labso2022-3--esame–info+testo

ISTRUZIONI PRATICHE

Esame del modulo di laboratorio di “Sistemi Operativi”

Durata: 120’ (2 ore)

● Creare una cartella principale denominata con il proprio numero di matricola e dentro tutti i contenuti richiesti dal compito oltre a rispondere alle domande previste nel modulo online.

● Questa cartella andrà consegnata “zippandola” (compressione formato “zip”) in modo da creare un file avente per nome il proprio numero di matricola più l’estensione “.zip”. Deve essere compressa l’intera cartella e non solo il suo contenuto.

Se il proprio numero di matricola fosse 123456 questo deve essere anche il nome della cartella e l’archivio compresso da consegnare deve chiamarsi 123456.zip.

● Consegna:

○ Dopo 50’ ed entro 60’ dall’inizio della prova si deve fare una prima consegna (lavoro parziale) con il lavoro compiuto complessivo fino a tale momento ANCHE SE NON FUNZIONANTE utilizzando il modulo online che sarà indicato.

○ Dopo 90’ ed entro 120’ dall’inizio della prova si deve fare una seconda consegna (lavoro finale) utilizzando il modulo online che sarà indicato (diverso dal precedente): questa consegna è l’unica considerata per la valutazione finale. Rispondere inoltre alle domande valide per la valutazione.

○ I moduli sono “moduli Google”: effettuare il login con il proprio account universitario e inserire tutti i dati richiesti (orientativamente: nome, cognome, numero di matricola e file “zip” da allegare)

○ I punteggi sono indicativi dato che la valutazione tiene conto anche di dettagli “trasversali” che non sono riferibili a singoli punti.

La parte di sviluppo vale fino a 30 punti, mentre la parte con le domande vale fino a 10 punti con diversi quesiti a risposta multipla, vero falso o risposta aperta. Ogni risposta sbagliata contribuisce in maniera negativa al punteggio finale. Il totale massimo è 40 punti.
Il voto equivale al punteggio finale se minore di 30, è pari a 30 per punteggi da 30 a 35 ed è pari a 30L per punteggi da 36 e 40.

NOTA: parte delle verifiche può avvenire con procedure completamente o parzialmente automatizzate per cui le denominazioni e gli output devono essere rigorosamente aderenti alle indicazioni. Dove sono indicate stringhe con sequenze di “escape” vanno rispettate, ad esempio generare un output tipo “7\n” significa che sono presenti esattamente 2 bytes: quello che rappresenta il carattere “7” e un “a capo” (sequenza “\n”. Il carattere “a capo” può essere rappresentato anche dal simbolo ⏎.

Eventuali irregolarità comportano l’esclusione dalla prova oltre a possibili sanzioni disciplinari.

Funzionalità:
Viene richiesta la creazione di un programma (*SignalProxy*) in grado di soddisfare determinate richieste e requisiti. L'implementazione delle funzionalità, e le tecnologie da esse impiegate, sono a discrezione dell'esaminato. Nel seguente testo di esame, per *SignalProxy* ci si riferisce al processo generato all'esecuzione del binario *SignalProxy.out.* Nel resto del testo, ci si riferirà ad un Processo A come ad un qualunque altro processo eseguito sul sistema.

VINCOLI e ACCORGIMENTI:

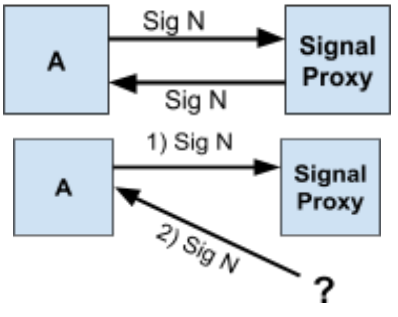
- La cartella finale deve contenere uno o più file sorgente (almeno il "`main.c`") ed eventualmente altri header .h ed un Makefile "`Makefile`". Nessun altro tipo di file, sia esso un eseguibile o un file testuale, è ammesso.
- Qualora non sia presente un Makefile, l'applicazione deve essere compilabile tramite un unico file main.c.
- La sintassi dei comandi deve essere rispettata. In caso di discrepanze il punteggio attribuito sarà nullo.
- Sono concessi output di debugging a discrezione dell'esaminato.
- Il codice consegnato deve essere commentato spiegando le varie funzionalità ed i vari passaggi.

TEST:
Al fine di verificare il corretto funzionamento del vostro elaborato, è vivamente consigliato testare il codice in tutte le sue parti. Al fine di facilitare le operazioni di test, è possibile scaricare un binario apposito al seguente link **bit.ly/verifica-labos**.

ATTIVITÀ:

- **[3/30] Makefile:** realizzare correttamente il "Makefile" e far sì che digitando "`make FILE=<path>`" venga creato il binario *SignalProxy* (partendo dai file sorgente nella cartella). Inoltre, se il file <path> non esiste, esso deve essere creato dal Makefile contenente la scritta "start⏎". Qualora il file esistesse già, il makefile deve semplicemente creare il binario, senza modificare il contenuto del file. Creare infine una regola "clean" per la rimozione del binario generato e del file <path>.

- Creare un programma in C *SignalProxy* che accetti esattamente un parametro *<pathToLogFile>* (quindi che possa essere lanciato con "`./SignalProxy <pathToLogFile>`"). E che gestisca le seguenti funzionalità:
  - **[5/30] Repeater:** se un processo A invia un segnale SIGUSR1 al processo *SignalProxy,* il processo A deve ricevere subito un segnale SIGUSR1 dallo stesso processo contattato da A.



  - **[5/30] Relay di un segnale:** se un processo A invia un segnale SIGUSR2 al processo *SignalProxy*, il processo A deve subito ricevere lo stesso segnale SIGUSR2 da un processo B qualunque, diverso da quello contattato da A.

  - **[6/30] Signal Logger:** se un processo A invia un segnale SIGUSR1 o SIGUSR2 a *SignalProxy*, il file *logFile* viene aggiornato **dopo 3 secondi** (append) con la dicitura "*#pidA-#signNo*⏎" dove, *#pidA* è il PID del processo A, *#signNo* è il numero del segnale inviato. Al comando <CTRL-C> il programma deve scrivere "stop⏎" in fondo al file (scritto subito senza attendere). Durante i 3 secondi di ritardo, il programma deve essere in grado

di rispondere a qualunque altra richiesta. Si può assumere che il file esista già (per esempio è stato creato manualmente o tramite makefile).

- **[6/30] Queue checker:** Se un processo A scrive un messaggio sulla coda associata alla chiave generata dalla coppia *(*`<logFile>, 1`*)* il PID di un processo "`#noPid`", *SignalProxy* deve occuparsi, in qualche modo, di far ricevere al processo con il `PID = #noPid` un segnale `SIGALRM`. Si può assumere che la coda esista e che il PID comunicato abbia una sintassi corretta e corrisponda ad un processo esistente.

- **[5/30] Bash script**: Creare uno script bash che riceva come parametro il nome di un file avente la stessa sintassi sopra citata, e restituisca tutte le righe stampate in maniera alternata su *stderr* e *stdout* (con la prima stampata su stdout).