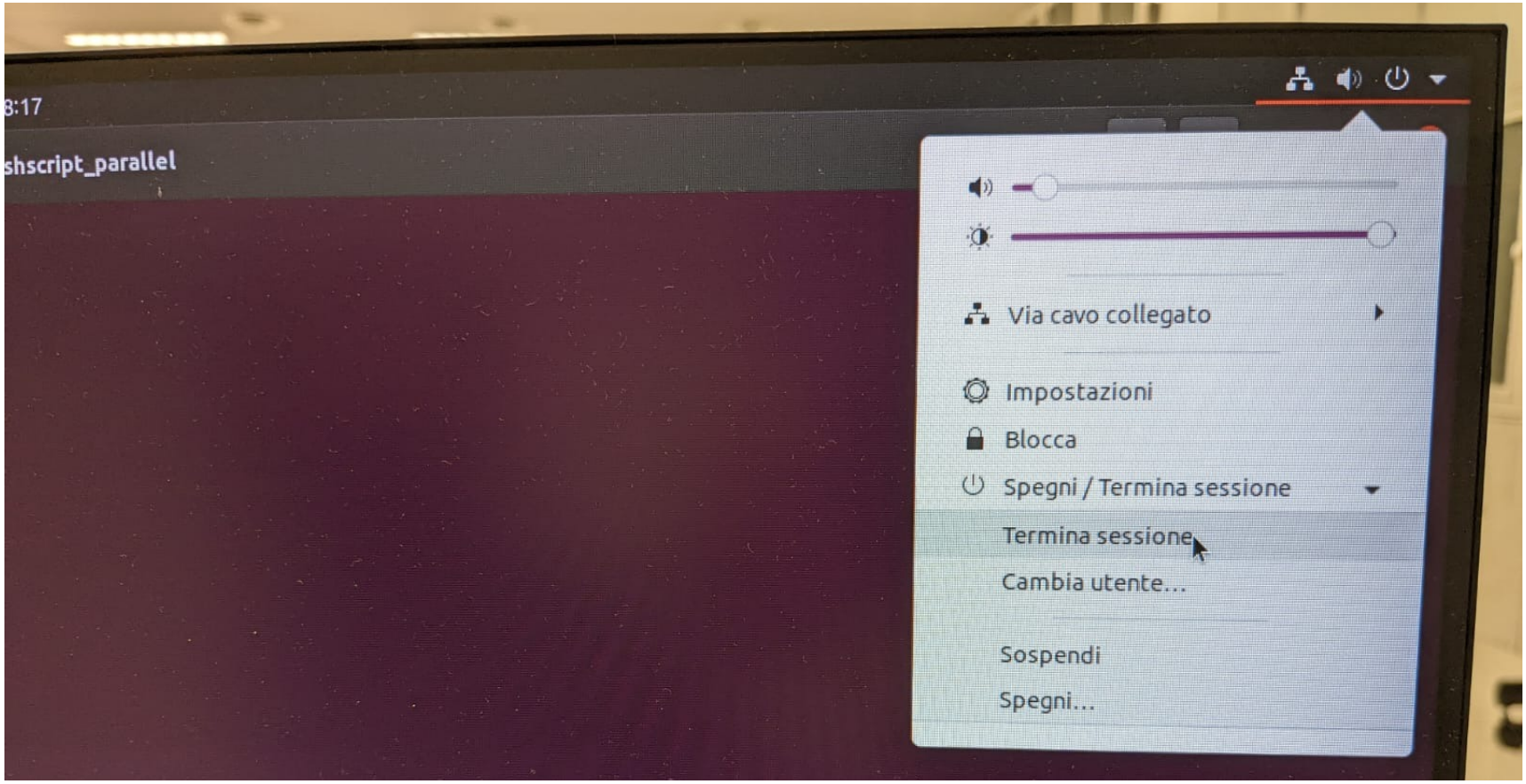
Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta - 16 Febbraio 2023

ISTRUZIONI PER CHI È IN PRESENZA:

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**”, lascia la postazione



e NON spegnere il PC.

SALVA SPESSO

ISTRUZIONI PER CHI SI TROVA ONLINE:

1. **Questo file contiene il testo che ti è stato dato ieri, incluso il codice;**
2. **Mantieni a tutto schermo** questo file per tutta la durata della prova; puoi scorrere liberamente tra le sue pagine, ma non puoi cambiare applicazione;
3. **Firma** preliminarmente il foglio che userai per la consegna con nome cognome e matricola;
4. **Svolgi** il compito; puoi usare solo carta, penna e il tuo cervello;
5. **Aiutati** con i numeri di linea per indicare le eventuali modifiche che vorresti fare al codice che ti è stato dato.
6. **Alla scadenza** termina *immediatamente* di scrivere, e attendi di essere chiamato, pena l'esclusione dalla prova;
7. **Quando è il tuo turno** mostra il foglio ben visibile in webcam, e poi metti una foto dello stesso foglio in una chat privata Microsoft Teams con il prof.

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 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. Analogamente, i metodi esistenti possono essere modificati nel loro codice, ma non se ne deve cambiare 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 del 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)

Punto 1.

Come primo punto, modifica ogni `VasettoDiMiele` in maniera da ottimizzare l'uso delle condition e non risvegliare inutilmente più thread del necessario. In particolare, rimuovi la condition esistente e introduci una `condition_aumento`, da usare quando si aspetta che il vasetto aumenti il suo contenuto e una `condition_diminuzione`, da usare quando si aspetta che il vasetto diminuisca il suo contenuto. Puoi anche ottimizzare il codice a tua scelta introducendo più di due condition, se lo ritieni opportuno.

Punto 2

Avrai notato che è molto difficile che le mamme orse riescano a rabboccare un vasetto di miele, poiché è necessario che un vasetto di miele sia completamente vuoto. Per risolvere il problema, fai in modo che quando almeno una mamma orsa è bloccata all'interno del metodo `riempi`, nessun papà orso possa *aggiungere* del miele. Inoltre, quando una mamma orsa invoca il metodo `riempi`, ma il vaso è già pieno fino all'orlo, bisogna uscire immediatamente anziché aspettare.

Nota. Non preoccuparti dell'eventuale condizione di deadlock che potrebbe verificarsi

Punto 3

Implementa una funzione `totaleMiele()` che stampa a video la quantità di miele attualmente presente in tutti i vasetti, e la quantità che è stata finora mangiata da qualcuno. Bonus: osserva che potrebbero esserci delle quantità di miele temporaneamente fuori dai vasetti, poiché nelle mani di un papà orso nell'intervallo di tempo tra `prendi()` e `aggiungi()`. Puoi fare in modo che `totaleMiele()` tenga conto anche di queste quantità?

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

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

Il file **`.bash_history`** presente nella home utente conserva lo storico dei comandi usati nella shell dall'utente corrispondente (nota che si tratta di un file nascosto, non compare per tanto con `ls`, ma può essere normalmente aperto indicandone il nome). Scrivi uno script `history_stats.pl` che si occupi di ricavare alcune statistiche da questo file. In particolare, lo script deve poter essere invocato nel seguente modo:

```
./history_stats.pl [-n VALUE]
```

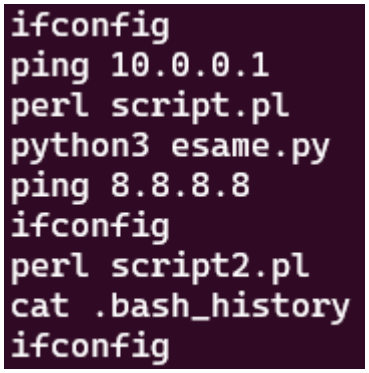
dove `-n` è opzionale ma, se specificato, richiede obbligatoriamente di essere seguito da un valore intero (`VALUE`).

Quando il parametro `-n` **non** è presente, lo script dovrà stampare su `STDOUT` prima il comando shell maggiormente usato e poi il comando usato meno volte dall'utente corrente. Accanto al nome del comando, va stampato il numero di volte in cui è stato utilizzato. A parità di utilizzo, la stampa dovrà essere effettuata in ordine alfabetico inverso. **Per semplicità**, si intende per "comando shell" la prima stringa che compare in ogni linea della history che sia priva di spazi.

Quando viene invece specificato il parametro `-n`, bisogna tenere conto del valore `VALUE` stampando sul file `last_calls` gli ultimi `VALUE` comandi presenti nel file.

Esempio:

Supponiamo che il contenuto del file **`.bash_history`** sia il seguente



- Se lo script fosse invocato come

  `history_stats.pl`

  Lo script dovrebbe stampare

  ```
  ifconfig 3
  python3 1
  cat 1
  ```

- Se invece lo script fosse invocato come

  `history_stats.pl -n 3`

  Lo script dovrebbe stampare sul file `last_calls`

  ```
  perl script2.pl
  cat .bash_history
  ifconfig
  ```