Progetto per il corso di Sistemi Operativi 1
A.A. 2012-2013 v1.1

Regole per il progetto di laboratorio:

- Le specifiche di progetto e le regole di ammissione all'esame di laboratorio sono contenute nel presente documento: sara' sempre disponibile online in ESSE3 l'ultima versione aggiornata da ritenersi ufficiale e insindacabile in sede di esame.
- Sono ammessi all'esame di laboratorio solo gruppi composti da massimo tre (3) studenti. I gruppi che presentano il proprio progetto devono obbligatoriamente comunicarlo per mezzo posta elettronica a **labso@disi.unitn.it** entro il **31/03/2012.**
- La mail dovrà contenere il cognome, nome e la matricola di ciascun membro del gruppo, mettendo per primo il rappresentate del gruppo, secondo lo schema:

----SCHEMA----
Cognome_1 Nome_1 Matricola_1 Titolo progetto
Cognome_2 Nome_2 Matricola_2
----ESEMPIO----
Bianchi Luca 256252 Progetto2
Rossi Mario 256345

- La consegna del progetto deve avvenire entro il **03/06/2012** in formato elettronico all'indirizzo di posta elettronica **labso@disi.unitn.it**.
- LE DUE SCADENZE SONO INVIOLABILI E NON SONO AMMESSI RITARDI, PENA ESCLUSIONE AUTOMATICA DALL'ESAME.
- Il linguaggio di riferimento è il C.
  - Il progetto deve funzionare nell'ambiente di riferimento utilizzato a lezione. Il sistema di riferimento è GNU/Linux basato sulla distribuzione **Debian 6.0.5 "Squeeze"** - **kernel 2.3.4** scaricato da http://www.kernel.org e compilato secondo quanto visto in laboratorio e riportato nelle slide disponibili su ESSE3. Deroghe a questa regola sono ammesse solo se concordate con i docenti e motivate anche in fase di stesura della relazione.
- FUNZIONAMENTO = compilazione + esecuzione.
- La presentazione del progetto avverrà utilizzando il sistema di riferimento virtualizzato mediante qemu. Il PC utilizzato e' quello del docente e non e' ammesso l'uso di laptop personali per tale presentazione.
- Il progetto spedito a **labso@disi.unitn.it** deve essere contenuto in un unico file in formato tar.gz. Il nome del file deve iniziare con il numero del progetto scelto (e.g. progetto2), seguito dal nome del rappresentante del gruppo e dal numero di matricola del rappresentante del gruppo, e.g., progetto2-LucaBianchi-256252.tar.gz. Per chiarire eventuali dubbi su come comprimere in tar.gz consultate "man tar" e "man gzip" nella shell.
- Il file tar.gz conterra' SOLO il codice sorgente e tutto il necessario per la compilazione ed esecuzione a meno della versione del kernel di riferimento sopra indicato: non vi sara' codice oggetto prodotto durante la fase di sviluppo, test o file temporanei.
- Scompattando l'archivio tar.gz, i file dell'intero progetto devono essere contenuti in una directory denominata prjroot-matricole-studenti, e.g., prjroot-256252-256345, inserendo il numero di matricola nell'ordine in cui sono stati riportati nella

mail relative alla comunicazione del gruppo. Ovviamente tale directory puo' contenere altre directory al suo interno.

- A tutti i membri del gruppo, in sede di discussione, verranno poste delle domande potenzialmente su tutto il programma svolto in laboratorio 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 soprattutto laddove vi sono modifiche a codice esistente: codice parzialmente o non documentato non sarà neppure preso in esame.
- Ogni file presente nell'archivio tar.gz deve indicare al suo interno l'anno accademico, il corso di studio, il titolo del progetto e gli autori (cognome, nome, numero di matricola).
- E' obbligatoria l'adozione degli strumenti offerti da Makefile.
- Saranno valutati inoltre: (i) pulizia del codice, (ii) strutturazione del codice in più' file sorgenti (.h .c).
- Ogni gruppo e' tenuto a presentare una relazione di massimo sei (6) pagine in italiano in formato PDF. Nella relazione sono descritti gli scenari, l'approccio architetturale che implementa la soluzione adottata, ed eventuali grafici. Attenzione, non riportare il codice sorgente nella relazione, non riscrivere il testo del problema, non oltrepassare il limite di pagine sopra indicato, limitando allo stretto necessario la parte teorica. L'obiettivo e' produrre un documento sintetico e completo che colleghi il problema alla soluzione proposta dal gruppo di studenti. La relazione può contenere elementi teorici introduttivi, ma deve focalizzarsi sulle scelte fatte dai componenti del gruppo di lavoro, conseguenze delle scelte, problemi riscontrati ed eventuali limiti.

  La relazione deve concludersi con una sequenza di comandi. Tali comandi, se eseguiti, devono dare i risultati attesi e presentati nella relazione stessa.
- Progetti palesemente copiati (o affini) da altri colleghi o dagli esempi dati in classe comportano l'immediata esclusione degli studenti facenti parte dei gruppi: sia del gruppo che ha copiato sia quello che ha permesso di copiare.
- Per le domande spedite una email a **labso@disi.unitn.it** o utilizzate il forum pubblico in ESSE3: Se utilizzate il forum, anche gli altri studenti ne trarranno vantaggio! Domande ritenute non inerenti al progetto, o che chiedono informazioni già discusse nel presente documento saranno ignorate.

Progetto 1: Modifica utility di base di GNU/Linux

Premessa

Le utility di base di GNU/Linux (ls, cd, cat, ecc.) sono distribuite all'interno di tutte le distribuzione GNU/Linux. Il progetto e' quello di implementare le utility elencate ai punti seguenti:

Utility:

mkbkp

- crea un file di archivio per il salvataggio di files e directories passati come argomenti
- flags:
  - -f <archivio> indica l'archivio da creare od estrarre.
  - -c indica la creazione dell'archivio
  - -x indica l'estrazione dell'archivio nella directory corrente
  - -t indica che deve essere visualizzato un elenco del contenuto dell'archivio

es:

mkbkp -c -f etc.bkp /etc

produce un file "etc.bkp" contenente un salvataggio di tutti i files presenti in /etc

equal

- Prevede due argomenti, che possono essere nomi di files o directories.
- Ritorna 0 se gli argomenti hanno contenuto identico
- Ritorna 1, se gli argomenti differiscono, elencando a video quali elementi differiscono

plive

- *Ogni secondo mostra un elenco dei (-n <num>, default 10) processi che usano più cpu sul sistema, in ordine di occupazione.*
- *Per ogni processo vengono mostrati:*
  <process id> <process id del padre> <percentuale di cpu usata> <nome dell'eseguibile del processo>
- *premendo un numero da 1 a 9 la visualizzazione si aggiorna al numero di secondi corrispondente; premendo "q" il comando termina.*

Note:

- Obbligatorio usare la system call *getopt*
- Si possono usare solo le system call, all'interno del codice sorgente non e' ammesso l'utilizzo di comandi gia' disponibili nel sistema GNU/Linux in forma di codice oggetto compilato
- Tutti i processi generati non devono produrre processi zombie
- Ogni utility sviluppata deve produrre il corretto comportamento a video e, allo stesso tempo, registrare tutte le attivita' di input, output ed error in file di log che devono obbligatoriamente essere memorizzati in /var/log/utility/<nome_utility> con il medesimo nome dell'utility in uso.
- Se da console viene cancellato un file di log, la successiva invocazione del comando dovra' segnalare il fatto che il file non esiste e ricrearlo.

Progetto 2: Threads

Si deve implementare un programma che genera 4 threads: *Tr*, *Te*, *Td*, e *Tw*.

- *Tr* legge da tastiera una sequenza di caratteri (*S*) arbitrariamente lunga fino a che non riconosce il carattere <CR> (ENTER).
- *Te* legge dal device "*/dev/random*" una stringa di caratteri *R* della dimensione di *S*, e successivamente, ne fa lo XOR byte-per-byte con S ottenendo Se = XOR(R, S).
- *Td* legge *R* e *Se* e calcola *Sd = XOR(R, Se).*
- *Tw* stampa a schermo la stringa *Sd*.

Esempio:
$> Testo di prova <CR>
R: df546tgfg435trf
Se: j43jd9k30-fi8h3
Sd: Testo di prova
$>

Specifiche:

- *Tr* e' un "produttore di dati" che termina solo quando riconosce la stringa in ingresso "*quit*".
- I messaggi di log dei *threads* devono comparire nella cartella */var/log/threads* all'interno dei file *tr.log, te.log, td.log, tw.log.*

Note:

- Obbligatorio usare la system call *getopt*
- Si possono usare solo le system call, all'interno del codice sorgente non e' ammesso l'utilizzo di comandi gia' disponibili nel sistema GNU/Linux in forma di codice oggetto compilato
- La terminazione di Tr non deve interrompere l'elaborazione degli altri threads.
- Ciascuna stringa di log deve essere caratterizzata da data e ora.

Progetto 3: Comunicazione fra processi

Si deve implementare un programma che genera una sequenza di processi che comunicano fra di loro. Il programma si chiama *prog* e la sintassi e' la seguente:

./prog “cmd1 arg1_1” “cmd2 arg1_2 arg2_2“ “cmd3” ...

dove cmdX rappresenta un comando qualunque della shell linux mentre argY_X rappresenta l'argomento Y passato al comando X.
Ciascun processo generato dovra' comunicare al proprio figlio l'output che a sua volta lo processera', i.e., cmd1 → cmd2 → cmd3 → … → prog
Il padre di tutti I processi raccoglie l'output e lo stampa a schermo.

Esempio:

./prog “ps ufax” “grep gabriele” “sort”

deve avere lo stesso output di:

ps ufax | *grep gabriele* | *sort*

./prog “ps ufax” “gawk '{printf $2"\n"}' ” “sort -n”

deve avere lo stesso output di:

ps ufax | *gawk '{printf $2"\n"}'* | *sort -n*

Specifiche:

- I messaggi di log dei *processi* devono comparire nella cartella */var/log/ipc* all’interno dei file *ipc.log*

Note:

- E' obbligatorio l'utilizzo di dup/dup2
- Obbligatorio usare la system call *getopt*
- Si possono usare solo le system call, all’interno del codice sorgente non e’ ammesso l’utilizzo di comandi gia’ disponibili nel sistema GNU/Linux in forma di codice oggetto compilato
- Tutti i processi generati non devono produrre processi zombie
- Ciascuna stringa di log deve essere caratterizzata da data e ora.

Progetto 4: Shell

Il progetto consiste nell'implementazione di una shell i cui comandi vengono eseguiti con priorita' dinamiche. La priorita (P) del comando e' funzione del tempo, i.e., P(t) = f(t).
I comandi della shell vengono eseguiti (di default) in background e il loro output compare sia su un file che a video.
Il processo shell controlla periodicamente (con un tempo T) lo stato di tutti i comandi in esecuzione e ne diminuisce la priorita' in accordo alla funzione f(t) = A + B * exp(-C*t), dove i parametri A, B e C devono essere scelti in maniera opportuna.

Esempio di esecuzione:

Shell -T 10

Specifiche:

- L'output dei comandi deve comparire su un file di log, i.e., /var/log/shell.log con gli eventuali tempi/date di esecuzione.
- La scelta dei parametri A, B e C deve essere opportunamente motivata nella relazione finale.
- Deve essere prevista l'esecuzione con parametri di default, i.e., ./shell