

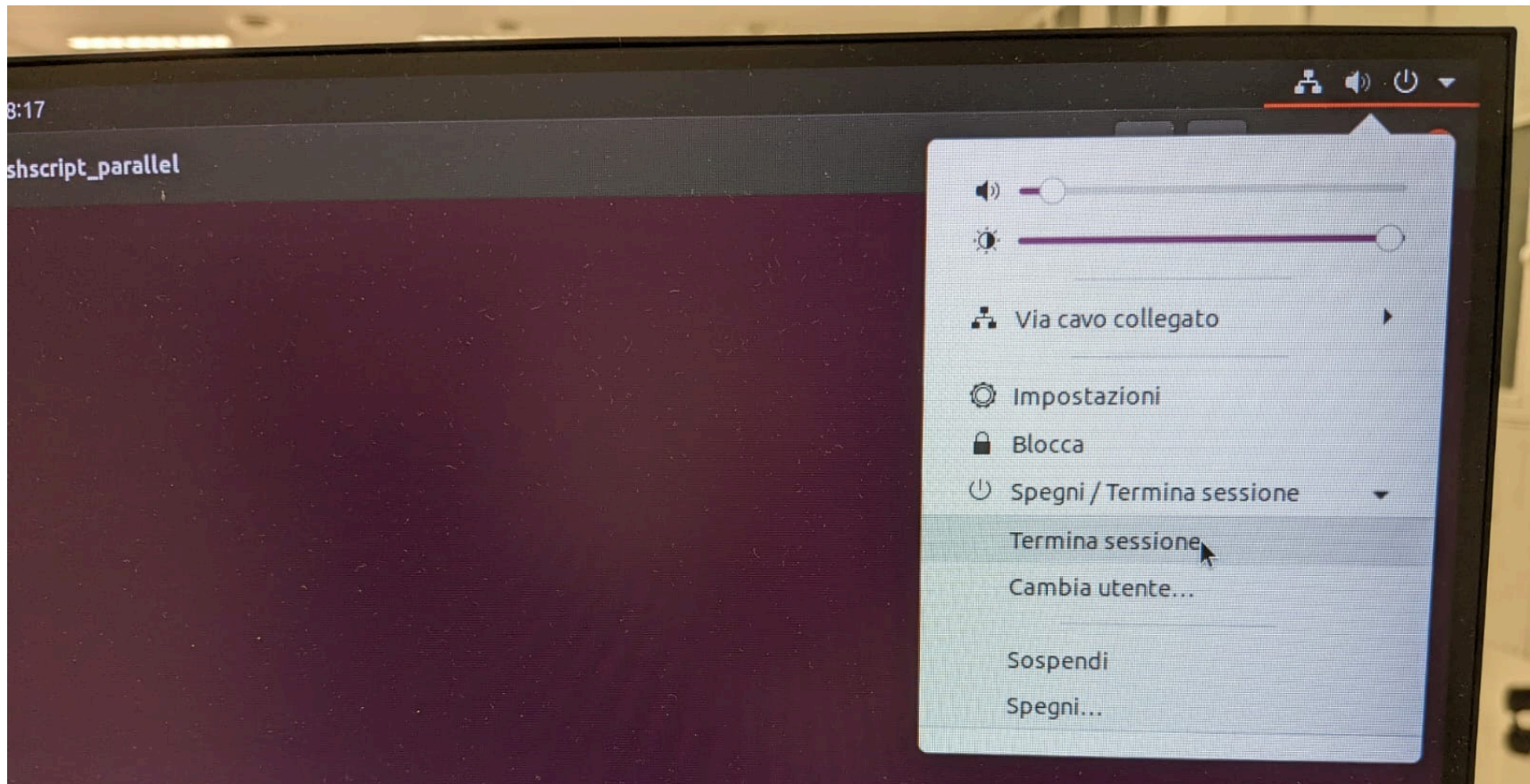
Corso di Sistemi Operativi

Corso estinto di Sistemi Operativi e Reti - Modulo Sistemi Operativi

Prova scritta - Novembre 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

Il codice che scriverai è pensato per essere usato come un modulo di libreria. Un modulo di libreria potrebbe essere usato da altri programmatori, i quali si aspettano di trovare una specifica interfaccia.

Non è quindi consentito modificare il prototipo dei metodi pubblici di una classe se questi sono stati forniti. Puoi 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 $\frac{2}{3}$ del voto finale)

Svolgi tutti i punti della traccia in un unico file che chiamerai SOLUZIONE.py

Punto 1 Aggiungi due nuovi ingredienti:

- Ingrediente 5: "Pane di sopra" (Simbolo: `/==\`)
- Ingrediente 6: "Pane di sotto" (Simbolo: `\==/`)

Assicurati che questi nuovi ingredienti vengano prodotti da dei rispettivi thread produttori. Inoltre, modifica il comportamento dell'assemblatore in modo che ogni ordine includa sempre automaticamente un "Pane di sopra" all'inizio e un "Pane di sotto" alla fine, anche se non sono specificati esplicitamente nell'ordine.

Punto 2 Modifica l'assemblatore in modo che stampi l'intero ordine completato in un'unica volta, invece di stampare un ingrediente alla volta. Assicurati inoltre che "Pane di sopra" sia sempre il primo elemento e "Pane di sotto" l'ultimo. Ad esempio, se un ordine include gli ingredienti `[0, 1, 2, 3, 4]`, l'output dovrà essere:

```
/==\  
@@@@  
====  
####  
----  
oooo  
\\==/
```

Tutta la sequenza deve essere stampata insieme, senza interruzioni tra gli ingredienti e senza che altri thread abbiano la possibilità di interferire inframezzando le proprie stampe durante la stampa completa di un ordine.

Punto 3 Crea un nuovo tipo di thread chiamato `InviatoreOrdiniAvanzato`. Questo thread invia ordini come l'`InviatoreOrdini` già presente, ma poi si mette in attesa bloccante fino a quando l'ordine appena inviato non viene completamente assemblato e stampato a video. Modifica il codice in modo appropriato per supportare questa funzionalità.

Punto 4 Limita la produzione degli ingredienti: nell'arco del suo ciclo di vita, ogni thread produttore deve generare un massimo di 20 unità totali del proprio ingrediente e poi terminare. Se un certo ingrediente si esaurisce, il thread assemblatore deve rilevare questa situazione e creare un ordine parziale utilizzando solo gli ingredienti ancora disponibili.

Punto 5 Fai in modo che tutti i thread — compresi i produttori, gli assembleri e gli inviati di ordini — terminino in modo ordinato una volta che tutti gli ordini sono stati inviati, completamente assemblati e stampati a video. Assicurati che il programma intero termini senza lasciare thread attivi.

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

(Punteggio minimo richiesto 18/30. Pesa per $\frac{1}{3}$ del voto finale)

Scrivi uno script Perl chiamato `empty.pl` che riceva il path ad una cartella come parametro ed esegua le seguenti operazioni:

- Cerca tutti i file vuoti (dimensione 0 byte) presenti all'interno della cartella, comprese le sottodirectory.
- Rinomina tutti i file vuoti trovati concatenando al nome originale la stringa `_vuoto` e stampa il numero totale di file rinominati.
- Se non viene specificata alcuna directory, utilizza la directory corrente come predefinita.

Esempio:

Lo script potrebbe essere invocato come: `./empty.pl ../Download`

e, se in `../Download` ci fossero 5 file vuoti (dimensione 0 byte), stamperebbe

Rinominati 5 file vuoti nella directory `../Download`