TESINA DI PROGETTAZIONE DEI SISTEMI OPERATIVI
(Laurea Magistrale in Ingegneria Informatica)
A.A. 2022-2023

(ver. del 4/12/2022)

Sommario

1. POSSIBILI ARGOMENTI PER LA TESINA

Si ricorda che la tesina (come illustrato alla URL http://www.didattica.agentgroup.unimo.it/wiki/index.php/Progettazione_di_Sistemi_Operativi_-_LM#Tesina) è *facoltativa*; nel caso di decida di svolgerla, va concordato l'argomento e quindi va presentata e discussa prima di partecipare ad una prova orale.

ATTENZIONE: qualunque sia la scelta effettuata e concordata, non può essere scelto come argomento qualcosa presentato per altri esami o altre ragioni, anche in altre sedi Universitarie.

1.1 Prettamente pratici

Nel seguito si trovano una serie di URL che possono essere consultate per scegliere un argomento per la tesina:

1) http://www.cs.kent.edu/~ruttan/sysprog/lectures/multi-thread/multi-thread.html
   ➔ sono proposti 3 esempi (il secondo è una versione ampliata del primo)

2) http://www.ce.uniroma2.it/courses/sdcc1617/lucidi/Pthreads.pdf
   ➔ dalla slide 79 vengono presentate 3 versioni di un server TCP multi-threaded, il cui codice completo dovrebbe essere qui http://www.ce.uniroma2.it/courses/sdcc1617/#prog_pthread
   Poiché il sito di cui sopra sta avendo dei problemi in questo periodo, la collega Prof.ssa Valeria Cardellini mi ha gentilmente fornito i file che potete trovare qui
   https://www.didattica.agentgroup.unimo.it/didattica/ProgettazioneSO/Lucidi/Laboratorio/ValeriaCardellini/

3) https://www.cdac.in/index.aspx?id=ev_hpc_hypack13_mode01_multicore_pthreads
   ➔ sembra ci siano 7 esempi di programmi multi-threaded

4) https://www.cs.swarthmore.edu/~newhall/cs31/f19/Labs/lab09/
   ➔ richiede di parallelizzare una applicazione che fa uso di memoria condivisa (l'applicazione nella versione sequenziale era richiesta nel seguente laboratorio
   https://www.cs.swarthmore.edu/~newhall/cs31/f19/Labs/lab06/)

5) https://w3.cs.jmu.edu/lam2mo/cs470_2022_01/p1_pthread.html

➔ elenca le specifiche di una Pthread application che deve implementare una versione parallela del programma sequenziale sum.c (fornito)

6. http://www.didattica.agentgroup.unimo.it/wiki/index.php/EserciziPSO
   ➔ elenca 26 esercizi svolti in Java che possono essere riscritti con i Pthread: si considerino almeno 3/4/5 esercizi per ogni tesina in dipendenza della difficoltà degli stessi (quindi 3 esercizi se molto complessi, 4 se di difficoltà media, 5 se esercizi facili)

7. http://didattica.agentgroup.unimo.it/didattica/psoNOD/
   elenca almeno un 40-ina di esercizi (tralasciando quelli eventualmente già visti ad esercitazione) svolti in pseudo-linguaggio che possono essere riscritti con i Pthread. Anche in questo caso, si considerino almeno 3/4/5 esercizi per ogni tesina in dipendenza della difficoltà degli stessi (quindi 3 esercizi se molto complessi, 4 se di difficoltà media, 5 se esercizi facili)

8. http://didattica.agentgroup.unimo.it/didattica/ProgettazioneSO/Lucidi/Laboratorio/LittleBookOfSemaphores(Pag.127-250).pdf*
   elenca 15 esercizi (tralasciando quelli eventualmente già visti ad esercitazione) svolti in pseudo-linguaggio che possono essere riscritti con i Pthread. Anche in questo caso, si considerino almeno 3/4/5 esercizi per ogni tesina in dipendenza della difficoltà degli stessi (quindi 3 esercizi se molto complessi, 4 se di difficoltà media, 5 se esercizi facili)

9. http://www.0x04.net/doc/posix/Multi-Threaded%20Programming%20with%20POSIX%20Threads%20-%20Linux%20Systems%20Programming.pdf#%5B%7B%22num%22%3A132%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C0%2C792%2Cnull%5D
   7 Idee per piccoli progetti: 1) simple configuration file parser; 2) Web server “Visitor Tracking” Log Analyzer; 3) Log file rotator; 4) Implementing Hash Tables; 5) Writing a CT-100-based PacMan Game; 6) Implementing a simple-File-Transfert Client and Server; 7) Tiny multi-threaded http server

10. https://users.cs.cf.ac.uk/dave/C/node32.html#SECTION00327000000000000000
    2 idee per progetti: 1) Threaded version of UNIX grep; 2) multithreded quicksort

Uno studente può anche proporre un argomento relativo alla programmazione concorrente che non sia fra quelli sopra elencati.

Nel caso di studenti che non si trovino a loro agio con la programmazione concorrente di UNIX in ambiente globale usando Pthread, mutex, semafori o mutex e variabili condizioni, possono essere considerate anche proposte di argomenti per tesine svolte con gli strumenti della programmazione concorrente di UNIX in ambiente locale usando processi pesanti, pipe, segnali o socket.

1.2 Più teorici

Nel caso di studenti che non si trovino a loro agio con la programmazione concorrente **in generale**, possono essere considerate anche tesine il cui scopo sia:
- approfondire uno degli argomenti trattati a lezione tramite articoli di survey e/o di ricerca;
- esplorare argomenti non trattati a lezione, relativi in generale ai Sistemi Operativi o in specifico alla programmazione concorrente.

Nell’ambito di quest’ultima possibilità si riportano alcuni argomenti non trattati a lezione che potrebbe essere interessante esplorare:

A) il linguaggio RAKU (cioè Perl 6) che presenta delle caratteristiche di concorrenza; a riguardo, si veda la seguente documentazione:
   - https://raku.guide/it/ che contiene una descrizione generale;

* Questo pdf contiene una parte del ‘Little book of semaphores’ il cui contenuto completo può essere scaricato dalla URL: http://greenteapress.com/semaphores/LittleBookOfSemaphores.pdf

- https://docs.raku.org/language/concurrency che contiene una descrizione specifica sugli aspetti di concorrenza. In questo caso, la tesina potrebbe essere solo teorica o comprendere anche parti pratiche di programmazione concorrente in RAKU.

B) le librerie (threading e multiprocessing) del linguaggio Python per la gestione della concorrenza; a riguardo, si veda la seguente documentazione:
https://docs.python.org/3/library/concurrency.html.
Anche in questo caso, la tesina potrebbe essere solo teorica o comprendere anche parti pratiche di programmazione concorrente in Python.

C) Slurm, un sistema open source, fault-tolerant e altamente scalabile di gestione job scheduling per cluster Linux; si veda
https://slurm.schedmd.com/documentation.html

2. ASSEGNAZIONE DELLA TESINA

L'assegnazione di un argomento per una tesina deve avvenire tramite l'invio di una mail a silvia.cascianelli@unimore.it indicando l'argomento scelto e attendendo la risposta sull'assegnazione, che avverrà solo dopo avere verificato che tale argomento non sia già stato assegnato.
Solo in casi specifici (argomento particolarmente complesso), la tesina può essere svolta in gruppo: si considera ragionevole svolgerla al massimo in un gruppo di due studenti.
IMPORTANTE: L'assegnazione dell'argomento della tesina deve avvenire *come minimo* almeno una settimana rispetto alla data di consegna della tesina stessa!

3. REDAZIONE DELLA TESINA

La tesina va redatta con una struttura che si ispira ad un elaborato di Laurea Triennale o ad una tesi di Laurea Magistrale quindi con: una Introduzione, almeno un capitolo che illustra il problema oggetto della tesina, una Conclusione e una bibliografia/riferimenti bibliografici.
Se l'argomento riguarda la programmazione concorrente, il codice deve essere, di norma, eseguibile su un sistema *Linux* e deve essere adeguatamente commentato e ragionevolmente parametrizzato (si veda il seguente paragrafo per il formato di consegna del codice). Si segnala di NON riportare nella tesina delle videate di codice su sfondo nero, sia perché la stampa della tesina comporta uno spreco di inchiostro/toner e sia perché la leggibilità ne risente; se necessario, è accettabile inserire alcune parti di codice considerate particolarmente significative o videate che riportano esempi di esecuzione, che comunque possono essere riportate come semplici copia-incolla (e quindi non su sfondo nero). Particolarmente importante è spiegare come vanno invocati i singoli programmi, spiegando significato e ordine degli eventuali parametri.
Nel caso di tesina di gruppo, le tesine consegnate da ognuno dei due partecipanti al gruppo devono essere comunque distinte e illustrare chiaramente il contributo del singolo studente e la relazione con il lavoro svolto dall'altro partecipante al gruppo.
N.B. Si ricorda che è fatto divieto utilizzare il logo dell'Ateneo!

4. FORMATO DI CONSEGNA DELLA TESINA

La tesina va consegnata *esclusivamente* mandando un mail a silvia.cascianelli@unimore.it avente come OGGETTO: TESINA-COGNOME-NOME e con allegato la tesina in **formato pdf** e denominata *cognome_nome_matricola.pdf*; se l'argomento riguarda la programmazione concorrente, va consegnato, inoltre, in un file compresso denominato *cognome_nome_matricola.zip*, il codice sorgente (eventualmente costituito da più file .c) unitamente al **makefile** per ottenere il codice eseguibile su qualunque sistema *Linux* (fare attenzione ad inserire eventuali opzioni per l'uso di eventuali standard del linguaggio C).

5. TEMPI DI CONSEGNA DELLA TESINA

La tesina va consegnata *inderogabilmente* **almeno due settimane** prima della data della prova orale.

6. PRESENTAZIONE E DISCUSSIONE DELLA TESINA

La tesina va presentata e discussa prima di poter accedere ad una prova orale; dopo la consegna verrà analizzata velocemente e, in caso di rispetto delle specifiche e di funzionamento del codice (nel caso sia previsto), verrà fissato

un appuntamento per la presentazione e discussione (che non ha bisogno di testimoni e potrebbe anche essere svolto in remoto); in caso di tesina di gruppo, i singoli studenti dovranno presentare il proprio lavoro individualmente e non necessariamente nello stesso giorno/ora. Nel caso di non rispetto delle specifiche e/o di non funzionamento, verrà richiesta la ripresentazione della tesina che può essere effettuata al massimo una seconda volta e che potrebbe comportare uno slittamento della data dell'orale.

7. VALUTAZIONE DELLA TESINA

Il voto assegnato allo studente per la tesina, al termine della discussione della stessa, influirà sul voto finale andando ad incrementare il voto della prova orale di un valore da **1 a 4 punti**.
Nella valutazione verranno considerati vari aspetti, come:
- scelta dell'argomento della tesina (semplice, medio, complesso);
- qualità del codice prodotto, se previsto;
- qualità della stesura della tesina;
- qualità della presentazione e discussione della tesina.