Sistemi di Calcolo 2 (A.A. 2019-2020 e successivi)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello straordinario - 17 Settembre 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - System and Network Security (rispondere nel file teoria1.txt)

Discutere delle varie policy per il controllo degli accessi a un sistema informatico

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := 1+ max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j] do skip
7           while num[j] != 0 AND {num[j], j } < {num[i],i} do skip
8     end
9     num[i] := 0
10    CS
```

forever

L'algoritmo del panettiere è stato modificato invertendo la riga 9 con la riga 10

A. Descrivere cosa comporta tale modifica nell'esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un server multi-process a concorrenza controllata

Un server multi-process predispone una listening socket per accettare connessioni in ingresso, e ad ogni connessione da parte di un client il server crea un processo figlio che si occuperà di gestire la richiesta. La peculiarità di questo server è che pur non volendo per design gestire più di `MAX_CONNECTIONS` contemporaneamente, per ogni connessione ulteriore in ingresso crea un processo figlio che all'occorrenza viene messo in attesa.

Il server riceve un messaggio di lunghezza variabile (inferiore a `1024` bytes, si faccia riferimento al codice) e dopo aver simulato una computazione invia al client una risposta rappresentata da una stringa di lunghezza variabile, comprensiva del suo terminatore. Il livello di concorrenza nel server viene gestito tramite un semaforo named su cui operano i processi che gestiscono le connessioni in ingresso accettate dal server.

Obiettivi

1. Gestione di connessioni in ingresso con paradigma multi-process
2. Comunicazione su socket con lettura best-effort
3. Gestione del livello di concorrenza interno tramite semafori named

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2 (A.A. 2019-2020 e successivi)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello - 26 Gennaio 2022

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Socket (rispondere nel file teoria1.txt)

Descrivere come si utilizza una datagram socket all'interno di un'applicazione client-server, avvalendosi di pseudo codice. Parlare inoltre del protocollo di comunicazione che ne fa uso.

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j];
7           while num[j] != 0 AND {num[j], j } < {num[i],i};
8     end
9     CS
10    num[i] := 0
```

forever

L'algoritmo del panettiere è stato modificato a linea 3 eliminando la somma di 1 al max,

(istruzione corretta: `num[i] := 1 + max {num[j] : 0 <= j < N}`)

A. Descrivere cosa comporta tale modifica nell'esecuzione del programma

B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un sistema multi-process

All'interno di un sistema uno o più processi producer generano degli input che devono poi essere elaborati da un solo processo elaborator. Ogni producer ha un bilancio iniziale in euro e ogni volta che trasmette un input all'elaborator fornisce anche una ricompensa (reward) per la sua elaborazione. Lo scambio dei dati tra i processi avviene tramite una memoria condivisa che contiene un buffer in cui i processi producer e il processo elaborator accedono secondo la modalità produttore/consumatore; gli indici per l'accesso alla memoria, i semafori unnamed di sincronizzazione. Memoria e semafori sono istanziati dall'elaborator, che deve pertanto essere avviato prima di qualsiasi producer. Ogni volta che un producer genera un task, memorizza input e reward all'interno di una struttura dati che viene inserita all'interno del buffer e decrementa il proprio bilancio, quando il bilancio non è sufficiente a pagare il task il producer termina la propria attività. L'elaborator preleva un input del task dopo l'altro insieme al reward, che viene sommato al proprio bilancio.

Si chiede di completare i file common.h, producer.c e elaborator.c

Obiettivi

1. Gestione di una memoria condivisa
2. Paradigma produttore/consumatore tramite semafori unnamed inclusi nella memoria condivisa

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2 (A.A. 2019-2020 e successivi)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello straordinario - 11 Giugno 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Reti (rispondere nel file teoria1.txt)

Descrivere i protocolli TCP e UDP, illustrando le differenze tra di essi

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := 1+ max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j] do skip
7           while num[j] != 0 AND {num[j], j } < {num[i],i} do skip
8     end
9     CS
10    num[i] := 1
```

forever

L'algoritmo del panettiere è stato modificato alla riga 10 cambiando l'istruzione che dopo la CS azzera il num del processo in `num[i] := 1`

A. Descrivere cosa comporta tale modifica nell'esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un sistema multi-process

All'interno di un sistema uno o più processi producer generano degli input che devono poi essere elaborati da un solo processo elaborator. Ogni producer ha un bilancio iniziale in euro e ogni volta che trasmette un input all'elaborator fornisce anche una ricompensa (reward) per la sua elaborazione. Lo scambio dei dati tra i processi avviene tramite una memoria condivisa che contiene un buffer in cui i processi producer e il processo elaborator accedono secondo la modalità produttore/consumatore; gli indici per l'accesso alla memoria, i semafori unnamed di sincronizzazione. Memoria e semafori sono istanziati dall'elaborator, che deve pertanto essere avviato prima di qualsiasi producer. Ogni volta che un producer genera un task, memorizza input e reward all'interno di una struttura dati che viene inserita all'interno del buffer e decrementa il proprio bilancio, quando il bilancio non è sufficiente a pagare il task il producer termina la propria attività. L'elaborator preleva un input del task dopo l'altro insieme al reward, che viene sommato al proprio bilancio.

Si chiede di completare i file producer.c e elaborator.c

Obiettivi

1. Gestione di una memoria condivisa
2. Paradigma produttore/consumatore tramite semafori unnamed inclusi nella memoria condivisa

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2 (A.A. 2019-2020)

Secondo Appello - 20 Luglio 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Processi e Thread (rispondere nel file teoria1.txt)

Spiegare cosa è Inter-Process Communication e motivare la necessità di meccanismi di IPC.

Fornire illustrazioni cartacee a supporto della risposta, se ritenuto utile.

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo di Dijkstra “modificato”:

```
Initially
     /* global info */
     boolean interested[N] = {false, ..., false};
     boolean passed[N] = {false, ..., false};
     int k = <any> ;       // k ∈ {0, 1, ..., N-1}
     /* local info */
     int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
   1 <Non Critical Section>;
   2 interested[i] = true;
   3 while (k != i) {
   4     passed[i] = false;
   5     if (!interested[k]) then k = i;
   6 }
   7 passed[i] = true;
   8 for j in 0 ... N-1 except i do
   9     if (!passed[j]) then goto 3;
  10 <critical section>;
  11 passed[i] = false; interested[i] = false;
forever
```

L’algoritmo di Dijkstra è stato modificato alla riga 9 negando la condizione

A. Descrivere cosa comporta tale modifica nell’esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - ReverseServer multi processo

Un processo client invia, tramite socket, messaggi a un ReverseServer. Quest'ultimo è un processo (file `server.c`) che, per ogni connessione ricevuta, genera un processo figlio per la gestione delle comunicazioni. Per ogni messaggio ricevuto, il ReverseServer invia al client, tramite socket, la stringa ricevuta invertita. Le funzioni di invio e ricezione su socket andranno implementate nel file `io.c`. Lo studente deve completare e compilare i file `server.c, client.c` e `io.c`. Per testare indipendentemente i file `server.c` e `client.c`, possono essere utlizzati i file `profserver` e `profclient` inclusi nella distribuzione.
Obiettivi principali

1. Gestione multi-processo: creazione/terminazione processi figlio
2. Comunicazione via socket
3. Invio/Ricezione messaggi di lunghezza variabile (con condizione di terminazione)

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2 (A.A. 2019-2020 e seguenti)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello straordinario - 26 Gennaio 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Deadlock (rispondere nel file teoria1.txt)

Descrivere le condizioni necessarie e sufficienti per il verificarsi del deadlock e gli approcci per la gestione del deadlock (non mi riferisco a semafori, monitor….)

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := 1+ max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j] do skip
7           while num[j] != 0 OR {num[j], j } < {num[i],i} do skip
8     end
9     CS
10    num[i] := 0
```

forever

L’algoritmo del panettiere è stato modificato alla riga 7 sostituendo l’operatore logico AND con l’operatore logico OR

A. Descrivere cosa comporta tale modifica nell’esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un server multi-thread a concorrenza controllata

Un sistema offre un servizio di immagazzinamento e recupero delle informazioni, dando maggiore priorità all'informazione più recente. Un client può quindi richiedere al server di immagazzinare un dato (generato casualmente), o di recuperare l'ultimo elemento inserito (anche se l'elemento più recente potrebbe essere stato inserito da un altro client), a seconda del desiderio dell'utente, che ha a disposizione 3 comandi: w (inserisci), r (recupera), q (esci).

Il server multi-thread predispone una listening socket per accettare connessioni in ingresso, e ad ogni connessione da parte di un client il server crea un processo figlio che si occuperà di gestire la richiesta. Il server riceve un messaggio di lunghezza fissa (una struttura contenente il comando e un valore: il valore casuale da scrivere, oppure 0) e dopo aver gestito il comando (se 'q' termina immediatamente il thread) risponde con un intero (il valore richiesto se il comando era 'r', 1 per conferma se il comando era 'w'). I dati vengono memorizzati all'interno di un buffer gestito come una pila. Quando tale buffer è pieno i thread che vogliono scrivere vengono bloccati finché non si libera una posizione. Il livello di concorrenza nel server viene gestito tramite semafori unnamed su cui operano i thread che gestiscono le connessioni in ingresso accettate dal server.

Completare le parti identificate all'interno del file server.c

Obiettivi dell'esercizio

1. Gestione di connessioni in ingresso con paradigma multi-thread
2. Gestione del livello di concorrenza interno tramite semafori unnamed

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2

Esercitazione Finale

Esercizio 1 - Realizzazione di un server multi-thread con comunicazione su socket.

Un server espone N-1 risorse identificate da un numero compreso tra 1 e N-1. I client possono connettersi al server tramite socket e richiedere l'utilizzo di una risorsa inviando un numero tra 1 e N-1. Il server mantiene un array shared_counters con N contatori, dove il contatore i-esimo tiene traccia del numero di richieste ricevute per la risorsa i-esima.

Ogni client viene gestito con un thread (su cui non è previsto fare join) che esegue la funzione connection_handler(). Questo thread riceve comandi dal client ed invoca la funzione process_resource(), che incrementa il contatore di shared_counters relativo alla risorsa richiesta. L'accesso al contatore i-esimo di shared_counters è protetto dal semaforo i-esimo in semaphores. Es: se il client invia al server "1", viene incrementato il contatore in posizione 1. Qualsiasi altro comando viene considerato come "0", ed in tal caso è il contatore in posizione 0 ad essere incrementato. L'unica eccezione è il comando "QUIT", che fa terminare il thread.

Obiettivi:

1. Gestione semafori
    - Inizializzazione semafori
    - Gestione sezione critica nella funzione process_resource()
2. Gestione multi-thread
    - Creazione thread di gestione connessione client
    - Rilascio risorse allocate
3. Gestione scambio messaggi su socket
    - Invio/ricezione messaggi su socket con gestione interruzioni
    - Chiusura descrittori

Domanda 2

Spiegare che cosa sono le operazioni di "compare-and-swap" e "exchange", come possono essere utilizzate per sincronizzare processi per l'accesso ad una sezione critica, possibilmente tramite pseudo-codice. Elencare vantaggi e svantaggi di queste primitive.

Domanda 3

Considerate il seguente algoritmo di Dijkstra per l'accesso in mutua esclusione ad una sezione critica:

```
 /* global storage */

boolean interested[N]  = {false, ..., false}
boolean passed[N]  = {false, ..., false}
int k = <any>   // k ∈ {0, 1, ..., N-1}

/* local info */
int i = <entity ID>  // i ∈ {0, 1, ..., N-1}
while (True) {
        < NCS >
        1.  interested[i] = true
        2.  while (k != i) {
        3.        passed[i] = false
        4.        if ( interested[k] ) then k = i
             }
        5.  passed[i] = true
        6.  for j in 1 ... N except i do
        7.        if ( passed[j] ) then goto 2
        8.  <Critical Section>
        9.  passed[i] = false; interested[i] = false

}
```

Sostituendo nella riga 4 "**if not** interested[k] **then** k=i" con "**if interested**[k] **then** k=i" quali sono le problematiche che si creano legate alle proprietà di correttezza di Mutua_Esclusione e no deadlock? Se e quali vengono violate? Motivare la risposta.

Sistemi di Calcolo 2 (A.A. 2019-2020 e successivi)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello straordinario - 26 Gennaio 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Socket (rispondere nel file teoria1.txt)

Descrivere come si utilizza una datagram socket all'interno di un'applicazione client-server, avvalendosi di pseudo codice. Per quale protocollo di comunicazione si usa?

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := 1+ max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j] do skip
7           while num[j] != 0 AND {num[j], j } < {num[i],i} do skip
8     end
9     CS
```

forever

L'algoritmo del panettiere è stato modificato eliminando l'istruzione che dopo la CS azzera il num del processo (`num[i] := 0`)

A. Descrivere cosa comporta tale modifica nell'esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un sistema multi-process

All'interno di un sistema uno o più processi producer generano degli input che devono poi essere elaborati da un solo processo elaborator. Ogni producer ha un bilancio iniziale in euro e ogni volta che trasmette un input all'elaborator fornisce anche una ricompensa (reward) per la sua elaborazione. Lo scambio dei dati tra i processi avviene tramite una memoria condivisa che contiene un buffer in cui i processi producer e il processo elaborator accedono secondo la modalità produttore/consumatore; gli indici per l'accesso alla memoria, i semafori unnamed di sincronizzazione. Memoria e semafori sono istanziati dall'elaborator, che deve pertanto essere avviato prima di qualsiasi producer. Ogni volta che un producer genera un task, memorizza input e reward all'interno di una struttura dati che viene inserita all'interno del buffer e decrementa il proprio bilancio, quando il bilancio non è sufficiente a pagare il task il producer termina la propria attività. L'elaborator preleva un input del task dopo l'altro insieme al reward, che viene sommato al proprio bilancio.

Si chiede di completare i file producer.c e elaborator.c

Obiettivi

1. Gestione di una memoria condivisa
2. Paradigma produttore/consumatore tramite semafori unnamed inclusi nella memoria condivisa

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---

Sistemi di Calcolo 2 (A.A. 2019-2020 e successivi)

Prova congiunta Sistemi di calcolo 12 CFU - Seconda parte (A.A. 2017-2018 e precedenti)

Appello straordinario - 26 Gennaio 2021

Tempo a disposizione: 1h 45m.

Attenzione: assicurarsi di compilare il file **studente.txt** e che il codice prodotto non contenga **errori di compilazione**, pena una non correzione dell'elaborato.

Regole Esame

- Domande ammesse
  Le domande possono riguardare solo la specifica dell'esame e la struttura di alto livello del codice, nessuna domanda può riguardare singole istruzioni.
- Oggetti vietati
  I seguenti oggetti non devono essere presenti sulla scrivania, né tantomeno usati: smartphone, smartwatch, telefonini, tablet, portatili, dispositivi di archiviazione USB, copie cartacee della dispensa, astucci e qualsiasi forma di libri ed appunti. **Chi verrà sorpreso ad usare uno di questi oggetti verrà automaticamente espulso dall'esame.**
- Modalità di risposta
  Le risposte alle domande di teoria vanno fornite nei file txt indicati. Il professore fornisce fogli per appunti e dove fornire illustrazioni o codice utile ad integrazione delle risposte di teoria. Qualsiasi altro foglio portato dallo studente non può essere usato.
- Azioni vietate
  È assolutamente vietato comunicare in qualsiasi modo con gli altri studenti. **Chi verrà sorpreso a comunicare con gli altri studenti per la prima volta verrà richiamato, la seconda volta verrà invece automaticamente espulso dall'esame**.

Teoria 1 - Socket (rispondere nel file teoria1.txt)

Descrivere come si utilizza una datagram socket all'interno di un'applicazione client-server, avvalendosi di pseudo codice. Per quale protocollo di comunicazione si usa?

Teoria 2 - Algoritmi di concorrenza (rispondere nel file teoria2.txt)

Considerate il seguente algoritmo del panettiere "modificato":

```
Initially
/* global info */
boolean choosing[N]   = {false, ..., false};
integer num[N] = {0, ..., 0};
/* local info */
int i = <entity ID>;  // i ∈ {0, 1, ..., N-1}
repeat
1     NCS
2     choosing[i] := true
3     num[i] := 1+ max {num[j] : 0 <= j < N}
4     choosing[i] := false
5     for j := 0 to N-1 do begin
6           while choosing[j] do skip
7           while num[j] != 0 AND {num[j], j } < {num[i],i} do skip
8     end
9     CS
```

forever

L'algoritmo del panettiere è stato modificato eliminando l'istruzione che dopo la CS azzera il num del processo (`num[i] := 0`)

A. Descrivere cosa comporta tale modifica nell'esecuzione del programma
B. Quale o quali sono le problematiche che si creano legate alle proprietà di Mutua Esclusione, No-Deadlock e No-Starvation? Se e quali vengono violate? Motivare la risposta.

Programmazione - Realizzazione di un sistema multi-process

All'interno di un sistema uno o più processi producer generano degli input che devono poi essere elaborati da un solo processo elaborator. Ogni producer ha un bilancio iniziale in euro e ogni volta che trasmette un input all'elaborator fornisce anche una ricompensa (reward) per la sua elaborazione. Lo scambio dei dati tra i processi avviene tramite una memoria condivisa che contiene un buffer in cui i processi producer e il processo elaborator accedono secondo la modalità produttore/consumatore; gli indici per l'accesso alla memoria, i semafori unnamed di sincronizzazione. Memoria e semafori sono istanziati dall'elaborator, che deve pertanto essere avviato prima di qualsiasi producer. Ogni volta che un producer genera un task, memorizza input e reward all'interno di una struttura dati che viene inserita all'interno del buffer e decrementa il proprio bilancio, quando il bilancio non è sufficiente a pagare il task il producer termina la propria attività. L'elaborator preleva un input del task dopo l'altro insieme al reward, che viene sommato al proprio bilancio.

Si chiede di completare i file producer.c e elaborator.c

Obiettivi

1. Gestione di una memoria condivisa
2. Paradigma produttore/consumatore tramite semafori unnamed inclusi nella memoria condivisa

Altro

- **i commenti nel codice contengono molte informazioni utili per lo svolgimento della prova, si consiglia quindi di tenerli in debita considerazione**
- il file `dispensa.pdf` contiene una copia della dispensa *Primitive C per UNIX System Programming* preparata dai tutor di questo corso
- il file `raccomandazioni.pdf` contiene una serie di considerazioni sugli errori riscontrati più di frequente

---