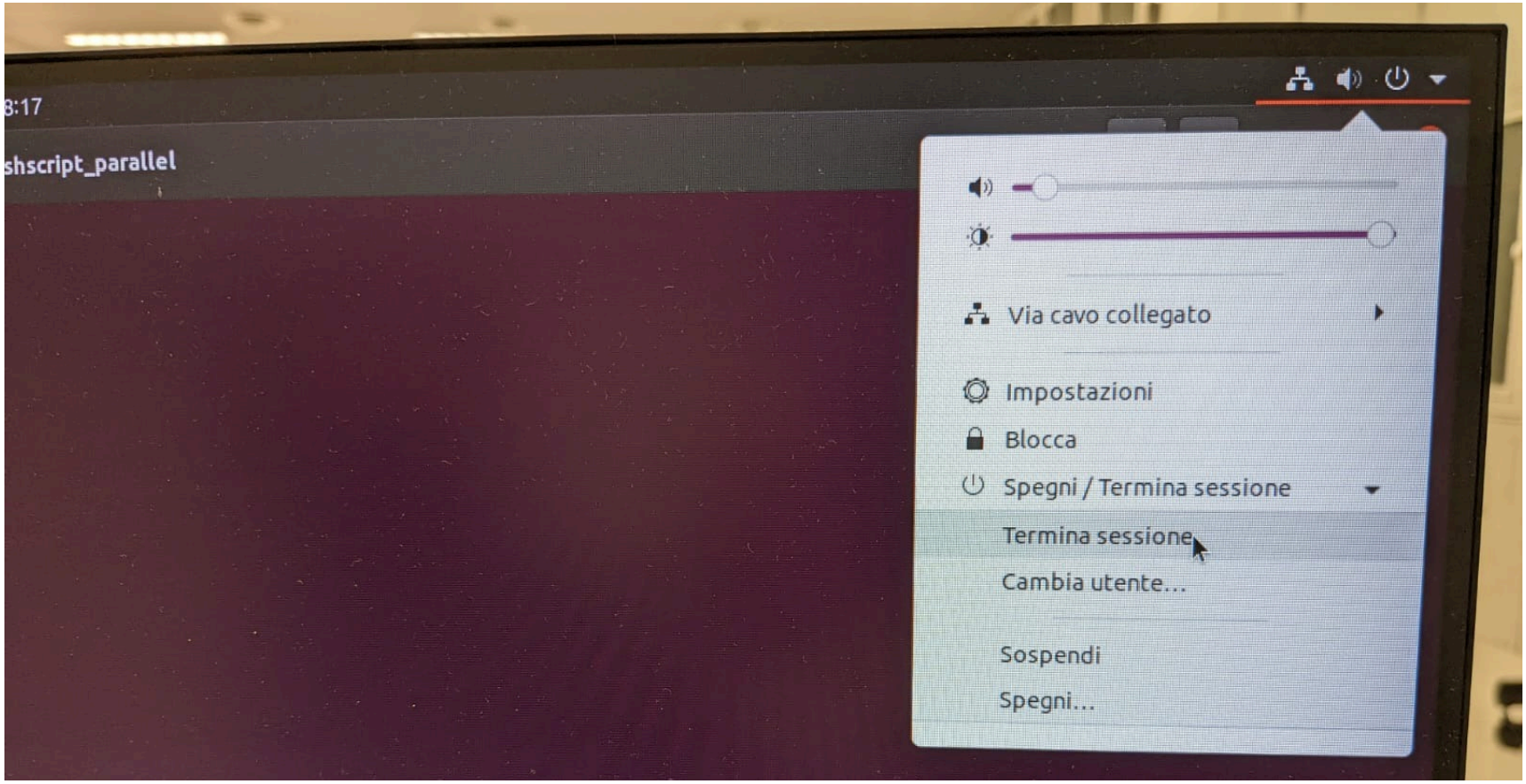
Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta - Febbraio 2024

ISTRUZIONI:

1. **Rinomina** subito 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 senza spazi**; 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 fai "**Termina Sessione/Logout**", ma lascia la postazione mantenendo il PC acceso.
5. **E' tua diretta responsabilità** garantire l'integrità del tuo elaborato, anche in caso di assenza di corrente. **Salva spesso il tuo lavoro**



e NON spegnere il PC.

SALVA SPESSO

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 pubblici di una classe se questi sono stati forniti. Potete aggiungere qualsivoglia campo e metodo privato, 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 l'interfaccia, il risultato finale o il significato.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X

Il linguaggio da utilizzare per l'implementazione è Python 3.6 o successivo. 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`

POSSO CONSENTIRE SITUAZIONI DI RACE CONDITION NEL MIO CODICE? NO
POSSO CONSENTIRE SITUAZIONI DI DEADLOCK NEL MIO CODICE? NO
POSSO CONSENTIRE ALTRE SITUAZIONI DI BLOCCO TOTALE NEL MIO CODICE, TIPO NESTED LOCKOUT, LIVELOCK O ALTRO? NO
POSSO CONSENTIRE SITUAZIONI DI STARVATION NEL MIO CODICE? SI, tranne quando ti viene chiesto esplicitamente di rimuoverle

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

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

ESERCIZIO 1 - PROGRAMMAZIONE MULTITHREADED

(Punteggio minimo richiesto 18/30. Pesa per ⅔ del voto finale)

Estendi la classe `BlockingQueue` aggiungendo una funzionalità di logging che registra le ultime M operazioni avvenute. Il sistema di logging deve essere attivabile (specificando il valore di M) e disattivabile a piacimento. Quando il logger è attivo bisogna registrare ogni operazione di inserimento e di estrazione, l'elemento che è stato estratto o inserito, e il tempo in cui ciò è avvenuto (usa la funzione `time.time()` per tale scopo). Deve essere inoltre possibile consultare i dati registrati con delle apposite funzioni.
I metodi che bisogna aggiungere a BlockingQueue sono:

`start_logging(self,M)`. Dal momento in cui questa operazione viene chiamata ogni operazione di put e get viene registrata per come prescritto, fino ad un massimo di M operazioni. Le operazioni più vecchie devono essere via via cancellate: in altre parole bisogna conservare traccia solo delle ultime M operazioni.

`stop_logging(self)`. Termina il logging e cancella tutto il diario delle operazioni precedentemente registrate.

`read_get_log(self,o) -> float`. Restituisce il tempo in cui l'oggetto `o` risulta essere stato prelevato per l'ultima volta dalla coda. Genera un'eccezione se il logger è disattivo, mentre restituisce 0 se nessun prelievo dell'oggetto risulta nel log.

`read_put_log(self,o) -> float`. Restituisce il tempo in cui l'oggetto `o` risulta essere stato inserito per l'ultima volta nella coda. Genera un'eccezione se il logger è disattivo, mentre restituisce 0 se nessun inserimento dell'oggetto risulta nel log.

`read_diff_log(self,o) -> float`. Restituisce la differenza temporale tra l'ultima volta in cui l'oggetto `o` è stato prelevato e quella in cui è stato inserito. Genera una eccezione se il logger è disattivo, mentre restituisce -1 se uno dei due tempi non risulta nel log.

Questo è tutto. Come sempre, è tuo specifico compito decidere cosa fare per tutti i dettagli che non sono stati espressamente definiti. Ricorda che tutti metodi richiesti sono da intendersi pubblici e devono essere tutti thread-safe.

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

(Punteggio minimo richiesto 18/30. Pesa per ⅓ del voto finale)

I file di log di sistema, `syslog*`, sono utilizzati per registrare eventi importanti, avvisi, errori e altre informazioni rilevanti durante il funzionamento del sistema. Questi file di log sono localizzati nella directory `/var/log/`. Questi file possono diventare molto grandi nel tempo, poiché continuano a registrare nuovi eventi durante il funzionamento del sistema. Per evitare che diventino eccessivamente grandi e per gestire meglio lo spazio su disco, è comune dividere periodicamente i file di log in archivi (i vecchi file sono normalmente chiamati `syslog.1.gz, syslog.2.gz` ecc). Questi archivi possono essere compressi per risparmiare spazio su disco e archiviati in modo da poter essere consultati in futuro per l'analisi o il debugging.

Scrivi uno script perl che analizzi tutti i file di log di sistema disponibili (archivi e non) e faccia quanto segue:

1. estragga i messaggi di log che non siano del mese corrente;
2. copi tutti i messaggi estratti in un nuovo file chiamato `my_log`;
3. comprima nell'archivio `my_log.gz` il file appena creato.

Potrebbero tornarti utili i comandi **date**, **zgrep** e **tar.**