LABORATORIO SISTEMI OPERATIVI 1 – NAIMOLI – A.A.2014-2015

Specifiche di progetto

- il progetto deve essere eseguito in un gruppo composto da massimo 2 persone

- Ogni scadenza è inviolabile e non sono ammessi ritardi, pena l'esclusione automatica dall'esame

- per sostenere l'esame è obbligatorio iscriversi inviando una mail all'indirizzo andrea.naimoli@unitn.elementica.com entro il 30/04/2015 indicando i componenti del gruppo secondo lo schema:

Cognome_1
Nome_1
Matricola_1
Cognome_2
Nome_2
Matricola_1

Esempio:
Rossi
Mario
012345
Bianchi
Lucia
678901

- il linguaggio di riferimento è il C (standard) con le sole strutture discusse (come visto a lezione)

- il progetto deve poter funzionare nell'ambiente di riferimento utilizzato a lezione (QEMU)

- la consegna deve avvenire con spedizione via e-mail all'indirizzo andrea.naimoli@unitn.elementica.com entro il 24/05/2015 con oggetto "*LABSO2015 – Progetto* " seguito dai numeri di matricola di ciascun componente del gruppo, esempio: "*LABSO2015 – Progetto 012349 056789*"

- il materiale deve essere contenuto in un unico file in formato ".tar.gz": il nome del file deve iniziare con "*LABSO2015_Progetto_*" seguito dal titolo del progetto e dal numero di matricola di ciascun compenente del gruppo che ha contribuito alla realizzazione, esempio: "*LABSO2015_Progetto_nomeprogetto_012349_056789.tar.gz*"

- l'archivio ".tar.gz" deve contenere una cartella con lo stesso nome del file con all'interno esclusivamente un "Makefile" (CHE DEVE ESSERE COMMENTATO), una sottocartella "src" (con eventuali sottocartelle) con tutti i file dell'elaborato, senza alcun contenuto automaticamente generato

- la compilazione/esecuzione dell'elaborato deve poter avvenire accedendo alla cartella di progetto e tramite l'esecuzione di un "make" con le seguenti modalità:
    - senza target definiti, si deve generare in output una breve descrizione del progetto e la lista delle opzioni di make (altri target) disponibili
    - con un target "bin" (make bin) si devono generare i binari compilati eseguibili dentro una cartella da crearsi automaticamente denominata "bin" che conterrà i soli file oggetto eseguibili
    - con un target "assets" (make assets) si devono poter generare dei possibili file di input (di prova) dentro una cartella da crearsi automaticamente denominata "assets" (cioè se il progetto può lavorare su dei file di input, questa direttiva ne genera un set utilizzabile)
    - con un target "test" (make test) si generano i binari e gli assets e si lancia l'eseguibile utile a provare i file di input generati
    - con un target "clean" si eliminano eventuali file temporanei, binari e assets generati (questa regola deve essere sempre richiamata automaticamente dalle altre)
    - altri target sono implementabili a discrezione, se ritenuti utili
    - in sede di discussione verranno poste delle domande potenzialmente su tutto il programma svolto in laboratorio a ciascun componente del gruppo per verificare il grado di conoscenza, apprendimento e collaborazione avuta nelle scelte architetturali, di progettazione e funzionamento

- tutto il codice sorgente deve essere BEN documentato: codice parzialmente o non documentato non sarà neppure preso in esame

- ogni gruppo è tenuto a presentare una relazione in italiano di non più di 4 pagine in cui sono descritti gli scenari, l'approccio architetturale che implementa la soluzione adottata: non allegare il codice sorgente alla relazione, non oltrepassare il limite di pagine sopra indicato e limitare la parte teorica: la relazione deve contenere solo elementi teorici introduttivi, per poi focalizzare sulle scelte fatte dai componenti del gruppo di lavoro, conseguenze delle scelte fatte, problemi riscontrati e limiti (con eventuali schemi / snapshot)

- ogni file presente nell'elaborato deve indicare chiaramente gil autori (nomi, cognomi, numeri di matricola e possibilmente una "fototessera") il titolo del progetto e l'indicazione del corso e dell'anno accademico

- progetti palesemente copiati o affini comportano l'immediata esclusione degli studenti facenti parte dei gruppi (sia del gruppo che ha copiato sia quello che consente l'atto di copiatura)

- per porre domande via email utilizzare il forum pubblico in ESSE3 e spedite all'indirizzo andrea.naimoli@unitn.elementica.com: domande ritenute non inerenti al progetto, o quelle che chiedono informazioni già discusse nel presente documento saranno ignorate

Progetto 01/02: SIMULATORE MULTIPLAYER GAME

Realizzare un simulatore di un multiplayer game secondo i seguenti criteri:

- l'eseguibile deve accettare un argomento obbligatorio con valore “server” o “client” che lancia un processo di tipo server o client
- se eseguito come “server”:
 . si accerta non ci siano altri server già in esecuzione
 . resta in attesa di processi client che partecipino al gioco
 . gestisce l'andamento del gioco
 . accetta come ulteriori argomenti: “--max <max>” per impostare il numero massimo di giocatori (fino a 10) e “--win <win>” per impostare il punteggio di vincita (un intero da 10 a 100)
 . mostra in output un monitoraggio della situazione, visualizzando i parametri attuali, l'entrata e uscita dei giocatori, i tentativi di risposta, la classifica aggiornata e il vincitore
- se eseguito come client:
 . verifica la disponibilità del processo server e prova a collegarsi
 . ascolta l'eventuale risposta del server alla disponibilità di un posto libero nel gioco
 . se accettato partecipa al gioco
 . mostra in output la “question” attiva e attende una risposta dall'utente (*)

Regole del gioco.
Il server genera due numeri casuali da 0 a 99 di cui sarà richiesta la somma (*) mostrando a tutti i giocatori la richiesta (esempio: “23 + 41?”): i client ricevono tale richiesta e la mostrano all'utente in attesa di una risposta che poi deve essere inviata al server. All'ingresso nel gioco ogni giocatore riceve un numero di punti pari al numero di giocatori che ancora possono partecipare. Successivamente si perde un punto per ogni risposta errata e se ne guadagna uno per ogni risposta giusta. Quando un giocatore risponde correttamente la domanda cambia per tutti. Quando un giocatore raggiunge o supera il punteggio-traguardo a tutti è comunicato il risultato.

Extra interessanti:
- gestire l'interruzione di server e client sulle controparti
- informare tutti i clienti su entrata/uscita di altri partecipanti e alla conclusione del gioco inviare anche classifica e vincitore

(*) come modello si possono usare cose come:

Numero casuale	Input numerico

`#include <time.h>`
`  #include <stdio.h>`
`  #include <stdlib.h>`
`  void main() {`
`  srand(time(NULL));`
`  int r = rand() % 100;`
`  printf("%d\n", r);`
`  }`
	`#include <stdio.h>`
	`  #include <stdlib.h>`
	`  int main(void) {`
	`  char buf[BUFSIZ];`
	`  int i;`
	`  printf ("Enter number: ");`
	`  if (fgets(buf, sizeof(buf), stdin) != NULL) {`
	`  i = atoi(buf);`
	`  printf ("You entered %d\n", i);`
	`  }`
	`  return(0);`
	`  }`

Progetto 02/02: SHELL CON LOGGER

Implementare una shell interattiva (che accetti almeno comandi basilari) con le seguenti caratteristiche:
- deve accettare tre argomenti opzionali: “--prompt <prompt>”, “--loglevel <loglevel>” e “--logfile <logfile>” per settare un prompt personalizzato, un livello di “logging” (un valore tra “low”, “middle” e “high”) e il file di log: argomenti errati devono mostrare un opportuno avviso
- mostra un prompt personalizzato (di default o settato con l'argomento) e accetta comandi tali che sono passati alla shell standard, a meno che il primo carattere non vuoto sia un punto esclamativo, in questo caso ciò che segue è interpretato come un comando “interno” speciale:
 . !showlevel : mostra il livello impostato (default o settato con l'argomento)
 . !logon : attiva il logging
 . !logoff: disattiva il logging
 . !logshow : mostra il log attuale (nome del file, eventualmente di default + contenuto)
 . !setlevel <loglevel> : cambia il livello di log
 . !setpromp <prompt> : cambia il prompt
 . !run … : esegue l'azione successiva in un processo separato
 . !quit : esce dalla shell
- la shell scrive in un file di log (di default o scelto dall'utente) le azioni svolte, a seconda del livello impostato:
 . ogni riga inizia con un riferimento al timestamp (*) seguito dal comando eseguito (senza parametri)
 . se il livello di log è “low” non ci sono altre informazioni intermedie
 . se il livello di log è “middle” deve essere riportata l'intera stringa (con tutti i parametri)
 . se il livello di log è “high” deve essere anche riportata l'indicazione se si tratta di un comando interno o esterno
 . ogni riga termina con lo status (codice d'errore) dell'esecuzione del comando lanciato

(*) come modello si può usare qualcosa come:

```
#include <stdio.h>
#include <time.h>
void main() {
  char s[1000];
  time_t t = time(NULL);
  struct tm * p = localtime(&t);
  strftime(s, 1000, "%A, %B %d %Y", p);
  printf("[%s]\n", s);
}
```