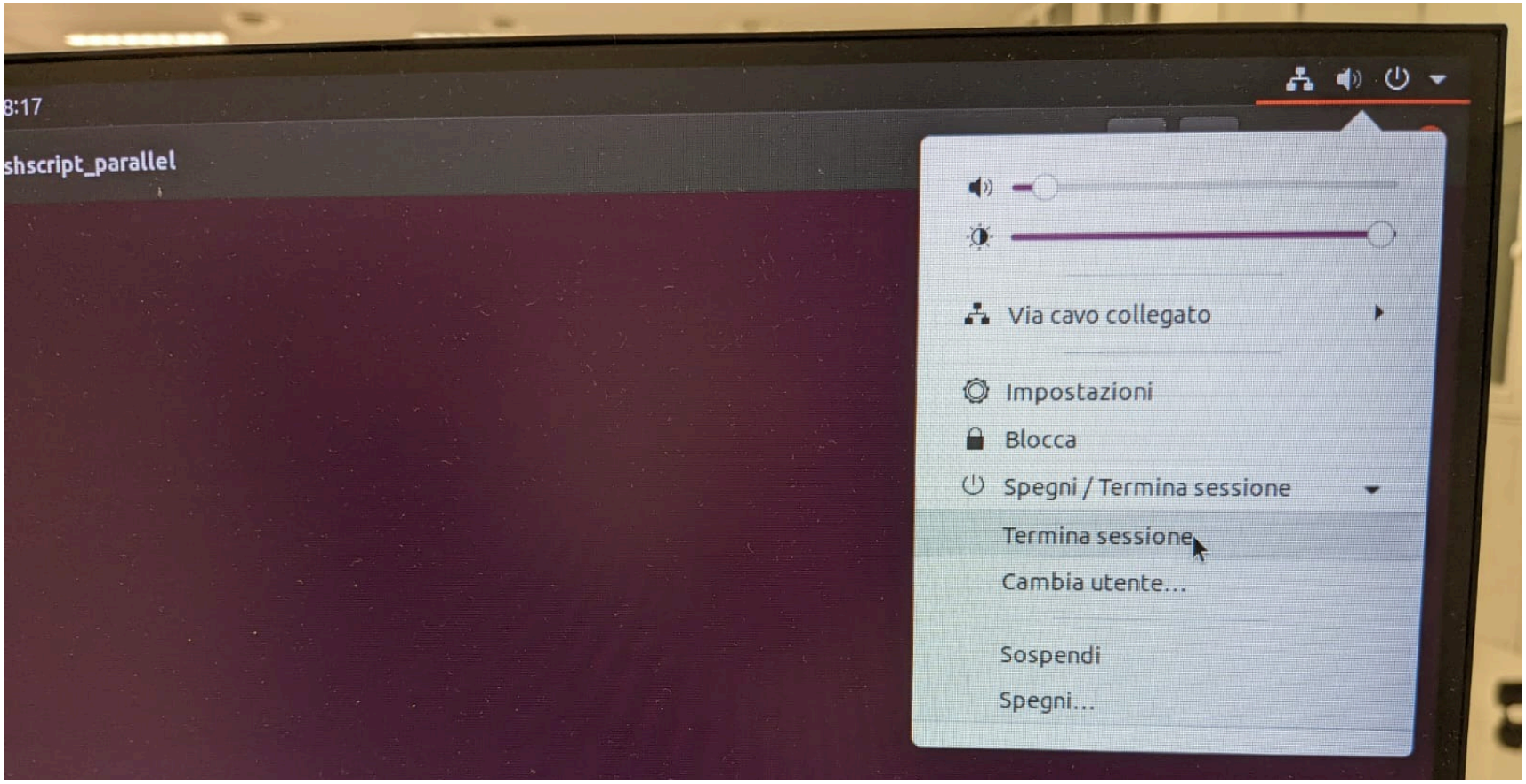
Corso di Sistemi Operativi

Corso estinto di Sistemi Operativi e Reti

Prova scritta - Luglio 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)

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

Punto 1

Aggiungi la possibilità che ogni giocatore sia visualizzato sulla griglia di gioco con un proprio simbolo distinto, anzichè "`G`". Devono essere consentite le lettere negli intervalli "`A..Z`" e "`a..z`".

Punto 2

Aggiungi un parametro `priorita` (di default, `priorita=False`), alla funzione `dprint`. Le stampe inviate con `priorita=True` devono essere stampate prima di tutte le altre. Prova a cambiare le scritte prioritarie in rosso stampando a video il codice speciale '`\x1b[31m`'. Puoi usare la stringa '`\x1b[30m`' per tornare al nero. Potresti ad esempio usare le stampe prioritarie per notificare alcuni eventi speciali, come quelli previsti nei successivi punti 3, 4, 5 e 6.

Punto 3

Fai in modo che una istanza di `GridGame` possa essere configurata con una durata massima del gioco `M`, espressa in secondi, al termine della quale tutti i thread devono terminare, infine terminando anche il programma principale.

Punto 4

Implementa la possibilità che un `Giocatore` possa "eliminare" un altro giocatore. Il giocatore A elimina un giocatore B, se a seguito di una mossa di A, le coordinate di A coincidono con quelle di B. Il thread associato a B deve in questo caso terminare, e il simbolo associato a B non deve più comparire durante le stampe. Non appena A si sposta, nel punto dove B è "morto" deve comparire il simbolo "`*`". Deve essere vietato muoversi in punti della griglia dove compaiono "`*`". Decidi tu come si deve comportare il codice se un Giocatore prova a spostarsi su un "`*`".

Punto 5

Fai in modo che il gioco, oltre al termine entro il tempo `M` di cui al Punto 2, possa terminare quando resta un solo giocatore "vivo".

Punto 6

Implementa il restringimento periodico della mappa. A intervalli regolari, un thread apposito riduce la griglia eliminando la prima e ultima riga, e la prima e ultima colonna. Decidi tu che regole applicare per i giocatori che dovessero in quel momento trovarsi in una cella che sta per essere eliminata. Per la scelta dell'intervallo tra un restringimento e l'altro, tieni conto di `M` e della variabile `dimensione_griglia`.

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

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

Scrivi uno script Perl dal nome `merge_dirs.pl` che riceve come parametri sulla linea di comando i percorsi di due directory. Lo script deve svolgere le seguenti operazioni:

1. Unire il contenuto delle due directory in una terza directory chiamata `merged`, che deve essere creata nella directory corrente.
2. Durante la copia dei file, evitare duplicati: se due file con lo stesso nome esistono in entrambe le directory, mantenere solo il file più recente (in termini di data di modifica).
3. Dopo aver creato la directory `merged` con i file uniti, lo script deve creare un file di log chiamato `merge.log` nella directory corrente. Questo file di log deve contenere:
    a. Il nome di ciascun file copiato nella directory merged.
    b. Il percorso originale del file (prima della copia).
    c. Un'indicazione se il file è stato sovrascritto perché duplicato.