

Prova Pratica di Laboratorio di Sistemi Operativi
29 maggio 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma testeventfd che faccia uso della system call eventfd.

In particolare il programma deve eseguire una fork, quando l'utente digita un numero letto dal processo padre, il processo figlio deve stampare un numero uguale di x. (negli esempi e' riportato in grassetto cio' che l'utente digita).

```
$ testeeventfd  
3  
x  
x  
x  
2  
x  
x
```

Esercizio 2: completamento (10 punti)

Scrivere un programma testeventalt che faccia uso di due descrittori aperti con la system call eventfd.

In particolare il programma deve eseguire una fork e fare in modo che i due processi padre e figlio leggano alternativamente da standard input e riportino in output quanto letto (indicando quale processo sta facendo l'azione)

```
$ testeeventalt  
ciao  
padre: ciao  
mare  
figlio: mare  
sistemi  
padre: sistemi  
operativi  
figlio: operativi
```

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno script bash che conti l'ampiezza totale in byte di tutti i file di una directory divisi per suffisso (i.e. Si considera suffisso tutto cio' che segue il primo carattere 'punto' nel filename). I file privi di suffisso vengono ignorati.

Es:

```
$ suffixlen dir  
.c: 44320  
.o: 123000  
.tar.gz: 555333  
.jpg: 44332
```

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
20 giugno 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma che stampi il numero di inode di ogni file presente in una direcotry passata come argomento (o della direcotry corrente se il programma viene chiamato senza parametri) e stampi l'elenco in ordine crescente di numero di i-node.

Es:

```
$ lsino demo  
demo/. 1972484  
demo/.. 1971834  
demo/1.c 1972528  
demo/a.out 1972485  
demo/l1 1972486  
demo/l2 1972486  
demo/l3 1972486  
demo/link.c 1972528
```

Esercizio 2: completamento (10 punti)

Si estenda l'esercizio 1 e lo si trasformi in un programma che cerchi in una directory la presenza di link (fisici).

Per ogni file avente piu' nomi all'interno della directory deve stampare una riga contenente l'elenco dei nomi che fanno riferimento allo stesso file. (suggerimento: piu' nomi fanno riferimento allo stesso file se corrisponde in numero di i-node del file)

```
$ lslink demo  
demo/1.c demo/link.c  
demo/l1 demo/l2 demo/l3
```

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno script bash per cercare all'interno della directory corrente tutti i file che abbiano lo stesso contenuto.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
18 luglio 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Una gara fra processi.

Scrivere un programma C che prenda in input (da standard input) alcune righe contenenti comandi con relativi parametri.

La lista dei comandi termina quando viene inserita una riga vuota.

A questo punto tutti i comandi vengono eseguiti in modo concorrente e deve venir stampato l'ordine di terminazione.
(E' vietato usare system, popen o simili)

Es.

```
$ prorace
sleep 30
sleep 10
sleep 20
ls -l
      input di una riga vuota
... output di ls -l ...
1 arrivato: ls -l
2 arrivato: sleep 10
3 arrivato: sleep 20
4 arrivato: sleep 30
```

Esercizio 2: completamento (10 punti)

Si estenda l'esercizio 1 in modo da avere una partenza piu' sincronizzata dei processi.

Tutti i processi devono attendere un evento che li sblocchi tutti insieme, per esempio tutti i processi possono mettersi in attesa con poll o select sul canale di lettura di una pipe. Quando “prorace” scrive un carattere sulla pipe tutti i processi diventano ready.

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno script bash che cataloghi per nome tutti i file presenti in un sottoalbero.

nameindex dir1 dir2

crea nel file dir2 un link simbolico ad ogni file contenuto nell'albero che ha come radice dir1. Quando sono presenti piu' file con lo stesso nome in diverse directory all'interno di dir1, viene posto un suffisso numerico al nome del link simbolico
se dir1 contiene i file pippo, pluto e paperino e la directory paperopoli e quest'ultima contiene i file paperino, qui quo e qua, all'interno di dir2 divranno essere contenuti i seguenti link simbolici:

```
pippo → dir1/pippo
pluto → dir1/pluto
paperino → dir1/paperino
paperino1 → dir1/paperopoli/paperino
qui → dir1/paperopoli/qui
quo → dir1/paperopoli/quo
qua → dir1/paperopoli/qua
```

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

13 settembre 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Lo scopo del programma che dovrete scrivere e' di confrontare fra loro i file di una directory, se ne trovate due (o piu') che hanno lo stesso contenuto dovete unificarli. Alla fine del processo l'elenco dei file della directory deve rimanere invariato ma i nomi dei file che avevano lo stesso contenuto devono essere link fisici che indicano lo stesso file.

In questo esercizio si richiede che l'intero contenuto dei file venga confrontato.

Esercizio 2: completamento (10 punti)

Si migliori l'efficienza dell'esercizio 1.

Prima di procedere al confronto si calcoli una chiave hash del contenuto di ogni file e si crei una struttura dati opportuna contenente tutte le chiavi hash dei file.

Il confronto del contenuto dei file deve avvenire solo fra i file che hanno identico valore di hash.

Esercizio 3: Script bash o Python: (10 punti):

Sia data una directory che contiene file di testo.

Scopo dell'esercizio e' di contare i caratteri delle corrispondenti righe di testo di tutti i file della directory, si vuole cioe' sapere il numero totale di caratteri presenti nelle prime righe di tutti i file, nelle seconde linee, ecc.

\$ ccpl mydir

1 234

2 21

3 333

.....
l'output significa che se contiamo tutti i caratteri contenuti nella prima riga di tutti i file in mydir otteniamo 234 (mydir/file1 puo' avere 40 caratteri nella prima riga, mydir/file2 ne puo' avere 20, ecc... procedendo per tutti i file di mydir la somma fa 234).

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
23 gennaio 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma in C “linean” che prenda come parametro il pathname di un file e un numero intero (che chiameremo n). Il programma deve stampare come output il numero di caratteri presenti nella n-ma riga del file se il file e’ un file regolare di testo, non deve stampare nulla negli altri casi. Un file viene considerato di testo se tutti i suoi byte hanno valori compresi nel range 1-127. Per controllare se il file e’ “regolare” usare la system call lstat.

Esercizio 2: completamento (10 punti)

Si scriva un programma C chiamato “lineandir”. Il risultato del programma, stampato su standard output, deve essere un solo numero intero: la somma del numero di caratteri presenti nelle n-me righe di tutti i file regolari, di testo, non nascosti (il primo carattere deve essere diverso da punto) della directory corrente.

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovrete implementare come script shell o programma python e’ updatedir. Updatedir prende due directory come parametri.

updatedir dira dirb

deve copiare in dirb tutti i file regolari che sono in dira e non in dirb. Se un file regolare e’ presente con lo stesso nome sia in dira sia in dirb, il file deve essere copiato dalla dira alla dirb solo se i contenuti differiscono.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
20 febbraio 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma in C “colonnan” che prenda come parametro il pathname di un file e un numero intero (che chiameremo n). Il programma deve stampare come output il numero di caratteri presenti nella n-ma colonna del file se il file e' un file regolare di testo, non deve stampare nulla negli altri casi. Un file viene considerato di testo se tutti i suoi byte hanno valori compresi nel range 1-127. Per controllare se il file e' “regolare” usare la system call lstat.

Esercizio 2: completamento (10 punti)

Si scriva un programma C chiamato “coloninandir”. Il risultato del programma, stampato su standard output, deve essere un solo numero intero: la somma del numero di caratteri presenti nelle n-me colonne di tutti i file regolari, di testo, non nascosti (il primo carattere deve essere diverso da punto) della directory passata come parametro, ovvero della directory corrente se coloninandir viene lanciato senza specificare parametri.

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovrete implementare come script shell o programma python e' linkdir. linkdir prende due directory come parametri.

linkdir dira dirb

e deve creare in dirb un link fisico (non simbolico) a tutti i file regolari che sono in dira e non in dirb. Se un file regolare e' presente con lo stesso nome sia in dira sia in dirb, nella directory dirb deve rimanere il file originariamente presente se e' piu' recente di quello in dira altrimenti un link al file di dira con lo stesso nome.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

29 maggio 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma con un solo parametro.

Come prima cosa il programma deve creare una directory con il path specificato nel parametro. Se la directory esiste già o si verifica un errore nella creazione, il programma deve terminare. Chiameremo questa directory “directory-base”

Il programma usando inotify rimane in attesa e stampa una riga di log per ogni file o directory creato o cancellato nella directory-base. (solo nella directory-base, non nelle sottodirectory).

Quando viene cancellata la directory-base il programma termina.

Esercizio 2: completamento (10 punti)

Si estenda il programma dell'esercizio 1 per operare anche nelle sottodirectory. Quindi il programma “dovrebbe” stampare una riga di log per ogni file o directory creata o cancellata in tutto il sottoalbero che ha nella directory-base la radice.

Nota: se necessario, usate strutture dati molto semplici come vettori o liste semplici, non preoccupatevi dell'efficienza.

In realtà per un problema nel design dell'API inotify, alcuni eventi di creazione di directory nidificate troppo vicini nel tempo possono venir perduti.

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovete implementare come script shell o programma python è statlen.

Data una directory statlen fa una statistica sulla lunghezza dei nomi dei file presenti in tutto il sottoalbero con radice nella directory passata come parametro.

es.

```
$ statlen /tmp  
2: 2  
3: 10  
5: 4
```

...

significa che in tmp (e in tutte le sottodirectory di /tmp) ci sono 2 file con nome di due caratteri, 10 con nomi di 3 caratteri e così via.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
17 giugno 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Occorre scrivere due programmi: mytar myuntar.

Mytar prende come parametro il nome di una directory e il nome di un file:

mytar ddd ddd.mytar

mytar crea il file indicato come secondo parametro e registra in esso tutti i file regolari presenti nella directory (ddd nell'esempio). Ogni file e' registrato nel secondo il seguente formato:

nome del file (stringa ASCII terminata da un byte 0)

lunghezza del file (stringa ASCII terminata da 0, la lunghezza e' registrata come stringa in rappresentazione in base 10 per non avere problemi di endianess e di ampiezza dei campi)

contenuto del file (un numero di byte corrispondente alla lunghezza indicata sopra).

Myuntar fa l'operazione inversa:

myuntar ddd.mytar newddd

crea la directory indicata come secondo parametro e, in essa, tutti i file registrati in ddd.mytar.

Per provare i due programmi, al termine dell'esecuzione di due comandi simili a quelli degli esempi mostrati qui sopra per mytar e myuntar, tutti i file regolari presenti in ddd devono esistere in newddd e devono avere tutti lo stesso contenuto.

Se create una directory ddd contenente solo file regolari l'output di “diff -R ddd newddd” deve essere vuoto.

Esercizio 2: completamento (10 punti)

Completate l'esercizio 1 inserendo nella codifica dei file la gestione dei permessi di accesso e la scansione ricorsiva delle directory. Si lascia al candidato ogni decisione in merito alla codifica e ai parametri dei programmi.

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovete implementare come script shell o programma python e' maxfreq.

Maxfreq ha come parametro un carattere alfanumerico e una directory.

Es:

maxfreq q mydir

Cerca in tutto il sottoalbero del file system originato da mydir il file che ha la maggior frequenza della lettera indicata (in questo caso la maggior frequenza di 'q'). Fornisce in output il nome del file e la frequenza in percentuale.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
02 luglio 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

prima parte:

Scrivere un programma che elenchi tutti i file di una directory.

mytx ddd ddd.tx

Ogni riga del file di output (secondo parametro) deve contenere la lunghezza, uno spazio e il nome del file. Dopo l'ultima riga deve inserire una riga bianca.

ddd.t2 deve contenere l'elenco dei file regolari. Il primo campo e' un numero intero seguito da uno spazio, tutto cio' che segue fino alla fine riga e' il nome del file.

es.

12 file1

235 file di prova

Seconda parte:

dopo la riga bianca inserire nel file di output (ordinatamente) il contenuto di tutti i file.
(quindi nel caso sopra i 12 byte di file1 seguiti dai 235 di “file di prova”).

Esercizio 2: completamento (10 punti)

fare un programma che consenta di recuperare un file dal formato generato dall'esercizio1.

demytx file1 ddd.tx

deve creare il file 'file1' recuperando il contenuto dal file generato da myt2 dell'esercizio1

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovete implementare come script shell o programma python e' maxfreq.

Maxfreq ha come parametro un carattere alfanumerico e una directory.

Es:

maxfreq q mydir 10

Cerca in tutto il sottoalbero del file system originato da mydir i 10 file che hanno la maggior frequenza della lettera indicata (in questo caso la maggior frequenza di 'q'). Fornisce in output i nomi dei file e le frequenze in percentuale.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
17 luglio 2014

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Non usare system o popen o simili!

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma che per ogni file .c nella directory corrente chiama il compilatore gcc per generare il file oggetto (.o) a meno che esista già nella directory un file oggetto relativo allo stesso sorgente che sia più nuovo (ultima modifica) del sorgente.

Tutti i parametri devono essere passati al compilatore.

Es:

genobj -I . -ggdb

se nella directory corrente esistono I file uno.c e due.c e il file due.o deve richiamare

gcc -I. -ggdb -c uno.c

e, solo se due.o ha ultima modifica precedente a due.c, deve chiamare

gcc -I. -ggdb -c due.c

Esercizio 2: completamento (10 punti)

Scrivere un programma genexe che generi l'eseguibile a partire da tutti i file oggetto della directory corrente.

genexe prog -lm

il primo parametro è il nome dell'eseguibile da generare, tutti gli altri parametri devono essere passati al gcc.

Prima di generare l'eseguibile genexe deve controllare che tutti i file oggetto siano più recenti (ultima modifica) del relativo sorgente (se esiste). In caso contrario non si deve generare l'eseguibile ma terminare con un output di errore:
“run genobj first”

Esercizio 3: Script bash o Python: (10 punti):

Il comando che dovete implementare come script shell o programma python è mytx.

Tale comando elenca tutti i file di una directory.

mytx ddd ddd.tx

Ogni riga del file di output (secondo parametro) deve contenere la lunghezza, uno spazio e il nome del file. Dopo l'ultima riga deve inserire una riga bianca.

ddd.t2 deve contenere l'elenco dei file regolari. Il primo campo è un numero intero seguito da uno spazio, tutto ciò che segue fino alla fine riga è il nome del file.

es.

12 file1

235 file di prova

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi **25 settembre 2014**

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Non usare system o popen o simili!

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma che preso come parametro un file contenente un elenco di comandi (con I relativi parametri) li attivi tutti in esecuzione concorrente e rimanga in attesa. Quando il primo termina, vengono terminati (con segnale SIGTERM) tutti gli altri. (consiglio: puo' essere utile usare la mia libreria s2argv-exec)

esempio:

```
wta commands
```

il file commands contiene:

```
./ttest 40  
./ttest 10  
./ttest 20
```

lo script ./ttest contiene:

```
#!/bin/bash  
echo waiting for $1 seconds  
sleep $1  
echo done $i
```

l'output sara':

```
waiting for 40 seconds  
waiting for 10 seconds  
waiting for 20 seconds  
done 10
```

e poi basta perche' gli altri processi verranno terminati.

Esercizio 2: completamento (10 punti)

Completare wta in modo che l'output di ogni comando venga salvato separatamente e solo l'output del processo terminato per primo venga mostrato.

Nell'esempio di prima l'output dovrà essere:

```
waiting for 10 seconds  
done 10
```

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno scrip bash che faccia un backup storico di un file.

backn file n

deve mantenere n versioni del file specificato. n>2

Esempio:

```
backn myfile 10
```

se esiste myfile.9 deve essere rinominato in myfile.10

se esiste myfile.8 deve essere rinominato in myfile.9

e cosi' via fino a myfile.2, rinominato myfile.3.

ora se myfile.1 ha lo stesso contenuto di myfile.3, myfile diventa un link fisico a myfile.2

myfile viene copiato in myfile.1

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi **21 gennaio 2015**

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Non usare system o popen o simili!

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma C di nome filepipe che abbia come unico parametro il pathnae di un file di testo.

Questo file contiene due comandi con i rispettivi parametri, uno per riga.

Il programma deve mettere in esecuzione concorrente i due comandi in modo che l'output del primo venga fornito come input del secondo usando una pipe.

Il programma deve terminare quando entrambi i comandi sono terminati.

Esempio: se il file ffff contiene:

```
ls -l  
tac
```

il comando:

```
filepipe ffff
```

deve restituire lo stesso output del comando:

```
ls -l | tac
```

Esercizio 2: completamento (10 punti)

Il secondo esercizio estende il primo. Il file passato come parametro in questo caso ha un numero arbitrario di righe e non solo due come nell'esercizio1. Ogni riga contiene un comando con i rispettivi parametri. Il programma deve attivare tutti i comandi del file in modo concorrente, e terminare quando tutti finiscono.

L'output di tutti i comandi, tranne l'ultimo, deve essere fornito in input all'ultimo comando (quello nell'ultima riga significativa del file).

Es: se il file contenesse:

```
ls -l  
ls -l  
ls -l  
cat
```

l'output dovrebbe essere la lista dei file della corrente directory 3 volte (nell'output puo' capitare che righe di comandi diversi vengano frapposte, l'accesso alla pipe e' conteso).

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno script bash che scandisca il sottoalbero relativo alle directory passate come parametri (o alla direcotry corrente se non ci sono parametri) e fornisca in output l'elenco dei file che hanno la stessa somma MD5 (i.e. l'output del comando md5sum).

In output ogni riga deve mostrare un elenco di pathname relativi a file che hanno la stessa somma MD5 (che quindi sono molto molto probabilmente uguali).

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

30 maggio 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (25 punti) Scrivere due programmi in modo che i parametri passati al primo vengano stampati dal secondo. La comunicazione deve avvenire tramite una shared memory realizzata con la chiamata POSIX shm_open, la sincronizzazione tramite segnali. Viene attivato per primo il programma ricevente. (anche i pid dei processi possono venir scambiati attraverso la shared memory!).

Es: scrivere in un terminale:

 \$./receiver

l'esecuzione di “receiver rimane in attesa”. in un secondo terminale scrivere:

 \$./sender a bb ccc

nel primo deve comparire

 ./sender

 a

 bb

 ccc

Esercizio 2 Script bash o Python: (10 punti): Dato il pathname di una directory passato come parametro, lo script bash o python deve fornire l'elenco dei nomi di file presenti nel sottoalbero in ordine alfabetico specificando per ogni nome la sottodirectory dove si puo' trovare il file. Possono esistere anche file con lo stesso nome in diverse sottodirectory. In questo caso occorre elencare tutte le directory.

Esempio: Se la directory *d* contiene i file *afilea* e *fiore* e le directory *b* e *qq*, la direcotry *b* contiene *bfile*, la directory *qq* contiene il file *ccc* e la directory *settete* e infine la directory *settete* contiene *ccc*, l'output sara' il seguente.

```
$ elencofile d
afilea  /
bfile   /b
ccc     /qq
ccc     /qq/settete
fiore   /
```

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

22 giugno 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (25 punti) Scrivere un programma che abbia come primo parametro un numero di segnale e come successivi parametri un comando coi rispettivi argomenti.

Ad esempio

sigstart 10 xclock -update 1

sigstart deve rimanere in attesa e lanciare una istanza del comando ogni volta che riceve un segnale del tipo indicato (in questo caso SIGUSR1=10).

Esercizio 2 Script bash o Python: (10 punti): Scrivere uno script bash o un programma python che metta in ordine tutti i file di un sottoalbero dal piu' vecchio al piu' recente.

La stampa finale deve mostrare solamente il path relativo dei file.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

22 luglio 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (25 punti) Scrivere il programma *lanciatutto* che esegua in modo concorrente tutti i file “eseguibili” della directory corrente passando ad ognuno gli stessi parametri (quelli usati nell’invocazione di lanciatutto).

NB: eseguibili=che possono essere eseguiti, script o binari

modo concorrente=il programma attiva tutti un processo per ogni file eseguibile (senza aspettare che termini il precedente per lanciare il successivo).

Il programma deve terminare quando tutti i processi attivati sono terminati.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria “system” e “popen”.

Esercizio 2 Script bash o Python: (10 punti):

Siano date due sottoalberi del file system (il programma e' pensato per due versioni di una gerarchia sorgente).

Lo script deve cercare i file .c e .h presenti nei due sottoalberi evidenziando:

- quali sono presenti in un solo sottoalbero
- quali sono presenti in entrambi i sottoalberi ma hanno contenuto differente.

Es:

```
cmpsource so.1.0 so.1.1
so.1.0/file.c not in so.1.1
so.1.0/p/q/search.c so.1.1/p/q/search.c differ
so.1.1/h/head.h not in so.1.0
```

(L’ordine delle segnalazioni in output non e’ importante).

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

12 settembre 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (20 punti)

Siano date due directory del file system.

Il programma deve cercare i file .c e .h presenti nelle due directory evidenziando quali sono presenti in una sola directory

Es:

```
$ cmpsource so.1.0 so.1.1  
so.1.0/file.c not in so.1.1  
so.1.1/head.h not in so.1.0
```

(L'ordine delle segnalazioni in output non e' importante).

Esercizio 2: Linguaggio C opzionale: (10 punti)

Controllare anche quali file .c o .h sono presenti in entrambe le directory ma hanno contenuto differente.

L'output dell'esempio deve contenere nel caso linee come:

```
so.1.0/search.c so.1.1/search.c differ
```

Esercizio 2 Script bash o Python: (10 punti):

Scrivere uno script bash o un programma python *lanciatutto* che esegua in modo concorrente tutti i file “eseguibili” della directory corrente passando ad ognuno gli stessi parametri (quelli usati nell'invocazione di lanciatutto).

NB: eseguibili=che possono essere eseguiti, script o binari

modo concorrente=il programma attiva tutti un processo per ogni file eseguibile (senza aspettare che termini il precedente per lanciare il successivo).

Il programma deve terminare quando tutti i processi attivati sono terminati.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi **20 gennaio 2012**

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma zerocopy che prenda come parametro (obbligatorio) il nome di un file. Alla fine il file passato come parametro deve mantenere il proprio contenuto ma le aree inutilizzate (blocchi da 1K completamente a 0) devono essere non allocati. Per provare il vostro programma copiate il file /public/holefile

```
$ cp /public/holefile holefile
```

```
$ ls -ls holefile
```

```
200 -r--r-- 1 renzo renzo 204800 2012-01-19 18:56 holefile
```

occupa 200 blocchi, ma e' un file “sparso” ha ampie parti nulle.

```
$ ./zerocopy holefile
```

```
$ diff /public/holefile holefile
```

nessun output, sono identici!

```
$ ls -ls holefile
```

```
136 -rw----- 1 renzo renzo 204800 2012-01-19 19:11 holefile
```

Occupava solo 136 blocchi.

(Consiglio: copiare il file blocco per blocco in un file temporaneo, saltando la scrittura delle parti nulle, e poi alla fine rinominare il file temporaneo col nome del file di partenza).

Esercizio 2: Lingaggio C opzionale: (10 punti)

Estendere la soluzione dell'esercizio 1: usare la funzione getoptlong per prendere come parametro opzionale -s per specificare la lunghezza del blocco da prendere in esame.

Es:

```
$ ./zerocopy -s 1024 holefile
```

deve essere equivalente al programma lanciato senza parametri.

```
$ ./zerocopy -s 10240 holefile
```

deve usare blocchi da 10K

Esercizio 2 Script bash o Python: (10 punti):

Guardate la directory /public/nbatch. Contiene alcuni script che stampano il proprio nome e attendono un secondo. Lanciatene uno per prova. Scrivere uno script bash o un programma python che esegua tutti gli script in ordine crescente numerico. Gli script che hanno lo stesso numero devono essere eseguiti in concorrenza:

L'output deve essere:

```
$ ./countrun /public/nbatch
```

```
./00a
```

... attende un secondo

```
./01b
```

```
./01c
```

```
./01d
```

... attende un secondo

```
./02f
```

```
./02e
```

... attende un secondo

```
./10g
```

... attende un secondo

```
./20i
```

```
./20h
```

... attende un secondo

(l'ordine degli output dei programmi con lo stesso numero puo' cambiare).

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi **15 febbraio 2012**

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Guardate la directory /public/nbatch. Contiene alcuni script che stampano il proprio nome e i parametri e attendono un secondo. Lanciatene uno per prova. Scrivere un programma C che esegua tutti gli script in ordine crescente numerico. Gli script che hanno lo stesso numero devono essere eseguiti in concorrenza passando a tutti gli stessi parametri:

L'output deve essere:

```
$ ./countrun /public/nbatch -a -b -c  
./00a -a -b -c  
... attende un secondo  
./01b -a -b -c  
./01c -a -b -c  
./01d -a -b -c  
... attende un secondo  
./02f -a -b -c  
./02e -a -b -c  
... attende un secondo  
./10g -a -b -c  
... attende un secondo  
./20i -a -b -c  
./20h -a -b -c  
... attende un secondo
```

(l'ordine degli output dei programmi con lo stesso numero puo' cambiare).

NOTA: e' vietato usare le funzioni system o popen.

Esercizio 2: Linguaggio C opzionale: (10 punti)

Estendere la soluzione dell'esercizio 1: l'output oltre a venir visualizzato deve essere salvato in un file nella directory corrente avente come nome il numero d'ordine degli script.

L'output del nuovo comando:

```
$ ./countrunx /public/nbatch -a -b -c
```

deve essere identico a quello di prima ma al termine devono esistere 5 file chiamati 00, 01, 02, 10, 20.

```
$ cat 00  
./00a -a -b -c  
$ cat 01  
./01b -a -b -c  
./01c -a -b -c  
./01d -a -b -c  
ecc.
```

Esercizio 3: Script bash o Python: (10 punti):

Creare uno script o un programma python in grado di creare file con contenuti casuali e sequenze di byte nulli: l'eseguibile risultante deve avere un numero variabile di parametri, il primo e' il nome del file, il secondo e' l'ampiezza del file, i successivi indicano l'inizio e la lunghezza delle sequenze nulle. I dati sono espressi in numero di blocchi da 4K. Per esempio:

\$nullfile nul1 100 2:1 10:4

deve creare un file nul1 di 400K (409600) e ha 5 blocchi interamente nulli: il terzo (dal byte 8192 al byte 12287) e quelli dall'undicesimo al quattordicesimo (dal byte 40960 al byte 57343).

Hints: per creare contenuti casuali si puo' leggere il device /dev/urandom, per inserire zeri si puo' usare il device /dev/zero. Per chi vuole scrivere uno shellscript si consiglia la lettura del manuale dd.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
15 febbraio 2012

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (24 punti)

Scrivere un minimake, cioe' una versione minimale del programma make.

Questo programma legge il file minimakefile nella directory corrente che ha il seguente formato:

target: comando

Il minimakefile e' scritto in modo che se le dipendenze sono generate da regole del minimakefile queste vengano definite prima. Ad esempio il minimakefile per compilare ed eseguire il programma hw.c (hello world) sara':

hw.o: gcc -c hw.c

hw: gcc -o hw hw.o

run: ./hw

minimake deve scorrere il minimakefile, se il file indicato come target non esiste esegue il comando (e' vietato l'uso di popen, system e simili!):

Esercizio 2: completamento minimake (10 punti)

la sintassi del minimakefile per l'esercizio2 deve prevedere le dipendenze come il vero makefile.

target: dependence1 dependence2

 comando

 comando

(come per il makefile il target viene scritto a inizio linea, i comandi hanno un tab come primo carattere)

es per compilare hw.c.

hw.o: hw.c

 gcc -c hw.c

hw: hw.o

 gcc -o hw hw.o

run: hw

 hw

in questo caso occorre eseguire i comandi che seguono la regola di dipendenza solo se il file target non esiste o se un file indicato come dipendenza e' stato modificato piu' recentemente del target.

Esercizio 3: Script bash o Python: (6 punti):

Lo script o il programma python prende il nome di una directory come parametro e deve listare i file nella directory che hanno un file corrispondente con nome scritto al contrario (il file ai lati deve essere nella lista solo se nella directory c'e' anche italia, i palindromi devono essere listati una sola volta).

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
20 giugno 2012

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (24 punti)

Scrivere un programma chiamato spy che tenga sotto controllo una directory (il cui pathname viene passato come unico parametromo), e segnali, stampandone il nome, ogni file che viene creato in tale directory.

Si faccia uso della interfaccia inotify (leggere la pagina di manuale).

Attenzione: il buffer per gli eventi deve avere dimensione superiore a quella della struttura inotify_event altrimenti non c'e' spazio per il campo name.

Esercizio 2: completamento spy (6 punti)

Si completi l'esercizio 1: se il nuovo file e' eseguibile, il file deve essere eseguito e cancellato. (e' sempre vietato l'uso di system, popen e simili).

Esercizio 3: Script bash o Python: (10 punti):

Lo script o il programma python da realizzare deve fornire l'occupazione totale in byte dei file della directory corrente che corrispondono alla espressione regolare passata come parametro.

e.s. sizere 'l[az]x'

444

significa che la somma delle ampiezze dei file di tre lettere alfabetiche minuscole che iniziano per l finiscono per x, e' di 444 byte.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi
17 luglio 2012

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Il programma lancian deve far attivare un programma in n copie **concorrenti**.

es. lancian 10 ls -l

deve lanciare 10 copie di “ls -l”.

Per ogni copia la variabile di ambiente (environment) NCOPIA deve contenere un numero diverso (per la prima copia vale 0 e così' via fino al numero di copie -1).

il programma lancian deve finire solo quando tutte le copie hanno terminato l'esecuzione.

Esercizio 2: completamento (10 punti)

lancian deve essere completato in modo che lo standard output di tutti le copie venga unificato ed emesso come standard output del programma lancian, in modo ordinato: l'intero output della prima istanza (NCOPIA==0) deve precedere quello della seconda e così' via.

Esercizio 3: Script bash o Python: (10 punti):

Lo script o il programma Python deve fornire una lista dei file all'interno di un sottoalbero ordinati secondo il la “profondità” nell'albero (prima tutti quelli nella radice del sottoalbero, poi tutti quelli al secondo livello), in ordine alfabetico fra quelli allo stesso livello.

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

19 settembre 2012

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma cpinout che copi i dati letti dal file standard input nel file standard output e che:

- al ricevimento del segnale SIGUSR1 indichi sul file standard error quanti byte sono stati copiati al momento attuale
- mostri la propria attivita' sovrascrivendo un carattere (stampando –, \, |, / e poi ritornando a – per dare l'impressione di un'asta che gira).

Provate il programma con un comando come:

```
cat /dev/zero | ./cpinout | cat >/dev/null
```

da un altro terminale poi mandate segnali con kill -USR1 xxxx, dove xxx e' il numero del processo cpinout

Esercizio 2: completamento (10 punti)

Completare l'esercizio 1:

- ogni secondo deve indicare i byte trasferiti (sempre su stderr)
- alla fine deve calcolare la media di dati trasferiti (byte totali / tempo di esecuzione).

Per provare:

```
dd if=/dev/zero count=1000000 bs=1000 status=noxfer | ./a.out | cat >/dev/null
```

Esercizio 3: Script bash o Python: (10 punti):

Scrivere uno script bash o un programma python che presi in input un file e una directory fornisca la lista dei file nella directory e in tutto il sottoalbero generato che hanno lo stesso tipo del file (cioe' tutti i file per i quali il comando "file" fornisca lo stesso output)

es:

```
$lsametype file.c /tmp/test
```

```
f1.c
```

```
helloworld.c
```

(sono tutti di tipo “C source, ASCII text”)

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

25 gennaio 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma listexe che fornisca in output l'elenco dei processi attivi nel sistema mettendo in output per ogni processo il pid e il path dell'eseguibile.

L'informazione puo' essere trovata scandendo la directory proc, infatti ad ogni processo attivo corrisponde una directory in /proc che ha come nome il numero del processo (ad esempio al processo 9801 corrisponde la directory /proc/9801) e all'interno di queste directory il file exe e' un link simbolico all'eseguibile.

Esempio:

```
$ ls -l /proc/9801/exe  
lrwxrwxrwx 1 renzo renzo 0 Jan 22 18:26 /proc/9801/exe -> /bin/bash
```

l'output del programma listexe dovrebbe essere:

```
$ listexe  
.....  
9801 /bin/bash  
9933 /usr/bin/vim  
.....
```

(alcuni link simbolici possono essere non leggibili per sicurezza, verranno omessi).

Esercizio 2: completamento (10 punti)

svolgere l'esercizio 1 ma in output mettete in ordine alfabetico tutti gli eseguibili e per ognuno indicare i relativi numeri di processo (lo stesso eseguibile potrebbe essere in esecuzione da parte di molteplici programmi).

```
$ listexep  
...  
/bin/bash 9801 4332 2233  
/usr/bin/vim 2255 9933  
....
```

Esercizio 3: Script bash o Python: (10 punti):

Nella distribuzione Debian e nelle distro derivate, le directory /etc/rc0.d /etc/rc1.d, etc contengono link simbolici a file contenuti in /etc/init.d. Nelle directory /etc/rc?.d i link simbolici hanno nomi che iniziano con S o con K (a seconda se il servizio debba essere attivato o disattivato in quel livello di funzionamento)

Scrivere un programma python o uno script bash che per ogni file di /etc/init.d indichi in quali directory compare come file S e in quali come file K.

Es:

```
acpi-support K1 S2 S3 S4 S5  
(infatti /etc/rc1.d/K20acpi-support /etc/rc2.d/S99acpi-support /etc/rc3.d/S99acpi-support /etc/rc4.d/S99acpi-support  
/etc/rc5.d/S99acpi-support sono tutti link simbolici a /etc/init.d/acpi-support)
```

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

15 febbraio 2013

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1: Linguaggio C (obbligatorio): (20 punti)

Scrivere un programma C di nome lanciaxp che esegua in modo concorrente tutti i file eseguibili con come palindromo nella directory corrente.

Se la directory corrente contiene i file:

```
$ ls -l
total 28
-rwxr-xr-x 1 renzo renzo 21 Feb 10 11:10 anna
-rwxr-xr-x 1 renzo renzo 4851 Feb 10 11:12 bib
-rw-r--r-- 1 renzo renzo 82 Feb 10 11:12 bib.c
-rwxr-xr-x 1 renzo renzo 21 Feb 10 11:10 emma
-rw-r--r-- 1 renzo renzo 5 Feb 10 11:13 erre
-rwxr-xr-x 1 renzo renzo 21 Feb 10 11:10 laura
```

lanciaxp deve eseguire file anna, emma e bib. Al contrario non devono essere eseguiti laura (il nome non è palindromo), bib.c e erre (perché non sono eseguibili). Lanciaxp termina quando tutti i processi attivati sono terminati.

Esercizio 2: completamento (10 punti)

La scelta di quali file eseguire deve essere selezionabile con parametri:

-p: palindromo
-b: script (i primi due caratteri del file contengono la stringa “!#”).

-c: in modo concorrente. Se -c non è specificato viene lanciato un programma alla volta e si attende la terminazione prima di attivare il successivo

Quindi:

```
lanciax -p -c
ha lo stesso effetto del programma dell'esercizio 1
```

```
lanciax
lancia tutti gli eseguibili uno dopo l'altro
```

```
lanciax -b -p
esegue tutti gli script che hanno il nome palindromo (non il bib dell'esempio sopra), uno alla volta.
```

Esercizio 3: Script bash o Python: (10 punti):

Scrivere un programma python o uno script bash listexe che fornisca in output l'elenco dei processi attivi nel sistema mettendo in output per ogni processo il pid e il path dell'eseguibile.

L'informazione deve essere trovata scandendo la directory proc, infatti ad ogni processo attivo corrisponde una directory in /proc che ha come nome il numero del processo (ad esempio al processo 9801 corrisponde la directory /proc/9801) e all'interno di queste directory il file exe è un link simbolico all'eseguibile.

Esempio:

```
$ ls -l /proc/9801/exe
lrwxrwxrwx 1 renzo renzo 0 Jan 22 18:26 /proc/9801/exe -> /bin/bash
```

L'output del programma listexe dovrebbe essere:

```
$ listexe
.....
9801 /bin/bash
9933 /usr/bin/vim
.....
(alcuni link simbolici possono essere non leggibili per sicurezza, verranno omessi).
```

Esercizio 4: (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

12 febbraio 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (10 punti) Scrivere un programma C denominato “invarg” che esegua il programma passato come parametro invertendo gli argomenti.

Esempio:

```
invarg cat a b c  
deve avere l'effetto di  
cat c b a
```

Esercizio 2 (15 punti) Studiare il manuale della system call poll(2).

Scrivere un programma in linguaggio C che crei due named pipe (passate per argomento), le apra in lettura e e copi in standard output i dati via via disponibili da ogni pipe.

Test di funzionamento: aprire tre finestre di emulazione terminale. Nella prima lanciare: mergepipe p1 p2, nella seconda: cat >p1, nella terza cat>p2.

Tutto cio' che scriverete o nella seconda o nella terza finestra deve comparire nella prima.

Esercizio 3 Script: (10 punti)

Scrivere uno script “ls.” che faccia il listato dei file di una directory suddivisi per suffisso (in ordine alfabetico di suffisso, e fra i file con lo stesso suffisso in ordine alfabetico).

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chioccia.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

23 giugno 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (15 punti) Scrivere un programma in linguaggio C denominato “scriptexec” che venga richiamato con un solo parametro: il nome di un file che contiene un elenco di comandi con i rispettivi parametri, uno per riga.

Le righe che iniziano per '#' sono commenti.

Il programma esegue uno dopo l'altro i comandi presenti nel file.

(E' vietato l'uso di chiamate quali system o popen).

Esercizio2 (10 punti): Scrivere un programma (linguaggio C) che incrementi un contatore ogni secondo. Quando riceve un segnale SIGUSR1 il programma deve stampare il valore attuale del contatore.

Esercizio 3 Script: (10 punti)

Scrivere uno script bash che stampa l'elenco degli utenti di informatica (nome e cognome) in ordine di cognome, rielaborando l'output del comando ypcat. (Il cognome e' nel campo di informazione-GCOS, prima della virgola. Nella scrittura del file passwd i nomi o i cognomi composti sono un campo unico, le varie parti del nome o del cognome sono unite da underscore , es Enrico_Maria_Rossi_Forti).

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

14 luglio 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (20 punti) Scrivere un programma in linguaggio C denominato “stampallafine” che tramite la chiamata open_memstream (leggere il man!) salvi tutto cio’ che riceve da standard input in una stringa. Quando lo standard input termina, l’intera stringa deve essere copiata sullo standard output.
(Ricordate di definire la costante _GNU_SOURCE prima di includere stdio.h)

Esercizio 3 Script: (15 punti): scrivere uno script bash che scandisca tutto un sottoalbero del file system e cambi la data di ultimo accesso di tutti i file acceduti oggi ponendola a ieri. (provare il comando “date -d yesterday”, leggere il manuale di “touch”).

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chioccia.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

03 settembre 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (20 punti) Scrivere un programma mygroup che mostri la lista dei gruppi “supplementari” attualmente attivi. In altre parole deve comportarsi come il comando group. (hint: guardare la system call getgroups)
\$ mygroup
dialout cdrom floppy audio video scanner

Esercizio2: (5 punti) aggiungere all'esercizio precedente l'ordinamento alfabetico dell'elenco posto in output.

Esercizio 3 Script: (10 punti): Occorre scrivere uno script bash che dato un file e la sua precedente versione a.old aggiorni il file a su una macchina remota.

Es: remote-update /tmp/testfile amleto

Il file di testo /tmp/testfile deve essere presente nella macchina corrente e nella macchina remota (amleto nell'esempio). Nella macchina dove viene digitato il comando deve essere presente anche /tmp/testfile.old con contenuto identico a /tmp/testfile. Le modifiche fatte a /tmp/testfile devono essere riportate nel file presente nella macchina remota (senza copiare il file!). Usate diff, ssh e patch.

Per provare lo script controllate il nome della macchina libera di fianco alla vostra e usatela per l'esperimento. Mettete il file in /tmp altrimenti sarebbe condiviso.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

23 settembre 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (25 punti) Scrivere un programma che inverta i byte di un file: il file primo byte del file in output sia l'ultimo del file in input, il secondo in output sia il penultimo dell'input e cosi' via. Il programma deve operare su file di ogni dimensione usando un buffer di 4K byte.

(usare stat o fstat per leggere la dimensione del file, leggere blocchi da 4k con pread, invertirli localmente e scriverli nella posizione giusta nel nuovo file con pwrite).

Sintassi: inverti file_input file_output.

Se si vuole creare un file con 1MB di dati random per fare una prova usare:

```
dd if=/dev/urandom of=file_input bs=1024 count=1024
```

applicando inverti due volte al file casuale deve tornare il file dato.

Esercizio2: (5 punti): invertisulposto: come l'esercizio precedente ma il file di input e di output coincidono.

Usare un file temporaneo e rinominarlo alla fine.

Esercizio 3 Script: (10 punti): ypcat group fornisce il file dei gruppi, ypcat passwd e' il file degli utenti.

Fare uno script bash che elenchi quali sono gli utenti nominati nel file group che in realta' non esistono piu' (cioe' non appartengono al file passwd). Una prova effettuata il 22 settembre alle 23:54 ne conta 28.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

23 settembre 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (25 punti) Scrivere un programma che inverta i byte di un file: il file primo byte del file in output sia l'ultimo del file in input, il secondo in output sia il penultimo dell'input e cosi' via. Il programma deve operare su file di ogni dimensione usando un buffer di 4K byte.

(usare stat o fstat per leggere la dimensione del file, leggere blocchi da 4k con pread, invertirli localmente e scriverli nella posizione giusta nel nuovo file con pwrite).

Sintassi: inverti file_input file_output.

Se si vuole creare un file con 1MB di dati random per fare una prova usare:

```
dd if=/dev/urandom of=file_input bs=1024 count=1024
```

applicando inverti due volte al file casuale deve tornare il file dato.

Esercizio2: (5 punti): invertisulposto: come l'esercizio precedente ma il file di input e di output coincidono.

Usare un file temporaneo e rinominarlo alla fine.

Esercizio 3 Script: (10 punti): ypcat group fornisce il file dei gruppi, ypcat passwd e' il file degli utenti.

Fare uno script bash che elenchi quali sono gli utenti nominati nel file group che in realta' non esistono piu' (cioe' non appartengono al file passwd). Una prova effettuata il 22 settembre alle 23:54 ne conta 28.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

21 giugno 2010

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (20 punti) Scrivere un programma in linguaggio C che abbia come unico parametro il pathname di un file o di una directory.

Il programma, usando le recenti system call *inotify_init* e *inotify_add_watch* (descritte nella pagina di manuale inotify), deve scrivere “il file e' stato aperto” ogni volta che il file viene aperto e “il file e' stato modificato” se viene modificato. Il programma deve inoltre terminare quando il file viene cancellato.

Esercizio2: (10 punti): Scrivere un programma C che crei che risulti lungo 1 miliardo di byte ma che non occupi nessun blocco dati. Alla fine dell'esecuzione l'output del comando ls -sl del file deve essere:

```
ls -sl file  
0 -rw-r--r-- 1 user user 1000000000 Jun 20 21:00 file
```

Esercizio 3 Script: (10 punti): lo script deve creare una sottodirectory per ogni suffisso che compare nei file della directory corrente (e.g. Se sono presenti file a.pdf, b.odt, c.png, d.pdf, e.odt, f.pdf, g, h deve creare le directory .pdf .odt .png) e per ogni file con suffisso deve creare un link simbolico per poter accedere tutti i file con lo stesso suffisso a partire dalla directory relativa al suffisso. (nell'esempio precedente .pdf deve contenere tre link simbolici a.pdf, d.pdf, f.pdf che puntano ai corrispondenti file).

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

19 luglio 2010

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 Linguaggio C (obbligatorio): (20 punti) Scrivere un programma che crei un processo figlio. Il processo genitore e il processo figlio devono essere collegati da due pipe, una per la comunicazione genitore-figlio, una per la comunicazione figlio-genitore. Il processo figlio deve rispedire al genitore ogni messaggio ricevuto dalla pipe di input nell'altra (echo). Il processo genitore deve per 100000 volte spedire un messaggio di 40 byte al processo figlio e aspettare da questo che il messaggio venga rispedito.

Scopo dell'esercizio e' di verificare quanto tempo viene impiegato per il “ping” di 100000 messaggi su pipe.

Esercizio2: Linguaggio C (15 punti): Copiare l'esercizio precedente ma al posto delle due pipe utilizzare i messaggi dell'IPC di system V tramite le system call msgget, msgsnd, msgrcv.

Anche in questo caso misurare il tempo e verificare quale implementazione di IPC sia piu' veloce fra pipe e messaggi.

Esercizio 3 Script: (10 punti): Lo script deve prendere in input un file e invertire le righe pari con quelle dispari.

Se l'input e'

```
hello  
world  
goodbye  
moon
```

l'output deve essere

```
world  
hello  
moon  
goodbye
```

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

03 febbraio 2010

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (20 punti) Usando le chiamate opendir, readdir, closedir, qsort realizzare un programma in linguaggio C che metta in output la lista dei file della directory corrente in ordine alfabetico inverso.
(l'output deve essere equivalente a quello del comando “ls . | sort -r”).

Esercizio2: (5 punti): Scrivere in C un programma che stampi quanti file, quanti link simbolici, file speciali, quante directory sono presenti nella directory corrente.

Esercizio 3 Script: (10 punti): Scrivere uno script bash che svolga lo stesso compito del programma dell'esercizio 2.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo_chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

19 luglio 2010

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 Linguaggio C (obbligatorio): (20 punti) Scrivere un programma mergefifo che “misceli” l’input preso da due named pipe (passate come parametri) e dello standard input nello standard output. Il programma deve usare la system call poll.

Es:

```
mkfifo f1; mkfifo f2  
mergefifo f1 f2  
....
```

a questo punto tutto ciò che si scrive qui o da altri terminali in f1 o f2 deve comparire in output.

Esercizio2: Linguaggio C (15 punti):

Completare l’esercizio 1 con una gestione piu’ completa della linea comando (tramite getopt_long).

In particolare si vuole che siano gestiti:

```
mergefifo -h  
mergefifo --help  
che fornisce un semplice “usage”.
```

Il programma deve consentire di porre l’output in un file diverso dallo standard output tramite un parametro -o oppure --out seguito dal nome del file.

Esercizio 3 Script: (10 punti): Lo script deve produrre l’elenco dei file che sono stati modificati in due sottoalberi cioe’ per esempio:

```
diffy dir.v1 dir.v2
```

dove dir.v1 e dir.v2 sono due directory, deve produrre la lista dei file presenti nella stessa posizione relativa in dir.v1 e dir.v2 che hanno contenuto diverso.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e’ vietato l’uso delle funzioni di libreria “system” e “popen”.

Prova Pratica di Laboratorio di Sistemi Operativi

19 gennaio 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 Linguaggio C (obbligatorio): (20 punti) Scrivere un programma **redir** che facendo uso della getopt_long consente di eseguire un comando ridirezionando o no l'input e l'output.

Deve funzionare come segue:

redir ls -l	si comporta come “ls -l”
redir -o file ls -l	si comporta come “ls -l > file”
redir -i infile -o outfile sort	si comporta come “sort <infile >outfile”
redir -h	fornisce un semplice help

al posto dei parametri brevi devono poter essere usati i parametri

--in,--out,--help

Esercizio2: Linguaggio C (15 punti): il programma cathup si deve comportare come segue:

cathup file

copia cio' che riceve da standard input nel file file0 fino a quando non riceve un segnale SIGHUP, a quel punto chiude il file file0 e copia cio' che riceve da standard input sul file file1. Un nuovo SIGHUP deve di volta in volta fare in modo che l'output venga posto in file2 file3 e cosi' via.

Esercizio 3 Script: (10 punti): Scrivere uno script che faccia il merge (tramite link) dei contenuti due directory: newmerge a b c

La directory c deve contenere un link a tutti i file esistenti in a e in b. Nel caso un file con lo stesso nome compaia sia in a sia in b, il link creato in c deve riferirsi al file piu' recentemente modificato.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Prova Pratica di Laboratorio di Sistemi Operativi

15 febbraio 2011

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 Linguaggio C (obbligatorio): (25 punti) Scrivere un programma “riattiva” che lancia un comando con i propri parametri.

Riattiva ssh piripicchio.domain.it

Se il comando termina correttamente (i.e. Con exit status 0) l'esecuzione termina, altrimenti il comando viene rieseguito. Tutte le volte che il comando termina con exit status diverso da zero deve essere riattivato.

Esercizio 2 Script: (10 punti): Lo script bash deve elencare i file di una directory (passata per parametro, la directory corrente se non vengono passati parametri) divisi per tipo (come specificato dal comando 'file').

```
$ typels /tmp/test
ASCII text:
testo
bzip2 compressed data, block size = 900k:
    strace_4.5.20.orig.tar.bz2
gzip compressed data, extra field, from Unix:
    openssh_5.8p1.orig.tar.gz
gzip compressed data, from Unix, last modified:
    eglibc_2.11.2.orig.tar.gz
    strace_4.5.20-2.debian.tar.gz
gzip compressed data, from Unix, max compression:
    openssh_5.8p1-2.debian.tar.gz
PDF document, version 1.0:
    attestazione.pdf
PDF document, version 1.2:
    risPP.9dic03.pdf
    risparz.7nov03.pdf
$
```

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Nota importante: In tutti gli esercizi in C e' vietato l'uso delle funzioni di libreria "system" e "popen".

Esame di Laboratorio di Sistemi Operativi – 19/07/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Lettori / scrittori”) (20 punti)

Sia p un processo genitore, che crea N processi figli "lettori" e M processi figli "scrittori". Il compito del processo p è di arbitrare le letture e scritture dei processi figli, secondo le classiche regole del problema dei lettori/scrittori:

- sia nr il numero dei lettori che stanno accendo al database
- sia nw il numero di scrittori che stanno accedendo al database
- l'invariante è il seguente: $(nr > 0 \&\& nw==0) \parallel (nr == 0 \&\& nw <= 1)$

La vita dei processi lettori è la seguente: in un ciclo infinito, fanno richiesta di leggere il database, e restano in attesa di una conferma dal processo genitore. Dopo di che, fanno "finta" di leggere il database per un periodo casuale di secondi (tramite una sleep()), dopo di che fanno richiesta di rilasciare il database. La vita degli scrittori è simile, l'unica differenza è che fanno richiesta di scrivere il database e poi lo rilasciano.

Risolvere il problema tramite **pipe**. L'organizzazione dei meccanismi di comunicazione è lasciata alla vostra scelta.

Esercizio 2 ("Starvation?") (6 punti)

Risolvere il problema della starvation sia per i lettori che per gli scrittori.

Esercizio 3("??") (6 punti)

Si assuma di avere un file contenente un insieme di login name, ad esempio:

montreso
rossi
sacerdot
renzo

Scrivere uno script che esegua periodicamente le seguenti operazioni (ad esempio, ogni 30 secondi):

1. Verificare che non sia possibile accedere in nessun modo alla home degli utenti contenuti nell'elenco
2. Verificare che nella directory public non vi siano file appartenenti agli utenti contenuti nell'elenco

Nel caso una delle due condizioni non si verifichi per un utente, stampare un messaggio di warning.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresco_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-2a**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete sottoporvi alla discussione, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per discutere il vostro elaborato.

Esame di Laboratorio di Sistemi Operativi – 6/09/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

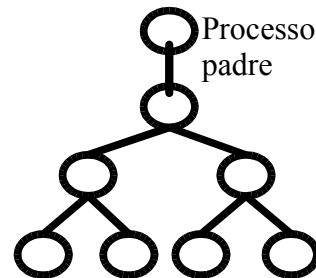
Esercizio 1 (“Pachinko”) (24 punti) (*)

Scrivete un programma che crei un albero binario di processi di altezza N. Si noti che il processo "padre originale" non fa parte dell'albero. Ad ognuno dei nodi foglia viene associato un valore numerico, scelto casualmente fra 0 e 1000.

Il processo padre genera un segnale, che spedisce al suo unico figlio. I processi dell'albero si comportano nel modo seguente:

nodi non foglia: aspettano un segnale dal genitore e inoltrano il segnale ad uno dei loro figlio, scelto casualmente
nodi foglia: stampano un messaggio che dice "hai vinto la quantità x!", dove x è il valore associato al nodo. Dopo di che, inviano un segnale di "risposta" al loro genitore, che viene propagato fino al nodo padre.

Una volta ricevuto il segnale di risposta, il nodo padre riprende a giocare da capo.



Esercizio 2 ("Grep a tre lettere") (8 punti)

Scrivere uno script che prende in input (da linea di comando) una stringa e una lista di file; lo script deve riportare il numero di occorrenze di tutte le sottosequenze di tre lettere della stringa nei file cercati. Ad esempio, se la stringa cercata è windows, lo script deve riportare un output del tipo:

```
win    7
ind   12
ndo   13
dow   14
ows    5
```

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochiocciola.cs.unibo.it o renzochiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-3**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete sottoporvi alla discussione, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per discutere il vostro elaborato.

(*) Per avere un'idea di cosa sia il Pachinko, date un'occhiata a <http://www.japan-guide.com/e/e2065.html> (non durante il compito!). Se andate in Giappone, andate in una sala Pachinko e date un'occhiata agli alienati che la frequentano. Al ritorno, i giocatori di video-poker vi sembreranno laureati al MIT....

Esame di Laboratorio di Sistemi Operativi – 24/09/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Matrix”) (18 punti)

Scrivere un programma per creare una griglia di NxN processi. Ogni processo deve essere identificato con una coppia di valori (x,y), dove $1 \leq x \leq N$ e $1 \leq y \leq N$.

La creazione della griglia deve avvenire in questo modo:

- il processo genitore crea la colonna "1" dei processi identificati da (1, y)
- ogni processo (x,y) crea il processo (x+1, y), con $1 \leq x \leq N-1$
- ogni processo stampa "(x,y)" generato da (w,z) al momento della propria creazione, dove (x,y) è l'identificatore del processo appena creato, (w,z) è l'identificatore del padre (per distinguerlo, il processo iniziale può assumere i valori (0,0))

Ovviamente queste regole influiscono sull'ordine in cui vengono stampati gli identificatori, per cui la correttezza del programma verrà valutata proprio su questo.

Esercizio 2 ("Quadrato magico", estensione di "Matrix") (+10 punti)

Nota: per poter fare questo esercizio, è obbligatorio consegnare anche una versione di Matrix che compila e funziona come richiesto.

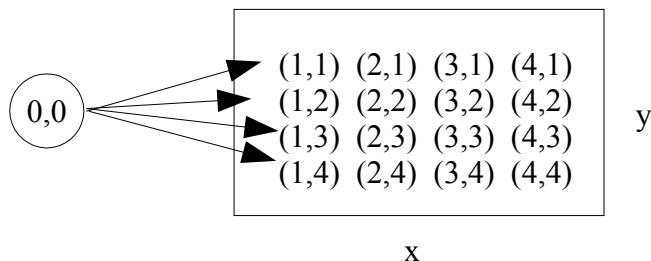
Estendere Matrix in modo tale che i processi possano comunicare lungo le righe, facendo uso di **pipe**. Ogni processo genera un numero casuale. I numeri vengono sommati per righe: il processo in posizione (N,y) comunica il proprio valore al processo (N-1,y), il quale lo somma al proprio numero e lo comunica al proprio genitore, e così via fino a quando il processo (1,y) riceve la somma parziale dal processo (2,y), e spedisce il risultato al genitore, il quale stampa il valore dopo avere ricevuto un valore da tutti i processi della prima colonna.

Esercizio 3 ("Pulizia doppioni") (4 punti)

Scrivere uno script che prende in input da linea di comando il nome di due directory ed elimina (da entrambe le directory) tutti i file che hanno lo stesso nome e lo stesso contenuto.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochiocciola.cs.unibo.it o renzochiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-4**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).



Esame di Laboratorio di Sistemi Operativi – 24/09/2004

Esercizio 0 ("Se copiate, vi caccio")

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 ("Matrix") (18 punti)

Scrivere un programma per creare una griglia di NxN processi. Ogni processo deve essere identificato con una coppia di valori (x,y), dove $1 \leq x \leq N$ e $1 \leq y \leq N$.

La creazione della griglia deve avvenire in questo modo:

- il processo genitore crea la riga "1" dei processi identificati da (x, 1)
- ogni processo (x,y) crea il processo (x, y+1), con $1 \leq y \leq N-1$
- ogni processo stampa "(x,y)" generato da (w,z) al momento della propria creazione, dove (x,y) è l'identificatore del processo appena creato, (w,z) è l'identificatore del padre (per distinguerlo, il processo iniziale può assumere i valori (0,0))

Ovviamente queste regole influiscono sull'ordine in cui vengono stampati gli identificatori, per cui la correttezza del programma verrà valutata proprio su questo.

Esercizio 2 ("Quadrato magico", estensione di "Matrix") (+10 punti)

Nota: per poter fare questo esercizio, è obbligatorio consegnare anche una versione di Matrix che compila e funziona come richiesto.

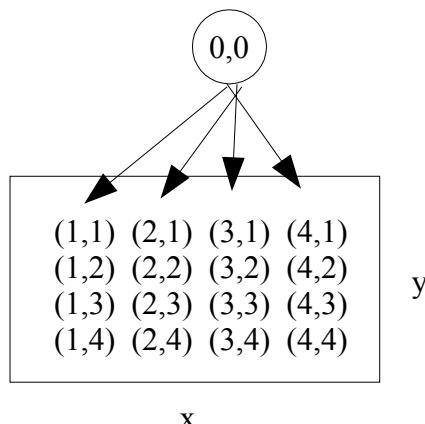
Estendere Matrix in modo tale che i processi possano comunicare lungo le colonne, facendo uso di **named pipe**. Ogni processo genera un numero casuale. I numeri vengono sommati per colonna: il processo in posizione (x,N) comunica il proprio valore al processo (x, N-1), il quale lo somma al proprio numero e lo comunica al proprio genitore, e così via fino a quando il processo (x,1) riceve la somma parziale dal processo (x,2), e spedisce il risultato al genitore, il quale stampa il valore dopo avere ricevuto un valore da tutti i processi della prima colonna.

Esercizio 3 ("Pulizia copie distinte") (4 punti)

Scrivere uno script che prende in input da linea di comando il nome di due directory ed elimina i file che hanno lo stesso nome, ma non lo stesso contenuto.

Esercizio 4 ("Consegnate! E' ora!":

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochiocciola.cs.unibo.it o renzochiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-4**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).



Esame di Laboratorio di Sistemi Operativi – 20/1/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Miss Italia”) (30 punti)

Scrivete un programma che simuli una serata del concorso di Miss Italia. Il processo genitore (che chiameremo “*mirigliani*”) crea un numero N=24 di processi “*concorrente*” e un numero M=6 di processi “*giuria*”. La serata si svolge in turni; ad ogni turno, vengono eliminate 4 concorrenti. Quando restano solo 4 concorrenti, viene proclamata la vincitrice. Ad ogni turno, i processi giuria comunicano al processo *mirigliani* un array di voti compresi fra 1 e 10, uno per ogni concorrente rimasta in gara. Quando ha ricevuto i voti da tutti i giurati, li somma e determina le concorrenti da eliminare (quelle con voto più basso). A questo punto, *mirigliani* “termina” le concorrenti eliminate e informa tutti i giurati di quali ragazze sono state eliminate. All’ultimo turno, *mirigliani* “incorona” la vincitrice e termina tutti gli altri processi.

Alcune note:

- Potete utilizzare il meccanismo di sincronizzazione/comunicazione che preferite
- Quando un processo termina, deve stampare un messaggio di saluto
- Più in generale, farcite il vostro programma di messaggi di debug.

Esercizio 2 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochiocciola_cs.unibo.it o renzochiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-3**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete sottoporvi alla discussione, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per discutere il vostro elaborato.

Esame di Laboratorio di Sistemi Operativi – 20/1/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Censore”) (22 punti)

Scrivere un programma “censore” in linguaggio C che funzioni come descritto in seguito.
con il comando:

censore parola comando

viene eseguito il comando. L'output viene filtrato e tutte le volte che si incontra la parola indicata questa viene cancellata.

ad esempio:

censore renzo cat myfile

pone in output il contenuto del file “myfile” cancellando tutti le parole “renzo”

Esercizio 2 (“miopid”) (10 punti)

Scrivere un programma “miopid” in linguaggio C che esegua un processo con il pid desiderato.

Il programma deve contenere una istruzione

printf(“PID=%d\n”,getpid());

Il programma

miopid 3333

tramite l'istruzione precedente deve stampare proprio

PID=3333

(hint: per tentativi, attenzione a non creare loop di fork con processi che non terminano).

Esercizio 3 (maxdir) (8 punti)

Scrivere uno script

maxdir size directory

e.g.

maxdir 100 .

che controlla l'ampiezza in blocchi di una directory. Se l'ampiezza indicata (100 blocchi nell'esempio) viene superata devono venir cancellati i file piu' vecchi fino a che l'ampiezza non sia minore o uguale dell'ampiezza desiderata.
Non si richiede la scansione ricorsiva.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a:
montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-3**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete sottoporvi alla discussione, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio.
Verrete richiamati uno alla volta per discutere il vostro elaborato.

Esame di Laboratorio di Sistemi Operativi – 10/2/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – Token-ring (22 punti)

In una rete token-ring, i nodi sono organizzati ad anello e si scambiano un “token” che passa da un nodo al successivo. Quando un processo riceve il token:

- stampa “*pid*: ho ricevuto il token”
- attende 1 secondo
- stampa “*pid*: spedisco il token al processo *x*”
- spedisce il token al processo *x*

dove *pid* è l'identificatore del processo che stampa, e *x* è l'identificatore del processo successivo.

Scrivere un programma che genera N processi che comunicano tramite "token-ring". Viene lasciata allo studente la scelta del meccanismo di sincronizzazione/comunicazione (ma vedi punto 2).

Esercizio 2 – Avanzato (8 punti)

Scrivere un programma in grado di "inserirsi" nella catena di processi vista in precedenza. Per farlo, è necessario specificare il pid di uno dei processi già presente nell'anello, e inserirsi in modo opportuno.

Si suggerisce di utilizzare named pipe; in ogni caso la scelta del meccanismo di sincronizzazione è libera.

Esercizio 3 - Etc log (4 punti)

Si crei un file di log degli accessi ai file presenti in /etc avvenuti nelle ultime 24 ore. Le informazioni devono comprendere: il nome del file, il nome del proprietario, l'ora di accesso. Si dovrà anche contrassegnare quel/quei file che hanno subito delle modifiche.

Esercizio 4 - Per chi ha occhio... (2 punti)

Scrive il più breve script auto-stampante (cioè che produce come output una copia di se stesso).

Esercizio 5 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Esame di Laboratorio di Sistemi Operativi – 27/6/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – “acceleratore lineare” (18 punti)

Si consideri una catena lineare di $N > 5$ processi, dove il processo originale 0 crea il processo 1, il processo 1 crea il processo 2, fino al processo $N-2$ che crea il processo $N-1$. I processi sono collegati fra di loro tramite pipe.

Esistono tre ruoli:

- “emitter”: il processo 0 emette periodicamente (1 volta al secondo) “particelle” che vengono mandate al primo processo “transport”
- “transport”: tutti i processi compresi fra 1 e $N-2$ (estremi inclusi) ricevono una particella dal processo precedente, e hanno il 50% di possibilità di trasmetterla al processo successivo. Altrimenti, la particella viene persa.
- “receiver”: il processo $N-1$ riceve le particelle, conta quante particelle sono state ricevute e riporta la media di particelle ricevute per secondo.

Esercizio 2 – Qualche dettaglio in più (14 punti)

Modificare l'esercizio precedente in modo tale che ogni processo deve monitorare l'esistenza in vita del processo padre e del processo figlio. Se scopre che il proprio processo padre è “morto”, diventa un emitter. Se scopre che il proprio processo figlio è morto, diventa un “receiver”. Se entrambi sono morti, termina.

Alcuni suggerimenti (poi sta a voi): guardate i lucidi relativi alle PIPE per scoprire come (i) rilevare la morte dell'estremo di scrittura e (ii) rilevare la morte dell'estremo di lettura.

Esercizio 3 - File duplicati (6 punti)

Scrivere uno script che prenda in input da riga di comando un insieme di directory e trovi tutti i file nelle directory specificate (in modo ricorsivo) che hanno lo stesso contenuto (ma non necessariamente lo stesso filename). Suggerimento: utilizzare i comandi find, md5sum, sort, uniq. L'output deve contenere solo i file duplicati, con i pathname di tutte le repliche.

Esempio di output:

```
aa45341430391cbc7969935926b9257 ./include/linux.h
aa45341430391cbc7969935926b9257 ./modulo1/include/linux.h
d2547d041aaafbf00b3d0a9c5536506 ./include/stats.h
d2547d041aaafbf00b3d0a9c5536506 ./modulo2/codice.h
```

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Esame di Laboratorio di Sistemi Operativi – 25/7/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – (24 punti)

Si consideri un processo padre che crea N processi figli.

I processi figli agiscono nel modo seguente:

- se ricevono un segnale dal padre, mandano al padre un segnale di risposta
- periodicamente (una volta al secondo), decidono (in modo casuale, con probabilità $0 < p \leq 1$) se continuare ad eseguire. In caso contrario, si suicidano chiamando exit()

Il processo padre agisce nel modo seguente:

- periodicamente (una volta al secondo), spedisce un segnale a tutti i figli e aspetta una risposta (suggerimento: contattate i figli uno alla volta)
- quando un processo figlio termina, stampa un messaggio di “addio”
- quando tutti i processi figli sono terminati, termina anch'esso.

Nota: “farcite” il vostro programma di stampe di controllo

Esercizio 2 – Statistica (6 punti)

Si consideri un file testuale dato da un insieme di righe suddivise in campi numerici separati da spazi. Scrivere uno script che prende in input dalla linea di comando il nome di tale file e un insieme di “indici” di campo. Per ogni “colonna” associata agli indici di campo, calcola e stampa il valore minimo, medio, e massimo per ognuna delle colonne.

Esempio:

```
5 7 2 3
4 4 2 1
3 2 1 4
4 4 4 4
parser file.txt 1 3
1: min:3 max: 5 avg:4
3: min:1 max: 4 avg: 2.25
```

Esercizio 3 - File duplicati (6 punti)

Scrivere uno script che prenda in input da riga di comando un insieme di directory e trovi tutti i file nelle directory specificate (in modo ricorsivo) che hanno lo stesso contenuto (ma non necessariamente lo stesso filename). Suggerimento: utilizzare i comandi find, md5sum, sort, uniq. L'output deve contenere solo i file duplicati, con i pathname di tutte le repliche.

Esempio di output:

```
aa45341430391cbc7969935926b9257 ./include/linux.h
aa45341430391cbc7969935926b9257 ./modulo1/include/linux.h
d2547d041aaafbfb00b3d0a9c5536506 ./include/stats.h
d2547d041aaafbfb00b3d0a9c5536506 ./modulo2/codice.h
```

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment devono contenere **il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi – 20/9/2005

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – (18 punti)

Scrivere un programma che faccia il ripetitore di segnali. Richiamato con il numero di un processo come argomento deve rispedire tutti i segnali che riceve al processo indicato come parametro (quelli “gestibili”, non ad esempio il segnale 9).

Es:

```
$ ripeti 2525  
rimandera' tutti i segnali ricevuti al processo 2525
```

Esercizio 2 – (10 punti)

Scrivere un programma che funzioni da “terminale” in grado di comunicare tramite due named pipe (fifo) con un programma gemello.

Es. dopo aver creato due fifo di nome f1 e f2 in una finestra si scrive:

```
$ fifoterm f1 f2  
in un'altra finestra  
$ fifoterm f2 f1
```

tutto quello che si digita in una finestra deve comparire nella seconda e viceversa.

Ogni carattere letto in input deve essere posto su uno dei due fifo (diciamo quello indicata come secondo parametro) e cio' che viene letto dall'altro fifo viene posto in output

Suggerimenti (aprire le fifo con parametri O_RDWR | O_NONBLOCK, leggere il manuale di poll o di select).

Esercizio 3 – (10 punti)

Scrivere uno script che elenchi tutti i file contenuti nel sottoalbero determinato da una directory, ordinandoli per lunghezza del pathname: prima quelli che hanno pathname relativo piu' corto poi via via quelli che hanno path piu' lungo.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochiocciola.cs.unibo.it o renzochiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi – 24/1/2006

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – (15 punti)

Scrivere un programma C che rispedisca al mittente ogni segnale ricevuto. (occorrono opportuni programmi campione di esempio per provarne il funzionamento).

Esercizio 2 – (15 punti)

Scrivere un programma C che costruisca la pipe circolare fra processi.

cpipe prog1 par11 par12 } prog2 par21 par22 } prog3 par31 par32 par33

esegua i programmi prog1 prog2 prog3 in modo che l'output del primo sia input per il secondo, l'output del secondo sia input per il terzo e così via. L'output dell'ultimo deve essere input del primo.

Esercizio 3 – (15 punti)

Scrivere uno script che faccia il merge di due alberi del file system copiandoli in un terzo.

La gerarchia risultante dovrebbe contenere tutti i file e le directory presenti nel primo o nel secondo albero.

Se due file hanno lo stesso percorso e nomi uguali nei due alberi di partenza i contenuti devono essere concatenati nel file risultante.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi – 15/02/2006

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 – (15 punti)

Scrivere un programma C che prese in input da riga comando il pathname di molteplici FIFO (almeno 1, numero imprecisato) metta in output tutto cio' che arriva da qualsiasi di esse.

Il programma **non** deve usare fork ma una chiamata a scelta fra select o poll.

Esercizio 2 – (10 punti)

Scrivere un programma in linguaggio C che attivi un programma e stia in attesa, se per qualche motivo il programma controllato termina lo deve lanciare nuovamente.

e.g.

```
respawn mydeamond -a -b -c
```

attiva mydaemond con i parametri indicati, se mydaemond termina viene rilanciato nello stesso modo.
(Hint usare wait o meglio waitpid).

Esercizio 2b - (+5 punti)

Mostrare il motivo della terminazione del processo.

Esercizio 3 – (15 punti)

Svolgere lo stesso compito dell'esercizio 2 con uno script BASH. Si chiede che lo script controlli ogni 10 secondi la presenza del processo da ps, se non esiste piu' lo deve riattivare.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment devono contenere il **vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Prova Pratica di Laboratorio di Sistemi Operativi

15 gennaio 2009

Esercizio 0 (“Se copiate, vi cacciamo”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio1 (obbligatorio): (15 punti) Scrivere un programma in linguaggio C "respawn" che provveda a riattivare un programma quando questo termina (Naturalmente o erroneamente). Es:

respawn test a b c

lancia il programma "test a b c". Se e quando questo dovesse terminare ne viene lanciato un altro uguale.

Esercizio 2: completamento dell'esercizio 1. (5 punti)

Modificare il programma dell'esercizio 1 per riattivare il programma solo se termina in modo anormale (per un segnale).

Esercizio 3 Script: (10 punti)

scrivere uno script bash *flatlink* con due parametri:

flatlink a b

dove a e' una directory e b e' una directory vuota.

Al termine dell'esecuzione la directory b deve contenere un link ad ogni file (non alle directory) contenuti in tutto il sottoalbero con radice in a.

b e' flat: non contiene sottodirectory, al contrario a puo' contenere sottodirectory.

Es:

se a contiene a1 a2 a3 a4 a5, a1 a2 sono dir e gli altri file

a1 contiene a11 a12 a13, dove solo a11 e' una dir

a11 contiene a111

a2 contiene a21 e a22

b al termine deve contenere a3 a4 a5 a12 a13 a111 a21 a22 link ai file omonimi in a.

Esercizio 4 Script: (10 punti)

scrivere uno script minimake.sh da invocare come

minimake.sh makefile

dove makefile deve contenere una sola linea secondo il seguente formato

fileout: filein1 filein2 ... fileinn | cmd arg1 ... argn

Se almeno uno dei file filein1 ... fileinn e' stato modificato l'ultima

volta dopo il file fileout, allora deve essere eseguito il comando

cmd con argomenti arg1 ... argn

Suggerimento: man test e guardare l'opzione -nt

Esercizio 5 (“Consegnate! E’ ora!”):

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: renzo.chioccia.cs.unibo.it. Il subject del mail deve essere uguale a **PROVAPRATICA**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

INOLTRE:

Se volete che il vostro lavoro venga giudicato, lasciate aperta la vostra sessione (incluso il vostro editor) e lasciate il laboratorio. Verrete richiamati uno alla volta per una breve discussione sul vostro elaborato.

Esame di Laboratorio di Sistemi Operativi - 23/7/2002 - 1

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Albero di sommatorie”) (30 punti)

Scrivere un programma C così organizzato. Si generi innanzitutto un albero binario di processi di altezza ALTEZZA. Se ALTEZZA=1, l’albero è costituito dal solo processo principale. Se ALTEZZA > 1, il processo padre genera due processi, che corrispondono ai figli destro e sinistro. Ognuno dei figli genera due alberi binari di processi di altezza ALTEZZA-1. I processi foglia (quelli dell’ultimo livello) generano un numero casuale, utilizzando le funzioni `rand()`, e lo comunicano al processo genitore. Il processo genitore riceve i valori dai due processi figli, li somma e comunica il risultato al proprio genitore. Il procedimento termina quando il processo radice riceve i valori dai propri figli e li somma. I processi foglia devono stampare l’identificatore del processo e il valore generato. Tutti gli altri processi devono stampare il proprio identificatore, i valori ricevuti dai figli e la somma così ottenuta. Risolvere il problema utilizzando **pipe**.

PS Per le prove, utilizzare valori bassi di altezza (3-4), per evitare di incorrere in limitazioni sul numero di processi.

Esercizio 2 (“Voto medio”) (valore 12 punti)

Si assuma di avere un file contenente linee con il seguente formato:

VOTO;<Cognome>;<Nome>;<Sesso>;<Voto>

Esempio:

VOTO;Montresor;Alberto;M;30

VOTO;Turrini;Elisa;F;30

VOTO;Davoli;Renzo;M;30

Scrivere uno o più script (di qualunque tipo, shell, awk, etc.) che stampi un elenco di tutte le persone di sesso maschile terminato dal corrispondente voto medio, seguito dall’elenco di tutte le persone di sesso femminile terminato dal corrispondente voto medio.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-5**, il nome dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi - 9/9/2002

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Morra cinese”) (22 punti)

Scrivere un programma C così organizzato. Un processo padre genera una coppia di processi figli. I tre processi giocano a morra cinese, con due processi giocatori e un processo arbitro. I processi si alternano (in modo round-robin) nel ruolo dell’arbitro. Ad ogni turno di gioco, ognuno dei processi giocatori (i) sceglie tra “carta, forbice e sasso” (ii) stampa la propria scelta (iii) comunica la propria scelta all’arbitro. L’arbitro (i) riceve la scelta da entrambi i giocatori (ii) decide chi è il vincitore (iii) stampa un messaggio con il risultato e comunica il risultato ai due giocatori. Il processo vincitore stampa un messaggio di esultanza, mentre il processo perdente stampa un messaggio di sconforto. Dopo di che, si passa al turno di gioco successivo. Il gioco va avanti per un certo numero (NTURNI) di gioco, dopo di che tutti i processi terminano. Per la soluzione del problema, utilizzate **segnali**.

PS Stabilite un’ordinamento tra i processi, noto a tutti i processi, per scegliere ad ogni turno chi fa l’arbitro.

Esercizio 2 (“Mettiamo ordine”) (valore 14 punti)

Si assuma di avere un albero di directory contenente sottodirectory e file con estensione .jpg, di varie dimensioni. Scrivere uno script copypics <sourcedir> <destdir> che prenda in input il path iniziale dell’albero di directory <sourcedir>, una directory destinazione <destdir>, e copi tutti i file jpg di dimensione superiore a 10240 byte nella directory destinazione, numerandoli progressivamente. Se nella directory di destinazione sono già presenti n file, numerare i file a partire dal valore n+1.

Esempio (per semplicità grafica, tutti i file hanno dimensione superiore a 10240 byte):

Directory sorgente:	Directory destinazione (prima):	Directory destinazione dopo il comando copypics:
/download/ img01.jpg img02.jpg img03.jpg backgrounds/ sanfrancisco.jpg newyork.jpg cartoons/ ippo.jpg paperino.jpg	/pics 1.jpg 2.jpg 3.jpg	/pics 1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg 9.jpg 10.jpg

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-6**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi -27 /9/2002

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Telefono senza fili al contrario”) (24 punti)

Scrivere un programma C così organizzato. Innanzitutto, si generano N processi nel modo seguente. Il primo processo (P_1) genera un processo figlio P_2 . P_2 genera un processo figlio P_3 . Questa costruzione va avanti fino a quando sono stati generati N processi. Quando un processo viene creato, stampa il proprio process id e l'id del processo padre. Il processo P_N genera casualmente una valore intero, e lo comunica a P_{N-1} . Un processo P_i (con $1 < i < N-1$) riceve il valore da P_{i+1} , gli somma +1 o -1 (casualmente) e comunica il nuovo valore a P_{i-1} . Il processo P_1 riceve un valore da P_2 , gli somma +1 o -1 (casualmente) e termina. Stampare un messaggio opportuno tutte le volte che si genera, si spedisce o si riceve un valore. Per la sincronizzazione, utilizzare named pipe.

Esercizio 2 (“Convertire figure da un formato all’altro”) (12 punti)

Nelle normali distribuzioni Linux, esistono numerosi tool per convertire figure da un formato all’altro. Ad esempio,

- **pdftoeps -eps <file>.pdf** converte una figura in formato PDF (Portable Data Format) in una figura EPS (Encapsulated PostScript), creando il file **<file>.eps** a partire dal file **<file>.pdf**
- **fig2dev -L eps <file>.fig <file>.eps** converte una figura in formato FIG in una figura in formato EPS, creando il file **<file>.eps** a partire dal file **<file>.fig**
- **fig2dev -L pdf <file>.fig <file>.pdf** converte una figura in formato FIG in una figura in formato PDF, creando il file **<file>.pdf** a partire dal file **<file>.fig**

Scrivere uno script **convertfig <dir>** che effettui le seguenti operazioni sulla directory **<dir>**:

- per ogni file con estensione **.fig** contenuto in **<dir>**, convertire tale file in due file con estensione **.eps** e **.pdf**, solo se tali file non esistono oppure sono meno recenti del file **<file>.fig**
- per ogni file con estensione **.pdf** contenuto in **<dir>**, convertire tale file in un file con estensione **.eps**, solo se non esiste un file con lo stesso nome ed estensione **.fig**, e solo se il file **.pdf** è più recente del file con estensione **.eps**.

Per effettuare le vostre prove, potete utilizzare la directory di esempio /public/provapratica/sourcedir che contiene i file da trasformare, e la directory /public/provapratica/sampledir che contiene un esempio del risultato.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-7**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 31/01/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Minmax bifronte”) (30 punti)

(Parte 1: creazione dei processi e creazione degli opportuni PIPE, 18 punti)

Scrivere un programma C chiamato così organizzato. Innanzitutto, vengono generati N processi nel modo seguente. Il processo padre P_1 genera un processo figlio P_2 , il quale genera un processo figlio P_3 ; il procedimento va avanti fino a quando sono stati generati N processi (incluso il padre). Utilizzando il meccanismo di comunicazione pipe, ogni processo P_i comunica con il proprio vicino di sinistra P_{i-1} (il genitore) e il proprio vicino di destra P_{i+1} (il figlio), tranne P_1 (che non ha genitore) e P_N (che non ha figli).

(Parte 2: comunicazione in un senso, 6 punti)

Inizialmente, ogni processo genera un numero casuale compreso tra 0 e 100 e lo mette in una variabile *valore*. Il processo P_N spedisce il valore generato al processo P_{N-1} . Ogni processo P_i (con $1 < i < N$) si mette in attesa di ricevere un valore h da P_{i+1} ; confronta il valore ricevuto h con quello contenuto in *valore* e spedisce al processo P_{i-1} il valore minore tra i due. Il processo P_1 attende di ricevere un valore da P_2 , lo confronta con il proprio, e stampa il valore minimo tra i due (corrispondente al minimo assoluto).

(Parte 3: comunicazione nel senso opposto, 6 punti)

Dopo di che, il processo P_1 spedisce il proprio valore al processo P_2 . Ogni processo P_i (con $1 < i < N$) si mette in attesa di ricevere un valore h da P_{i-1} ; confronta il valore ricevuto h con quello contenuto in *valore* e spedisce al processo P_{i+1} il valore massimo tra i due. Il processo P_N attende di ricevere un valore da P_{N-1} , lo confronta con il proprio, e stampa il valore massimo tra i due (corrispondente al massimo assoluto)

Nota: al fine di semplificare la correzione, stampate informazioni tipo “il processo *pid* ha spedito *h*” / “il processo *pid* ha ricevuto *h*”, “il processo *pid* ha creato il processo *pidfiglio*”

Esercizio 2 (“Aggiungere un commento iniziale”) (6 punti)

Scrivere uno script che prenda in input da linea di comando il pathname di una directory e il pathname di un file contenente testo. Lo script deve individuare tutti i file contenuti nella directory e nelle sue sottodirectory, che abbiano estensione .java, e che non contengano la parola “copyright” (case-insensitive). Lo script deve aggiungere (all’inizio, prima del testo esistente) ad ognuno di questi file il testo contenuto nel file specificato.

Questo script può essere utile per aggiungere un commento iniziale di copyright ad un insieme di sorgenti.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-1**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 18/02/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Ping pong”) (versione base, 24 punti)

Scrivere un programma C formato da due processi. Il processo padre crea un processo figlio. I due processi giocano una partita di ping pong. Una partita è formata da un certo numero di turni (1 nella versione base). Ogni partita viene iniziata dal padre. Il padre inizia la partita "battendo": : seleziona a caso un segnale fra SIGUSR1 e SIGUSR2, e lo invia al processo figlio. Il processo figlio si "difende", scegliendo a caso fra SIGUSR1 o SIGUSR2. Se seleziona il segnale sbagliato, stampa "punto al processo <pid>"; se seleziona il segnale giusto, "risponde" selezionando a caso un segnale fra SIGUSR1 e SIGUSR2 e lo invia al padre. Nel caso di segnale giusto, il padre continua a giocare rispondendo al figlio, e così via fino a quando qualcuno non seleziona il segnale sbagliato. (Nota: aggiungete stampe di debug ad ogni azione dei processi)

Esercizio 2 (“Ping pong”) (versione avanzata, +6 punti)

Completare l'esercizio precedente, facendo in modo che la partita termini non appena un processo raggiunge i 15 punti (un punto per ogni volta che l'altro processo difende il segnale sbagliato). Quando un processo fa "cadere la pallina", stampa "punto al processo <pid>" e poi inizia un nuovo turno.

Esercizio 3 (“Voto medio”) (valore 8 punti)

Si assuma di avere un file contenente linee con il seguente formato:

VOTO;<Cognome>;<Nome>;<Sesso>;<Voto>

Esempio:

VOTO;Montresor;Alberto;M;30

VOTO;Turrini;Elisa;F;30

VOTO;Davoli;Renzo;M;30

Scrivere uno o più script (di qualunque tipo, shell, awk, etc.) che stampi un elenco di tutte le persone di sesso maschile terminato dal corrispondente voto medio, seguito dall'elenco di tutte le persone di sesso femminile terminato dal corrispondente voto medio.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso.chiocciola.cs.unibo.it o renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-2**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

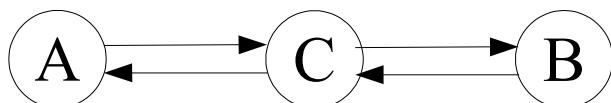
Esame di Laboratorio di Sistemi Operativi – 11/06/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Binario unico alternato”) (versione base, 24 punti)

Scrivere un programma C così formato. Un processo base (che poi non verrà utilizzato nel resto dell'esercizio) crea tre processi figli, connessi tra loro tramite named pipe nel modo seguente:



I processi A e B rappresentano due città, mentre il processo C rappresenta un binario unico alternato che collega le due città. Le frecce in direzione opposta significano che A può comunicare con C e viceversa, e che B può comunicare con C e viceversa.

La vita dei processi città A e B è la seguente:

1. il processo “dorme” per un periodo di tempo casuale (compreso tra 1 e 5 secondi)
2. genera un “treno” che deve andare verso la città opposta; per fare passare il treno
 1. il processo che genera il treno spedisce un messaggio al processo C
 2. attende un messaggio di via libera dal processo C
3. torna al punto 1

Come è possibile osservare, un singolo processo città non genera mai più di un treno alla volta, quindi in un singolo processo al più un treno può essere fermo in attesa che si liberi il binario.

La vita del processo C è la seguente:

1. il processo attende richieste di passaggio treno da una città
2. se nessun treno è sul binario, risponde con un messaggio di via libera al processo che ha fatto richiesta
3. quando un treno arriva sul binario, il processo C si sospende per 2 secondi (il tempo necessario per far passare il treno)
4. quando il tempo è scaduto, il processo libera il binario e si mette in attesa di nuove richieste dall'altra città.

Scrivere il programma qui descritto, facendo attenzione a evitare problemi di starvation. Aggiungere stampe di debug in modo opportuno.

Esercizio 2 (“Binario unico”) (versione estesa, 8 punti)

Migliorare il programma evitando un'alternanza stretta (per cui se per un periodo di tempo molto lungo non ci sono treni da una città, il binario lascia passare più treni dalla stessa).

Esercizio 3 (“Replace multi-file”) (10 punti)

Scrivere uno script che prende in input due stringhe e una lista di file; lo script deve scandire il contenuto dei file, e sostituire tutte le occorrenze della prima stringa con la seconda nei file stessi.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montres0.chiocciola.cs.unibo.it o renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-3**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

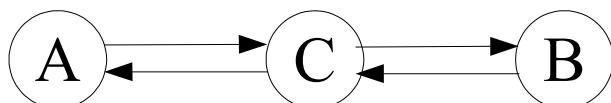
Esame di Laboratorio di Sistemi Operativi – 02/07/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Binario unico alternato con segnali”) (versione base, 18 punti)

Scrivere un programma C così formato. Un processo base (che poi non verrà utilizzato nel resto dell'esercizio) crea tre processi figli:



I processi A e B rappresentano due città, mentre il processo C rappresenta un binario unico alternato che collega le due città. Le frecce in direzione opposta significano che A può mandare segnali a C e viceversa, e B può mandare segnali a C e viceversa.

La vita dei processi città A e B è la seguente:

1. il processo “dorme” per un periodo di tempo casuale (compreso tra 1 e 5 secondi)
2. genera un “treno” che deve andare verso la città opposta; per fare passare il treno
 1. il processo che genera il treno spedisce un segnale al processo C
 2. attende un segnale di via libera dal processo C
3. torna al punto 1

Come è possibile osservare, un singolo processo città non genera mai più di un treno alla volta.

La vita del processo C è la seguente:

il processo organizza un meccanismo di turni, in base al quale prima soddisfa i processi della città A e poi i processi della città B, e così via

1. attende un segnale di richiesta passaggio dalla città di turno;
2. si sospende per due secondi (per simulare il tempo di passaggio di un treno)
3. risponde con un segnale di via libera alla città di turno
4. cambia turno

Scrivere il programma qui descritto, facendo attenzione a evitare problemi di starvation. Aggiungere stampe di debug in modo opportuno.

Esercizio 2 (“Binario unico con segnali”) (versione estesa, 12 punti)

Migliorare il programma evitando il meccanismo dei turni. In altre parole, se in un intervallo di tempo arrivano più treni da una città e nessun treno dall'altra, si lasciano passare più treni dalla stessa città.

Esercizio 3 (“Grep pirramidale (?)”) (8 punti)

Scrivere uno script che prende in input (da linea di comando) una stringa e una lista di file; lo script deve riportare il numero di occorrenze della stringa e di tutti i suoi prefissi nei file specificati. Ad esempio, se la stringa cercata è linux, lo script deve riportare un output del tipo:

```
1 512  
li 324  
lin 300  
linu 30  
linux 30
```

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montres0.chiocciola.cs.unibo.it o renzo.chiocciola.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-4**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 10/09/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Roulette russa”) (versione base, 18 punti)

Un processo "suicida" crea un processo figlio "pistola". La pistola è a sei colpi, ma uno solo è in canna.

Il processo suicida "gioca" per NTURNI turni, in ognuno dei quali: (i) stampa "Addio!" (ii) invia un segnale di "pressione grilletto" al processo pistola (iii) resta in attesa di un segnale di "cilecca" dalla pistola (iv) stampa "cilecca". Al termine dei turni previsti, il suicida invia un segnale "metti sicura" diverso da "pressione grilletto" chiedendo alla pistola di terminare.

La pistola agisce nel modo seguente: (i) attende un segnale "pressione grilletto" o "metti sicura"; (ii) nel primo caso lancia un numero casuale da 1 a 6; se il risultato è 1, manda un segnale SIGKILL al processo suicida ed esce; altrimenti, stampa "Click!" e manda un segnale "cilecca" al suicida. (iii) nel caso di "metti sicura", termina immediatamente.

Esercizio 2 ("Roulette russa") (versione estesa, 8 punti) (affinchè questo venga considerato, è obbligatorio consegnare l'esercizio 1 completo)

Scrivere una versione alternativa della roulette russa in cui il processo originario è la pistola. 2 figli vengono generati dalla pistola. La pistola viene utilizzata a turno dai figli, per NTURNI giri completi. Utilizzare un ulteriore scambio di segnali per rendere noto ad un processo che è il suo turno di utilizzare la pistola.

Esercizio 3 (“Come semplificare la vita del docente...”) (12 punti)

Nella directory /public/compiti trovate un insieme di file pdf che contengono compiti di S.O. I file hanno questa struttura: yyyy-mm-dd.zzz.pdf, dove yyyy è l'anno, mm è il mese, dd è il giorno, e zzz è uguale a "ris" (gestione risorse) oppure "con" (concorrenza). Scrivere uno script che stampi su stdout una serie di righe di tabella html, una per data, contenente una colonna per la data, una colonna per il compito di concorrenza e una colonna per il compito di gestione risorse, con l'aggiunta della dimensione del file. Ad esempio (singola riga):

```
<tr>
  <td>2003-09-08</td>
  <td><a href="compiti/2003-09-08.con.pdf">PDF 35 KB</a></td>
  <td><a href="compiti/2003-09-08.ris.pdf">PDF 27 KB</a></td>
</tr>
```

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochioccia.cs.unibo.it o renzochioccia.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-5**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

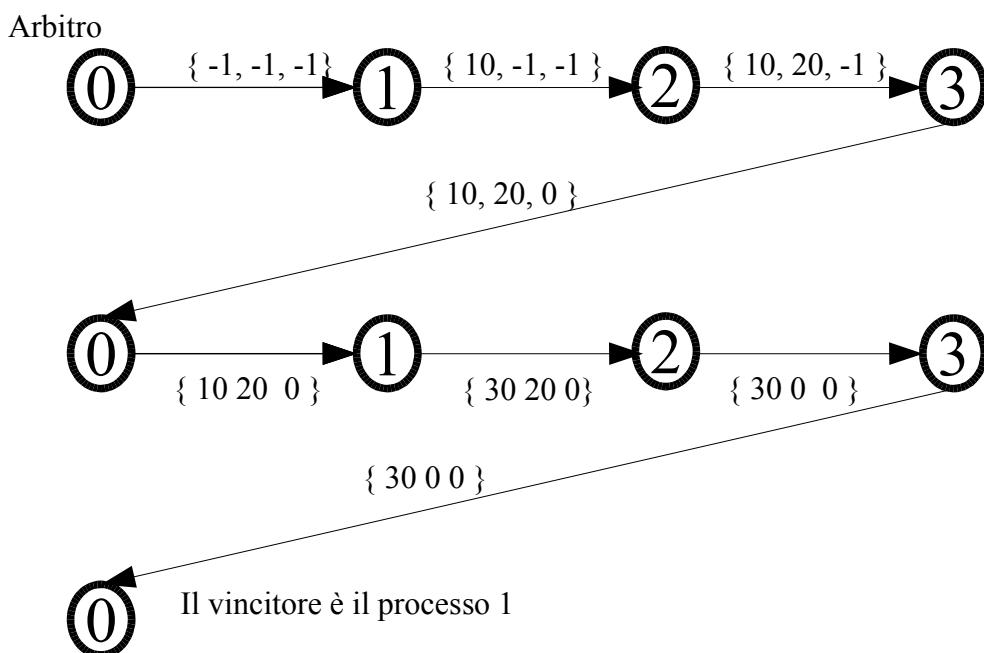
Esame di Laboratorio di Sistemi Operativi – 24/09/2003

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Asta on-line”) (24 punti)

Scrivere un programma che crea K processi che partecipano ad un'asta. I processi sono connessi tramite **pipe** per formare un anello: il processo padre (0) è connesso al suo unico processo figlio (1), il processo 1 è connesso al processo 2, etc, fino al processo K che è connesso al processo 0 (il padre originario). Il processo padre funge da arbitro, mentre gli altri processi partecipano come acquirenti. Il processo padre spedisce al primo figlio un array di K valori contenenti le offerte attuali (tutte -1, indicanti che nessuno ancora ha offerto). Quando un acquirente riceve una lista di offerte deve decidere se (a) fare una nuova offerta superiore all'attuale offerta massima o (b) lasciare l'asta. Se fa una nuova offerta, deve inserire il nuovo valore nell'array. Altrimenti, deve inserire il valore 0. Se mette il valore 0, non potrà più partecipare all'asta nei giri successivi. L'insieme delle offerte viene rispedito al processo successivo. L'asta può andare avanti per diversi giri di offerte effettuati da tutti i partecipanti. Tutte le volte che il processo arbitro riceve la sequenza di offerte, verifica: (a) se tutte le offerte sono 0, significa che l'asta è chiusa senza un acquirente per il prodotto (b) se rimane una sola offerta, il processo corrispondente viene dichiarato vincitore (c) altrimenti, rispedisce la sequenza così com'è per iniziare un nuovo giro.



Esercizio 3 (“Comprimi, ma solo se ne vale la pena”) (6 o 10 punti)

a) (6 punti) Scrivere uno script che prende in input da linea di comando il path di una directory, e per ogni file contenuto nella directory effettua la seguente trasformazione: se utilizzando un programma di compressione (a vostra scelta) sul file si ottiene un file compresso la cui dimensione è inferiore alla metà di quella del file originale, rimuove il file originale e lascia solo quello compresso. Altrimenti (dimensione superiore alla metà) lascia il file originale. Scrivere inoltre lo script che effettua l'operazione contraria, ovvero decomprime tutti i file.

b) (10 punti) Come sopra, ma ricorsivo.

Esercizio 4 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresochioccia.cs.unibo.it o renzochioccia.cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-6**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 27/01/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Di nuovo alberi!”)

a) 18 punti

Scrivete un programma così organizzato. Un processo progenitore genera un albero binario di processi (i.e., il progenitore ha due figli, ognuno di questi due figli ha due figli, e così via. La profondità dell'albero è N. Scegliete un N piccolo, tipo 3 o 4.

b) 8 punti

I processi foglia (quelli che non hanno figli) generano un numero casuale fra 1 e 10 e lo comunicano al loro processo padre. Il processi intermedi ricevono i dati dai loro processi figli, li sommano assieme e spediscono il risultato al loro processo padre. Il processo progenitore non fa nulla.

Tutti i processi devono stampare esaurientemente la loro attività: chi sono, la loro posizione nell'albero (tipo: liv. 2 – destra), cosa ricevono e cosa inviano.

c) 4 punti

Una volta ottenuto il risultato, il processo genitore lo comunica a tutti i processi sottostanti.

Risolvete l'esercizio tramite PIPE.

Esercizio 2 (“Finto tripwire”) (6 punti)

Scrivete uno script che prende in input da linea di comando il path di due directory, che si suppone abbiano lo stesso contenuto (suggerimento per fare test: copiate una directory in un'altra), e verifichi se sono stati modificati dei file (ad esempio, uno è più aggiornato dell'altro). Stampare le eventuali differenze.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-7**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 17/02/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“S.O.S”) (30 punti)

Due processi vogliono comunicare utilizzando solo segnali. Per farlo, si scambiano delle stringhe null-terminated (ovvero che terminano con il byte 0). Lo schema di funzionamento è il seguente: i caratteri che compongono una stringa vengono spediti uno alla volta dal processo mittente al processo destinazione, fino a quando il mittente spedisce il byte di terminazione 0. A quel punto, si passa alla stringa successiva. Tutti i byte vengono spediti bit per bit; bit 0 e bit 1 devono essere caratterizzati da segnali differenti.

a)

Scrivete una versione half-duplex di questo meccanismo: ovvero un solo processo spedisce, l’altro riceve. Spedite un certo numero di stringhe, prese in input dalla linea di comando. Create il processo destinatario.

b)

Scrivete una versione full-duplex dello stesso meccanismo (entrambi spediscono e ricevono). Attenzione ai problemi di sincronizzazione.

Esercizio 2 ("Trova duplicati")

Scrivete uno script chiamato dfind-<vostrcognome>, che prende in input da linea di comando una lista di directory e cerca se in una qualsiasi di queste directory esistono due file con pathname diverso che hanno lo stesso contenuto. Stampate in stdout la lista di questi file.

Requisiti:

- non utilizzate un approccio "quadratico": per ogni file, confronto il file con tutti gli altri. Una complessità quadratica è proibitiva e va evitata; è possibile risolvere il problema in tempo lineare sul numero di file.
- utilizzate invece md5sum (utilizzate man per dettagli), che calcola una somma crittografica del contenuto di un file. Se due file hanno lo stesso contenuto, l'output di md5sum è uguale per entrambi; se due file hanno contenuto diverso, l'output di md5sum è con altissima probabilità diverso (ma può anche essere uguale, quindi utilizzate cmp per verificare file che hanno la stessa somma crittografica)
- la ricerca deve essere ricorsiva, ovvero includere anche le sottodirectory.

Suggerimenti

- utilizzate comandi come find, sort, uniq, cmp
- risolvete prima il problema per una singola directory su linea di comando, poi per più directory

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-8**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi – 21/06/2004

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Visita albero ”) (24 punti)

Scrivere un programma C che crea un albero binario bilanciato di processi. L'altezza dell'albero è data da un parametro configurabile N; per evitare la creazione di un numero eccessivo di processi, utilizzate valori bassi (tipo 3 o 4).

Ogni processo non-foglia è collegato ai propri figli tramite *una coppia* di pipe in scrittura (senso di comunicazione: da padre a figli) e con *una* pipe in lettura (senso di comunicazione: da figli a padre).

Ogni processo p genera un numero casuale c_p . Questi numeri verranno sommati tramite visita "depth-first" dell'albero, nel modo seguente:

quando il processo p riceve una richiesta di "somma" dal proprio genitore

p manda una richiesta di "somma" al figlio sinistro

p riceve una risposta r_s dal figlio sinistro

p manda una richiesta di "somma" al figlio destro

p riceve una risposta r_d dal figlio destro

p spedisce la somma $r_s + r_d + c_p$ al proprio genitore

Il meccanismo viene fatto partire dalla radice, che inizia a spedire le prime richieste di somma.

Esercizio 2 ("Script, script delle mie brame, chi è il più potente del reame?") (9 punti)

Scrivere un meccanismo per ottenere la capacità di memoria e la frequenza di CPU di tutte le macchine di laboratorio che non siano attualmente "down" (guaste, scollegate, in update, etc.). Il formato deve avere la forma (sono ammesse leggere variazioni, se semplificano la realizzazione dello script):

remendado 1200 Mhz 257242 Kb

dotto 700 Mhz 514484 Kb

...

Per ottenere la lista delle macchine presenti, utilizzate ruptime. Per ottenere informazioni sulla memoria e sulla cpu, utilizzate /proc/meminfo e /proc/cpuinfo. Per eseguire comandi in remoto, utilizzate ssh -e. Non considerate eventuali problemi di richiesta di password; possono essere risolti con l'utilizzo di chiavi pubbliche/private.

Il risultato deve essere ordinato per frequenze decrescenti.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montreso_chiocciola_cs.unibo.it o renzo_chiocciola_cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-1**, i nomi dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).

Esame di Laboratorio di Sistemi Operativi

24/1/2002 - 1

Esercizio 1 (valore 10 punti)

Scrivere uno script che estragga messaggi con subject **SUBSCRIBE: <email address>** da un folder di posta elettronica e che inserisca **<email address>** in un file chiamato **mailing.list**, evitando di inserire duplicati. Esempio: da un file

```
From: montreso@cs.unibo.it
Subject: SUBSCRIBE: montreso@cs.unibo.it
Date: xx.xx.xx
From: montreso@cs.unibo.it
Subject: SUBSCRIBE: montreso@phd.cs.unibo.it
Date: xx.xx.xx
From: rossi@cs.unibo.it
Subject: SUBSCRIBE: rossi@cs.unibo.it
Date: xx.xx.xx
From: bononi@cs.unibo.it
Subject: SUBSCRIBE: rossi@cs.unibo.it
Date: xx.xx.xx
```

il file risultante deve contenere:

```
montreso@cs.unibo.it
montreso@phd.cs.unibo.it
rossi@cs.unibo.it
```

Esercizio 2 (valore 20 punti)

Scrivere un programma C così organizzato. Un processo padre genera una coppia di processi figli. I due processi figli giocano a testa o croce, mentre il programma padre fa da arbitro. Ad ogni turno di gioco, uno dei processi figli (a turno, alternandosi) (i) sceglie tra testa e croce (ii) stampa la propria scelta (iii) comunica la propria scelta al padre. Il padre (i) comunica la scelta all'altro figlio (ii) lancia una moneta (tramite funzione `srandom`) (iii) stampa un messaggio con il risultato e comunica il risultato dell'estrazione ad entrambi i processi figli. Il processo vincitore stampa un messaggio di esultanza, mentre il processo perdente stampa un messaggio di sconforto. Dopo di che, si passa al turno di gioco successivo. Il gioco va avanti per un certo numero (`NTURNI`) di giochi, dopo di che tutti i programmi escono.

Risolvere il problema utilizzando i segnali.

Consegnare il sorgente del programma C, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-1**.

Esame di Laboratorio di Sistemi Operativi

15/2/2002

Esercizio 1 (valore 18 punti)

Si assuma di avere un file contenente un insieme di login name, ad esempio:

montreso
rossi
sacerdot
renzo

Scrivere uno script che esegua periodicamente le seguenti operazioni (ad esempio, ogni 30 secondi):

1. Verificare che non sia possibile accedere in nessun modo alla home degli utenti contenuti nell'elenco
2. Verificare che nella directory public non vi siano file appartenenti agli utenti contenuti nell'elenco

Nel caso una delle due condizioni non si verifichi per un utente, stampare un messaggio di errore.

Esercizio 2 (valore 24 punti)

Scrivere un programma C così organizzato. Un processo padre genera una coppia di processi figli. I due processi figli giocano a testa o croce, mentre il programma padre fa da arbitro. Ad ogni turno di gioco, uno dei processi figli (a turno, alternandosi) (i) sceglie tra testa e croce (ii) stampa la propria scelta (iii) comunica la propria scelta al padre. Il padre (i) comunica la scelta all'altro figlio (ii) lancia una moneta (tramite funzione `srandom`) (iii) stampa un messaggio con il risultato e comunica il risultato dell'estrazione ad entrambi i processi figli. Il processo vincitore stampa un messaggio di esultanza, mentre il processo perdente stampa un messaggio di sconforto. Dopo di che, si passa al turno di gioco successivo. Il gioco va avanti per un certo numero (NTURNI) di giochi, dopo di che tutti i programmi escono.

Risolvere il problema utilizzando pipe.

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-2**

Esame di Laboratorio di Sistemi Operativi - 22/2/2002

Esercizio 1 (“Telefono senza fili”) (valore 24 punti)

Scrivere un programma C così organizzato. Innanzitutto, si generano NUMPROC processi nel modo seguente. Il primo processo (P_1) genera un processo figlio P_2 . P_2 genera un processo figlio P_3 . Questa costruzione va avanti fino a quando sono stati generati NUMPROC processi. Quando un processo viene creato, stampa il proprio process id e l'id del processo padre. Il processo P_1 genera casualmente un valore intero, e lo comunica a P_2 . Un processo P_i (con $1 < i < \text{NUMPROC}-1$) riceve il valore da P_{i-1} , gli somma +1 o -1 (casualmente) e comunica il nuovo valore a P_{i+1} . Il processo P_{NUMPROC} riceve un valore da $P_{\text{NUMPROC}-1}$, gli soma +1 o -1 (casualmente) e comunica il nuovo valore a P_1 . P_1 riceve questo valore e lo confronta con il valore iniziale. Stampare un messaggio opportuno tutte le volte che si genera, si spedisce o si riceve un valore.

Esercizio 2 (“Iscrizione appelli) (valore 18 punti completo, 14 punti senza la parte opzionale)

Si assuma di avere un file contenente linee con il seguente formato:

Subject: LSO-ESAME;<appello>;<matricola 6 cifre>;<indirizzo email>;<nome>;<cognome>

Esempio:

Subject: LSO-ESAME;1;118745;xx1@yyy.it;Gian_Piero;Favini
Subject: LSO-ESAME;2;119093;xx2@yyy.it;Matteo;Bagnasco
Subject: LSO-ESAME;1;119096;xx3@yyy.it;Nicola;Bagnasco
Subject: LSO-ESAME;2;119201;xx4@yyy.it;Davide;Ferrari
Subject: LSO-ESAME;1;119790;xx5@yyy.it;Marco;Rimondini
Subject: LSO-ESAME;1;119802;xx6@yyy.it;Simone;Frau
Subject: LSO-ESAME;1;120029;xx7@yyy.it;Andrea;Perdicchia
Subject: LSO-ESAME;2;120032;xx8@yyy.it;Paolo;Tomeo
Subject: LSO-ESAME;2;120157;xx9@yyy.it;Alfredo;Moretta
Subject: LSO-ESAME;2;120251;xxA@yyy.it;Alessandro;Martella

(durante la durata dell'esercizio, è possibile trovare una copia in /public/esercizio2)

Scrivere uno o più script (di qualunque tipo, shell, awk, etc.) che

1) verifichi la validità delle righe di iscrizione contenute nel file

- devono iniziare con Subject:
- proseguire con LSO-ESAME
- avere un numero di appello pari a 1 o 2
- avere sei cifre nel campo matricola
- avere un indirizzo email nel campo email (verificate che ci sia @)

2) spedisca un messaggio ad ogni persona confermando l'iscrizione all'appello desiderato, oppure un messaggio di avvertimento che l'iscrizione non è andata a buon fine (ovviamente tranne nel caso di indirizzo email errato).

3) stampi il numero di iscritti al 1° appello, il numero di iscritti al 2° appello, e il numero di messaggi errati

4) !opzionale!: elimini iscrizioni duplicate (stesso numero di matricola in due o più righe, assumiamo che non ci si possa iscrivere a due appelli)

Consegnare lo script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-3**, il nome dei file in attachment devono contenere il vostro cognome (per evitare confusioni dopo).

Esame di Laboratorio di Sistemi Operativi - 10/6/2002 - 2

Esercizio 0 (“Se copiate, vi caccio”)

Rendete la vostra directory home inaccessibile ad altri utenti (sia in lettura che in esecuzione). Rimuovete tutti i file che vi appartengono dalla directory /public.

Esercizio 1 (“Morra cinese”) (24 punti)

Scrivere un programma C così organizzato. Un processo padre genera una coppia di processi figli. I tre processi giocano a morra cinese, con due processi giocatori e un processo arbitro. I processi si alternano (in modo round-robin) nel ruolo dell’arbitro. Ad ogni turno di gioco, ognuno dei processi giocatori (i) sceglie tra “carta, forbice e sasso” (ii) stampa la propria scelta (iii) comunica la propria scelta all’arbitro. L’arbitro (i) riceve la scelta da entrambi i giocatori (ii) decide chi è il vincitore (iii) stampa un messaggio con il risultato e comunica il risultato ai due giocatori. Il processo vincitore stampa un messaggio di esultanza, mentre il processo perdente stampa un messaggio di sconforto. Dopo di che, si passa al turno di gioco successivo. Il gioco va avanti per un certo numero (NTURNI) di gioco, dopo di che tutti i processi terminano. Per la soluzione del problema, utilizzate **named pipe**.

PS Stabilite un’ordinamento tra i processi, noto a tutti i processi, per scegliere ad ogni turno chi fa l’arbitro.

Esercizio 2 (“Meglio A-L o M-Z?”) (valore 12 punti)

Si assuma di avere un file contenente linee con il seguente formato:

VOTO;<Cognome>;<Nome>;<Voto scritto>;<Voto finale>

Esempio:

VOTO;Montresor;Alberto;28;30

VOTO;Turrini;Elisa;29;30

VOTO;Davoli;Renzo;30;30

Scrivere uno o più script (di qualunque tipo, shell, awk, etc.) che stampi (i) il voto medio delle persone con il cognome che inizia con A-L, sia per lo scritto che per il voto finale (ii) il voto finale più alto fra le persone con il cognome che inizia con A-L; (iii) il voto medio delle persone con il cognome che inizia con M-Z, sia per lo scritto che per il voto finale (iv) il voto finale più alto fra le persone con il cognome che inizia con M-Z.

Esercizio 3 (“Consegnate! E’ ora!”):

Consegnare gli script e il sorgente del programma C, in attachment separati, entro il tempo a disposizione, via e-mail a: montresor@cs.unibo.it. Il subject del mail deve essere uguale a **LSO-PROVAPRATICA-4**, il nome dei file in attachment **devono contenere il vostro cognome** (per evitare confusioni in fase di correzione).