Progetti Laboratorio SISTEMI OPERATIVI 1
A.A. 2013-2014

UNITN

release 1.0

Specifiche di progetto

- gli studenti si devono unire in gruppi di 2 persone

- ogni gruppo deve scegliere uno dei 3 progetti proposti

- le specifiche di progetto e le regole di ammissione all'esame di laboratorio sono contenute nel presente documento: sarà sempre disponibile online in ESSE3 l'ultima versione aggiornata da ritenersi ufficiale e insindacabile in sede di esame

- sono ammessi all'esame di laboratorio i soli gruppi definiti attraverso lo spreadsheet raggiungibile su http://goo.gl/voAvpx (e poi pubblicati in ESSE3): inserire nel primo foglio il nome del progetto e il riferimento alla matricola del “compagno di gruppo”, nel secondo foglio riportare nome, cognome e matricola di entrambi e il titolo del progetto

- ogni gruppo (per voce di un membro: inviare un’unica mail) deve inviare il progetto scelto entro la scadenza attualmente fissata nel giorno 10/06/2014 per mezzo posta elettronica labSO@dit.unitn.it

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

- per sostenere l'esame è obbligatorio iscriversi in ESSE3

- il linguaggio di riferimento è il C

- il progetto deve funzionare nell'ambiente di riferimento utilizzato a lezione

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

- ogni file presente nell'elaborato deve indicare chiaramente gil autori (nomi, cognomi, numeri di matricola) 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 utilizzate il forum pubblico in ESSE3: domande ritenute non inerenti al progetto, o quelle che chiedono informazioni già discusse nel presente documento saranno ignorate

- il docente e i suoi assistenti si riservano il diritto di modificare l'assegnazione dei progetti ai vari gruppi

Progetto #01/03: CODEC (COdificatore DECodificatore)

Sistema Server-Client per codifica-decodifica testo tramite traslitterazione sull'alfabeto inglese.

Implementare un CODEC che si basi sull'esecuzione di un "server" come supporto di tutti i servizi necessari utilizzabili da uno o più "client". Più server possono essere eseguiti contemporaneamente.

La (de)codifica si basa sul principio che un messaggio sia traslitterato facendo corrispondere ad ogni lettera dell'alfabeto un'altra attraverso l'uso di una chiava alfabetica usata per calcolare ciclicamente uno scarto posizionale. Si considera la traslitterazione solo delle lettere dell'alfabeto anglosassone: procedendo un carattere alla volta si prende un carattere del messaggio e il corrispondente della chiave (senza distinzione minuscole/maiuscole e ripetendo la chiave se di dimensione inferiore al messaggio), si calcola la posizione nell'alfabeto di quest'ultima e la si aggiunge come distanza alla prima (ciclicamente).

Esempio:

messaggio = "`Testo del messaggio`"
chiave = "`Ciao`"

Codifica:

```
 T  e  s  t  o     d  e  l   m  e  s  s  a  g  g  i  o
20  5 19 20 15     4  5 12  13  5 19 19  1  7  7  9 15

 C  i  a  o  C     i  a  o   C  i  a  o  C  i  a  o  C
 3  9  1 15  3     9  1 15   3  9  1 15  3  9  1 15  3
```

(sommare le posizioni, ciclicamente se si supera l'ultima posizione)

```
23 13 20  9 18    13  6  1  16 14 20  8  4 16  8 26 18
 W  m  t  i  r     m  f  a   p  n  t  h  d  p  h  z  r
```

messaggio codificato: "`Wmtir mfa pnthdphzr`"

Il "server" deve essere configurabile all'esecuzione con i seguenti parametri:
- denominazione identificativo del servizio
- lunghezza massima messaggio
- lunghezza minima e massima della chiave

Durante l'esecuzione deve rispondere a semplici interazioni che consentano di:
- visualizzare l'elenco dei messaggi codificati

- decodificare uno dei messaggi codificati accettando in input la chiave
- codificare un messaggio accettando in input la chiave e il messaggio

I “client” devono poter accedere al servizio di codifica/decodifica ricevendo come parametri:
- l’identificativo del servizio (per scegliere a quale “server” collegarsi)
- una chiave testuale e un messaggio (testuale o file esterno) indicati nella riga di comando
- un’opzione che indichi se si vuole effettuare una codifica o una decodifica
- un’eventuale file di output

Il server effettua l’input/output a video, mentre il client può leggere i messaggi da un file esterno e l’output può andare a video o su file.

ESEMPIO INVOCAZIONI:

```
codecserver -name codec1 -min 5 -max 10
```

(imposta il codec server “codec1” con chiavi da 5 a 10 caratteri)

```
codecclient -name codec1 -key ciao -file -message msg_org.txt -encode
-output msg_enc.txt
```

(attiva un client sul codec “codec1” usando come chiave “ciao” e leggendo i messaggi da un file sterno denominato “msg_org.txt” e li codifica salvandoli in “msg_enc.txt”)

Progetto #02/03: SIMULATORE FASE FINALE CAMPIONATO DEL MONDO DI CALCIO

Realizzare un simulatore della fase finale del campionato del mondo di calcio secondo i seguenti criteri:

- la gara coinvolge 32 squadre
- ogni squadra ha una denominazione e un valore di abilità da 1 a 10 (configurabile come parametro o pari a 7 di default)
- ogni partita deve essere un processo a sè
- ogni partita ha una durata temporale di 45 minuti virtuali per tempo più da 1 a 5 minuti virtuali di recupero per tempo: in caso di pareggio devono essere simulati due tempi supplementari di 15 minuti virtuali ciascuno più da 1 a 3 minuti virtuali di recupero e in caso di ulteriore parità uno scontro ai rigori al meglio dei 5 tiri o a oltranza se necessario
- la durata reale per i tempi regolamentari e quelli supplementari è configurabile come parametro (altrimenti 5 secondi di default)
- la durata reale dei minuti di recupero è di 1 secondo per ogni minuto
- la quantità di minuti di recupero è casuale
- durante i tempi regolamentari c'è il 50% di probabilità che avvenga un gol ogni 15 minuti, in quelli di recupero ogni 5 minuti e la probabilità che segni una squadra è proporzionale al rapporto tra la sua abilità e quella dell'avversario (esempio con squadre di abilità 8 e 4: 8/4=2; significa che la prima squadra ha il doppio di probabilità di segnare dell'avversario)
- ogni rigore ha una probabilità del 75% di essere segnato (25% di essere parato)
- ogni vittoria da 3 punti, ogni pareggio 1 punto, ogni sconfitta 0 punti

L'implementazione deve accettare come parametri:
- la durata in secondi reali di ogni minuto virtuale
- un riferimento a un file di configurazione in cui siano indicate per ogni riga:
  `nomesquadra abilità` (spazio nel mezzo: per esempio "`Italia 9`")
- un possibile riferimento a un file di ouput dove copiare le informazioni calcolate

L'esecuzione prevede:
- formazione completamente casuale di 8 gruppi di 4 squadre per 8 gironi di "sola andata"
- determinazione classifiche dei giorni (numero punti, differenza reti, reti segnate, scontro diretto, casualità)
- abbinamento completamente casuale dei gironi a coppie per gli scontri successivi: prima di un girone contro seconda dell'altro e viceversa
- svolgimento con scontri diretti di ottavi, quarti, semifinali e finalissima
- in output deve visualizzare:
  - . fase a gironi:
    - . elenco gironi ordinati
    - . inizio di ogni partita
    - . fine di ogni parziale (primo e secondo tempo, eventuali supplementari e rigori)
    - . risultato finale
  - . fase a eliminazione diretta:
    - . visualizzazione fase in corso con gli abbinamenti creati
    - . svolgimento di ogni partita
  - . proclamazione vincitore

Progetto #03/03: TOOLS LINUX

Implementare i seguenti “tool” per linux senza ricorrere all’uso dei comandi “interni” già esistenti (sfruttando la cartella “/proc” e le sottocartelle come “/proc/modules):

custom-ps: simula il comando “ps” e almeno 3 opzioni non banali

custom-lsmod: simula il comando “lsmod” con la possibilità di passare come opzione un valore per il riordinamento alfabetico per nome o numerico per id

custom-console: una semplice shell che accetta comandi dell’utente anche con opzioni e li esegue e accetta le seguenti opzioni (senza indicarne alcuna o con “-h” deve mostrare un testo di help con le istruzioni):

-o nomefileoutput (utilizzabile insieme alle altre opzioni: effettua un log di tutte le esecuzioni nel file indicato: in ogni riga ci devono essere almeno il “timestamp” di avvio esecuzione, quello di fine esecuzione, il nome del comando e la riga completa con le opzioni)
-r (anche insieme a “-o”: attiva la shell interattiva, è alternativo a “-a”)
-a (insieme a “-o”: analizza il file di output, è alternativo a “-r”. L’analisi consiste in un report in cui si mostra l’elenco in ordine alfabetico dei comandi eseguiti, il numero di esecuzioni per ciascun comando e la durata totale di utilizzo di ciascuno)

Durante l’esecuzione della shell se si immette un comando che inizia con il carattere “!” (punto esclamativo) significa che si vuole eseguire un’azione interna:
`!o-name` (mostra il nome del file di log attuale)
`!o-print` (mostra il nome del file di log attuale e stampa il contenuto a video)
`!o-analyze` (mostra l’analisi del file di log)
`!quit` (esce dalla shell)

Esempi:
`custom-console -o /log/custom-console.log -r` avvia la shell e registra i log
`custom-console -o /log/custom-console.log -a` effettua l’analisi, per esempio:

```
ls 7 3’42”
ps 3 12”
```

(significa che il comando “ls” è stato eseguito 7 volte per 3’42” totali di tempo e “ps” per 3 volte e 12” totali)