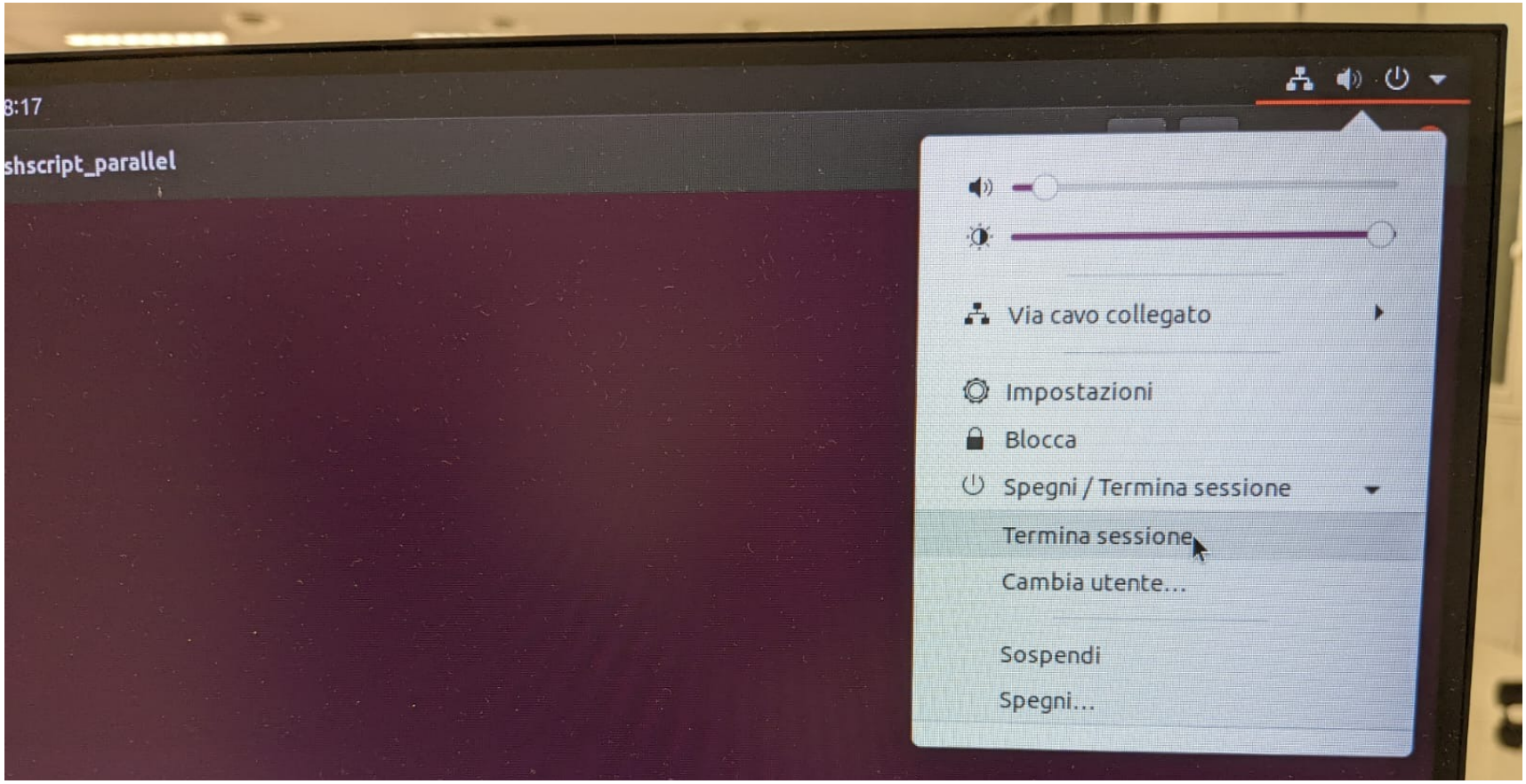
Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta - 5 Dicembre 2022

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

Punto 1

Aggiungi alla struttura dati `BlockingStack` il metodo `flush(self)`. Tale metodo elimina tutti gli elementi attualmente presenti nel `BlockingStack`.

Punto 2

Aggiungi alla struttura dati `BlockingStack` il metodo `putN(self,L : List)`. Tale metodo inserisce tutti gli elementi della lista `L` all'interno di `self`. Se `self` non dispone di almeno `len(L)` posti liberi, ci si pone in attesa bloccante finché tali posti non si rendano disponibili, effettuando a seguire l'inserimento degli elementi di `L`.

Punto 3

Dal momento che un `BlockingStack` è basato su una politica di inserimento ed estrazione di tipo LIFO, è evidente che esso soffre di problemi di starvation. Introduci dunque il metodo `setFIFO(self,onOff : bool)`. Quando `onOff = True`, il `BlockingStack` corrente deve cominciare a funzionare come una `BlockingQueue`, e cioè con politica di inserimento ed estrazione FIFO. Se invece `onOff = False`, il BlockingStack deve tornare a funzionare come uno stack LIFO. L'invocazione di `setFIFO` deve avere effetto anche sull'ordine di estrazione degli elementi ormai già inseriti.

SALVA SPESSO

ESERCIZIO 2 - LINGUAGGI DI SCRIPTING

La cartella `/usr/include` contiene gli header file utilizzati dai compilatori `C`.
Si scriva uno script Perl dal nome **`comments.pl`** che analizzi i file contenuti nella suddetta cartella e stampi alcune informazioni sui commenti al codice in essi contenuti. Lo script dovrà essere invocato nel seguente modo:

```
./comments.pl OPTIONS
```

dove il parametro OPTIONS è opzionale.

Quando lo script viene invocato senza parametri, dovrà stampare in STDOUT l'elenco dei file `.h` contenuti nella cartella `/usr/include` in ordine lessicografico.

- **Esempio:**
  Quando lo script sarà lanciato nel seguente modo:

  ```
  ./comments.pl
  ```

  si dovrà stampare su STDOUT un elenco di questo tipo:

  ```
  aio.h
  aliases.h
  …
  zconf.h
  zlib.h
  ```

In alternativa, lo script può essere invocato valorizzando `OPTIONS` con il nome di un file contenuto nella cartella `/usr/include`. In tal caso, lo script stamperà su STDOUT i commenti contenuti nel file (testo racchiuso tra "/*" e "*/", eventualmente distribuito su più righe), ordinando la stampa in base alla lunghezza dei commenti, dal più breve al più lungo. A parità di lunghezza, si deve seguire l'ordine lessicografico.

- **Esempio:**
  Supponendo che il file `/usr/include/threads.h` contenga SOLO il seguente testo

```
/* Exit and error codes.  */
enum
{
  thrd_success  = 0,
  thrd_busy     = 1,
  thrd_error    = 2,
  thrd_nomem    = 3,
  thrd_timedout = 4
};

/* Mutex types.  */
enum
{
  mtx_plain     = 0,
  mtx_recursive = 1,
  mtx_timed     = 2
};

typedef struct
{
  int __data __ONCE_ALIGNMENT;
} once_flag;
#define ONCE_FLAG_INIT { 0 }
```

Quando lo script sarà lanciato nel seguente modo:

```
./comments.pl threads.h
```

si dovrà stampare su STDOUT:

```
Mutex types.
Exit and error codes.
```

Non è necessario tenere conto dei commenti innestati o dei commenti su singola linea marcati con “//” .