labso2021-3--esame-info+testo

INDICAZIONI

All'interno della cartella principale creare solo ed esattamente 3 file: un sorgente C "`main.c`" (che dovrà generare un eseguibile denominato "`app`"), un makefile "`Makefile`" (eliminare file temporanei, di servizio, generati, etc. prima della consegna!!!) e uno script bash "`run.sh`".

Realizzare un'applicazione in C che accetta in input esattamente due argomenti: il primo è la "modalità" di esecuzione e deve essere la stringa "`server`" o "`client`", mentre il secondo è il nome di un file di "log" con eventuale percorso (relativo o assoluto).
Il codice di uscita deve essere 0 se non ci sono errori, maggiore di 0 altrimenti.

Esempi di chiamate: `./app server /tmp/log.txt` oppure `./app client /tmp/log.txt`

L'applicazione deve realizzare le funzionalità indicate più avanti riportando eventuali messaggi d'errore (ad esempio numero argomenti errato o valori non accettabili, errori di I/O, altro) su *stderr* con un codice di uscita maggiore di 0 (eventualmente come specificato di seguito).

1. [2 punti] Il Makefile deve funzionare in modo che eseguendo il comando "`make`" senza alcun parametro l'applicazione sia correttamente compilata generando l'eseguibile denominato "`app`", mentre con il comando "make NAME=..." la compilazione deve essere eseguita generando un eseguibile con il nome passato come argomento.

   Esempi:
   `make` -> genera eseguibile di nome `app`
   `make NAME=exam` -> genera eseguibile di nome `exam`

   L'applicazione "`app`" generata compilando "`main.c`" deve poi comportarsi in modo differente a seconda del primo parametro.

2. Se il primo parametro è "`server`":

   a. [5 punti] Scrivere sul file di "log" il proprio "pid" seguito da un "a capo" e poi restare in attesa di un segnale SIGUSR1, SIGUSR2 o SIGINT stampando su *stdout* la stringa "`[server:`*xxx*`]\n`" dove *xxx* è il proprio "pid". Se il file esiste già o non lo si può creare e utilizzare si ha un errore. Se riceve SIGINT deve scrivere sul file di log il numero di figli ancora presenti e poi uscire.

   b. [4 punti] Se si riceve il segnale SIGUSR1 si deve generare un nuovo processo figlio e scriverne il "pid" nel file di log preceduto dal carattere "+" stampando anche la stessa informazione su *stdout* però in questo caso preceduta da "`[server]` "

   c. [4 punti] Se si riceve il segnale SIGUSR2 si deve terminare uno qualunque dei figli generati in precedenza e scriverne il "pid" nel file di log preceduto dal carattere "-" o scrivere "0" (zero) se non ci sono figli, stampando anche la stessa informazione su *stdout* però in questo caso preceduta da "`[server]` "

3. Se il primo parametro è "`client`":

a. [5 punti] Leggere dal file di "log" la prima riga (che dovrebbe contenere il "pid" dell'applicazione lanciata con parametro "server") e stampare a video la stringa "`[client] server: xxx`" (con il "pid" letto al posto di *xxx*). Se il file non esiste NON si ha errore, ma si deve continuare a provare a leggerlo. Successivamente si deve restare in attesa della pressione di un tasto da parte dell'utente. Il processo termina con SIGINT (ad es. con "CTRL+C" da terminale)

b. [4 punti] Se si preme il tasto "+" (più) si deve inviare un segnale SIGUSR1 al processo "server" e incrementare un contatore interno di 1 (fino al massimo di 10, altrimenti non si deve fare nulla), se invece si preme il tasto "-" si deve inviare un segnale SIGUSR2 analogamente e decrementare lo stesso contatore (fino al minimo di zero, altrimenti non si deve fare nulla). Dopo ogni pressione di uno di questi tasti si deve visualizzare su *stdout* il valore del contatore interno aggiornato preceduto da "`[client] `"

c. [4 punti] Se si preme il tasto "ENTER" ("a capo") si devono inviare al processo "server" tanti SIGUSR2 quanto è il valore del contatore interno (intervallati da un secondo di attesa) stampando ogni volta il valore come sopra e poi un segnale SIGINT e quindi terminare.

4. [2 punti] Realizzare uno script bash denominato "run.sh" impostando il flag di esecuzione (flag "x") e con la riga di "hash-bang" iniziale corretta in modo che sia eseguibile dalla cartella corrente semplicemente digitando "`./run.sh`" che richiami la compilazione del file usando "`make`" e poi la esegua passando gli argomenti che riceve a sua volta. Se ad esempio si scrive `./run.sh server /tmp/log.txt` si deve invocare make e poi lanciare l'applicazione con argomenti `server` e `/tmp/log.txt`.