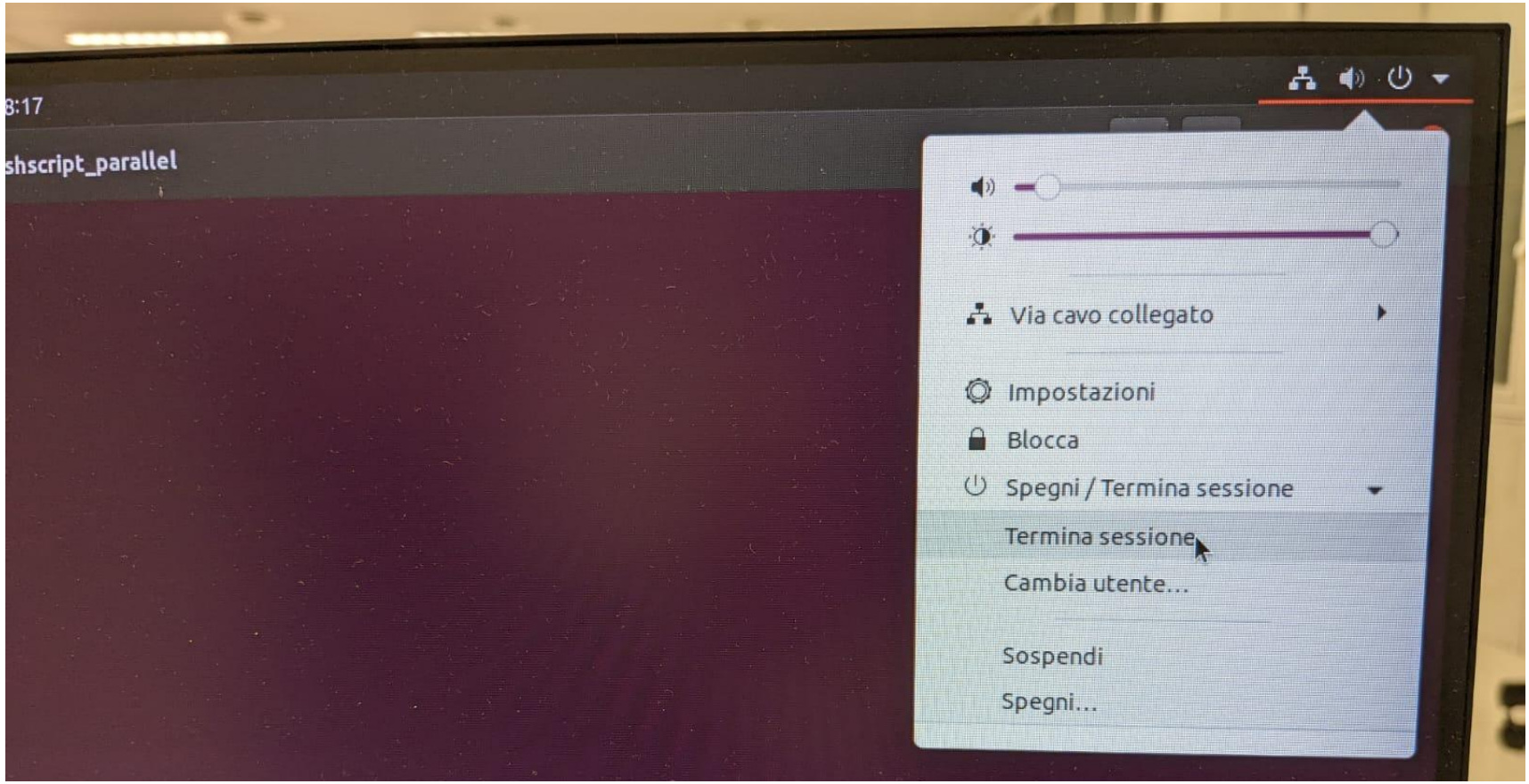
Corso di Sistemi Operativi

Corso estinto di Sistemi Operativi e Reti - Modulo Sistemi Operativi

Prova scritta - Giugno 2025

LEGGI ATTENTAMENTE LE 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? DECIDI TU COSA FARE
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

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 `SOLUZIONE.py`

-> Qualsiasi altro nome di file verrà IGNORATO <-

Punto 1.
Personalizza la libreria di gioco facendo in modo che per ogni partita si possa personalizzare il range di numeri ammissibili e il numero di giocate che deve fare ogni giocatore, che potrà dunque essere anche maggiore di 1.

Punto 2.
Introduci la possibilità per ciascun giocatore di "sbirciare", ma solo una e una sola volta, quanti numeri hanno già registrato almeno `n` giocate. Tale funzionalità deve essere realizzata con il metodo `sbircia(n)` che restituisce un intero corrispondente a quanti numeri hanno già registrato `n` giocate. Se uno stesso thread chiama questo metodo più di una volta nell'arco di una partita, viene squalificato. Non è necessario che tu garantisca che le giocate non cambino tra una invocazione di `sbircia(n)` e una invocazione di `puntaNumero`. Modifica la strategia di gioco dei giocatori per sfruttare la possibilità di "sbirciare".

Punto 3.
Introduci il metodo `attendi_e_gioca(n)`. Tale metodo va in attesa bloccante fintantoché qualcuno non gioca il numero `n+1`, dopodichè gioca il numero `n`.

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

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

NOTA BENE: In questo esercizio dovrai sistemare errori nello script che trovi nella cartella della traccia e non implementarne uno da zero. Leggi la traccia fino in fondo prima di procedere.

Lo script `controlla_file.pl` dovrebbe ricevere come argomenti di input il `path` ad una cartella del sistema, seguito da un parametro che può essere `-g` oppure `-u` e infine da una stringa `S`. Lo script dovrebbe ricercare all'interno della directory specificata tutti e solo i file che contengono nel loro nome la sottostringa `S`. Lo script dovrebbe poi stampare su un FILE dal nome `results.out`:

Se presente l'opzione `-u`, la size totale dei file trovati per ogni utente;
Se presente l'opzione `-g`, la size totale dei file trovati per ogni gruppo;

Esempio:$

Supponi di invocare lo script come

```
perl controlla-file.pl . -u report
```

dove il contenuto della cartella corrente è:

```
-rw-r--r-- 1     mario        staff        120      Apr 10 report1.txt
-rw-r--r-- 1     luigi        admin        300      Apr 11 report2.pdf
```

```
-rw-r--r-- 1     mario        staff        1200     Apr 10 file1.txt
-rw-r--r-- 1     luigi        admin        130      Apr 11 mio_report2.pdf
-rw-r--r-- 1     luigi        admin        375      Apr 11 file2.pdf
-rw-r--r-- 1     elena        admin        4023     Apr 11 my_report.pdf
-rw-r--r-- 1     anna         staff        100      Apr 10 file2.txt
```

il file `results.out` dovrebbe contenere

```
luigi: 430
elena: 4023
mario: 120
```

Se invece fosse invocato come

```
perl controlla-file.pl . -g report
```

allora il file `results.out` dovrebbe contenere

```
admin: 4453
staff: 120
```

Lo script allegato, però, presenta errori di sintassi e/o logici. Sistemalo per ottenere il comportamento sopra descritto.
NON devi stravolgerne il contenuto: la struttura dello script deve rimanere pressoché invariata.