

From morsiani@cs.unibo.it Mon Jul 17 13:00:28 2006
Date: Mon, 17 Jul 2006 13:00:28 +0200
From: Mauro Morsiani <morsiani@cs.unibo.it>
Newsgroups: unibo.cs.labso
Subject: a grande richiesta: soluzione commentata prova pratica 14/07/2006 +
consigli utili

Salve, sono Mauro Morsiani.

Visti i risultati della prova pratica del 14/07, mi sembra evidente la necessita' di un approfondimento su questo tema, sia per illustrare le soluzioni ai vari esercizi, sia sulle strategie da utilizzare per la preparazione alla prova stessa.

* PREMESSA INDISPENSABILE

In UNIX "there's more than one way to do it": le soluzioni qui presentate sono solo alcune di quelle possibili ed egualmente valide (in fin dei conti, quello che importa e' l'aderenza alle specifiche richieste).

* Esercizio 1 (15 punti)

Scrivere la funzione:

```
envfork (int argc, char * argv[])
```

che accetta in input una lista di argomenti sul modello di quella del main() in C, dove ogni argomento è il nome di un comando, ed esegua ciascun comando in parallelo con execvp() limitando però l'environment di esecuzione al solo PATH=/bin

Esempio:

se argv = {"ls", "df", "who"} allora dovranno normalmente essere eseguiti in parallelo solo ls e df (che stanno in /bin) e non who (che sta in /usr/bin).

Rileggiamo il testo:

- i parametri vengono passati come se fosse un main() (quindi niente di complicato)
- viene richiesto "esegua ciascun comando in parallelo" (quindi un loop con una fork())
- è prescritto l'uso di execvp() (che ha la caratteristica di andarsi a cercare il comando nel PATH)
- la vera difficoltà e' la limitazione dell'environment (che va manipolata p.es. con un setenv())

Ecco una possibile soluzione:

```
#include <stdio.h>  
#include <unistd.h>
```

```

#include <stdlib.h>

int envfork(int argc, char *argv[])
{
    int i,pid;
    char *comando;

    for(i=0; i<argc; i++)
    {
        comando=argv[i];
        pid = fork();
        if (pid == 0)
        {
            setenv("PATH", "/bin", 1);
            execvp(comando, &comando);
            exit(0);
        }
    }
    exit(0);
}

int main(int argc, char *argv[])
{
    envfork(argc-1,&argv[1]);
    exit(0);
}

```

* Esercizio 2 (10 punti)

Scrivere uno script:

```
largefiles nome_directory
```

che accetti come parametro il nome di una directory e che produca in output una lista dei nomi e delle dimensioni in Kilobytes dei file contenuti nella directory e nelle relative sottodirectory, ordinata in ordine decrescente.

Esempio: `largefiles /boot` produce:

```

4575 /boot/initrd.img-2.6.11-1-686
4575 /boot/initrd.img-2.6.10-5-686-smp
4563 /boot/initrd.img-2.6.10-5-686
4334 /boot/initrd.img-2.6.10-5-386
1797 /boot/vmlinuz-2.6.11-sx280
1231 /boot/vmlinuz-2.6.10-5-686-smp
~E

```

Rileggiamo il testo:

- lo script prevede un parametro (\$1)

- bisogna estrarre "una lista... dei file contenuti nella directory e nelle relative sottodirectory" (quindi serve un comando che genera liste di file ricorsivamente, come `find` o `"ls -R"`)

- la lista deve contenere solo dei file (quindi "find -type f", o "ls -R" congiunto a qualche flag e filtro per distinguere i file dalle directory)
- bisogna trovare la dimensione in KB del file (quindi "du -k" oppure filtri o parametri per estrarre la dimensione es. da un "ls -l")
- bisogna ordinare la lista (sort) in ordine numerico (sono dimensioni) e decrescente (quindi "sort -gr" o "sort -nr")

Ecco alcune possibili soluzioni:

```
find $1 -type f -printf '%k %p\n' | sort -nr
```

```
find $1 -type f -exec du -k {} \; | sort -gr
```

* Esercizio 3 (8 punti)

Scrivere uno script:

```
grouprate nome_file
```

che accetta come parametro un file con una lista di gruppi e punteggi formattato usando ";" come separatore, tipo:

```
lso06az03;69,00
```

```
lso06az08;70,00
```

```
lso06az21;65,33
```

...

e produca in output una lista dei gruppi in ordine decrescente di punteggio, p.es.:

```
70,00 lso06az08
```

```
69,00 lso06az03
```

```
65,33 lso06az21
```

...

Rileggiamo il testo:

- lo script prevede un parametro (\$1)
- occorre dividere un'unica stringa con dei separatori (";") in tanti token (quindi si possono usare sed, cut o awk)
- occorre manipolare dei token (quindi si puo' usare awk)
- bisogna ordinare la lista in ordine numerico (sono voti) e decrescente (quindi sort)

Ecco alcune possibili soluzioni:

```
cat $1 | sed -e 's/;/ /g' | awk '{print $2 " " $1}' | sort -nr
```

```
awk -F';' '{print $2 " " $1}' $1 | sort -gr
```

* Esercizio 4 (15 punti)

Scrivere uno script:

```
mailall file_lista_utenti "Subject" file_messaggio
```

che accetti come parametri un file contenente una lista di e-mail, una stringa da usare come subject e un file di testo, e che mandi singolarmente a ciascun utente della lista una e-mail con il messaggio fornito e con il subject indicato.

Rileggiamo il testo:

- lo script prevede tre parametri (\$1 \$2 \$3)
- bisogna mandare un tot di e-mail distinte (quindi serve un loop, e bisogna usare il comando "mail" o "sendmail")
- il loop deve scandire le righe contenute in un file (quindi "for" e "cat")
- la mail deve essere spedita con determinate caratteristiche (quindi vanno capiti i parametri di "mail" o "sendmail")

Ecco alcune possibili soluzioni:

```
for i in `cat $1` ; do mail -s $2 $i < $3; done
```

```
for i in $(cat $1) ; do cat $3 | mail -s $2 $i ; done
```

*** ALCUNI CONSIGLI UTILI

Come annunciato anche prima della prova, i principali ostacoli in questo tipo di esame sono due:

- 1) la scarsa preparazione
- 2) un eccessivo nervosismo o "panico da esame"

Questi due fattori sono controllabili, vediamo come.

1) SCARSA PREPARAZIONE

Se si ignorano l'inglese, i rudimenti del C, dei comandi UNIX e del linguaggio di scripting bash, e' estremamente improbabile superare la prova.

Nessuno pretende che uno studente conosca a memoria tutti i parametri di ciascun comando o sia in grado di scrivere codice C di getto e senza errori di compilazione; ma e' perfettamente inutile presentarsi alla prova se non si e' fatto almeno un minimo di pratica.

La vera difficoltà della prova dal punto di vista delle competenze richieste, infatti, non sta nella quantità di cose da sapere (anche se agli studenti può sembrare il contrario...), ma nel tipo di conoscenza richiesta.

C'è infatti una bella differenza fra `_sapere_` (p.es. "il manuale dice che esiste un comando 'ls' per avere la lista dei file"), `_capire_` ("mi serve una lista di file, quindi uso ls"), e `_fare_` ("ls -lR" etc.)

La prova pratica richiede di dimostrare di saper `_fare_` qualcosa: quindi non basta aver studiato il manuale / i lucidi / etc., bisogna anche esercitarsi fino a che non si abbia acquisito sufficiente manualità'.

Come chiunque di voi che vada in palestra può facilmente testimoniare (dove sono finiti i geek di una volta :-), per metter su un bel fisico non basta pagare la quota mensile, avere la tessera, e leggere dei libri sul fitness: in palestra bisogna andarci e sudarselo :-)

Allo stesso modo, per acquisire la competenza richiesta per la prova pratica di LSO, oltre a studiare e' necessario esercitarsi, ed esercitarsi nel modo giusto: ecco alcune dritte...

* non si scappa: alcune cose fondamentali bisogna saperle a memoria (p.es.: la sintassi base del C e del bash, i comandi principali, tipo `ls cd mv...`); in questo senso per il ripasso sono utili manuali (tipo "Bignami") come i seguenti:

http://wks.uts.ohio-state.edu/basic_unix_guide/unix_guide.html

http://homepage.powerup.com.au/~squadron/linux_manual.pdf

<http://refcards.com/refcards/c/c-refcard-a4.pdf>

* il comando "man": bisogna esercitarsi a estrarre (`man -k / apropos, man -a`), leggere e interpretare le man page, per riuscire rapidamente a capire la sintassi di un comando, trovare il parametro che serve, vedere i casi particolari, etc.

* "prima si impara a leggere e poi a scrivere": studiare p.es. i compiti passati o l'infinita' di script che sono disponibili su Internet o su una qualunque macchina UNIX e' un buon modo per cominciare a maturare l'esperienza necessaria.

Attenzione: in questo caso "studiare" non vuol dire leggere il testo dello script e basta, ma capire come funziona, provarlo ed eventualmente modificarlo o riscriverlo in modo diverso...

* gli esercizi sono caratterizzati da dei pattern ben riconoscibili (p.es. "prendi in input un file, elaboralo, produci un output strutturato in un certo modo") a ogni elemento di questi pattern sono associati degli strumenti ben precisi (es. `cat, awk, sort`), bisogna imparare a riconoscerli

* "esercizio, esercizio, esercizio": ogni occasione e' buona per fare

pratica, p.es. anche Windows ha un'interfaccia a linea di comando, e i comandi sono simili a quelli UNIX: perche' non usare quella p.es. per spostare o rinominare i file, invece dell'Explorer?

* Google e bibliografia:

on-line sono disponibili da varie fonti sia le man page di tutti i comandi, p.es.:

<http://unixhelp.ed.ac.uk/CGI/man-cgi>

sia HOWTO e guide pratiche di ogni genere, sia esercizi da svolgere e gia' svolti di tutti i tipi:

<http://www.google.com/search?q=bash+shell+exercises>

<http://www.google.com/search?q=C+exercises>

Per esempio questi:

<http://www.doc.ic.ac.uk/~wjk/UnixIntro/index.html>

* rassegnatevi: se non sapete l'inglese, e' ora di impararlo...

Il tempo nella prova e' contingentato: questo significa che bisogna usare il tempo a disposizione in maniera efficiente. Ecco alcune dritte al proposito...

* visto che la prova pratica si concentra sul dimostrare di saper fare, e che ogni esercizio prevede un output ben preciso, concentratevi sull'essenziale: realizzare quell'output. Quindi:

- leggere attentamente il testo prima di cominciare, e focalizzarsi su quello che l'esercizio chiede, non su quello che voi vorreste che chiedesse (p.es. perche' conoscete la soluzione di un esercizio simile)

- se avete dubbi sull'interpretazione chiedete subito, non quando diventa troppo tardi

- niente controllo della validita' dei parametri se non e' esplicitamente richiesto (ogni riga in piu' richiede di essere scritta e debuggata, e la cosa vi portera' via piu' tempo di quello che credete)

- meglio usare una soluzione meno elegante ma che sapete far funzionare alla svelta, che una che richiede p.es. un comando o un parametro esotico (non esagerare, pero': "non e' che siccome sapete usare un martello che tutti i problemi diventano chiodi")

* preparate il vostro ambiente di lavoro: una succinta lista di URL indispensabili (es. man page on-line) gia' pronti nella barra del browser, un po' di appunti cartacei (niente di voluminoso), poche finestre (xterm e l'editor) gia' aperte, niente processi strani che girano

* editor: e' fondamentale. Sceglierne uno di cui siate sicuri che:

- ci sia su tutte le macchine del laboratorio
- funzioni bene (non crashi improvvisamente)
- sia facile e rapido da usare

Oltre a vi, la mia personale preferenza va a editor come pico o joe, che sono semplici, robusti, non richiedono X, e si usano via tastiera (quindi non si perde tempo a spostare la mano da tastiera a mouse e viceversa), e usano solo i tasti principali (quelli che non cambiano da una tastiera USA a ITA e viceversa)

* allenatevi a usare l'interfaccia a linea di comando e i comandi tipo "less" per copiare/spostare/esaminare i file; vi servira' anche per imparare a navigare le man page ("/parola" per cercare, spazio e "b" per muoversi nel testo, etc.) e per testare gli script

* le tastiere del laboratorio sono USA, fateci pratica (potete anche settare la vostra tastiera casalinga da ITA in USA)

* se vi si pianta la sessione X, non potete permettervi di perdere tempo ad aspettare che qualcuno ve la sblocchi: fate pratica di una procedura di emergenza per questi casi (p.es. switch su un VT testuale e kill di tutti i processi, magari con uno scriptino gia' pronto)

* esercitatevi alla consegna via mail (le modalita' sono sempre le stesse), per non perdere minuti preziosi e angustiarsi sul "come faccio a consegnare" durante la prova

* nessuno vi vieta di preparare gia' una directory per l'esame con degli "scheletri" di script o codice C gia' con i nomi pronti (es. <cognome>es1 etc.), i diritti giusti, gli #include classici del C o #!/bin/bash, e una mail vuota alla bisogna

* aver gia' pronto uno script che testa la compilazione di un modulo in C vi sara' sicuramente utile: perche' non lo sviluppate?

2) NERVOSISMO O PANICO DA ESAME

Se i compiti in classe e le interrogazioni di almeno (5 anni di elementari + 3 di medie + 5 di superiori + 1 o piu' di universita') non sono bastati a mettervi in grado di sostenere lo stress di una prova pratica come questa, beh, non e' che si possono fare miracoli...

Lo stress e' come il veleno di Mitridate: con sufficiente pratica, ci si

puo' abituare a produrre un risultato che risponda a determinati standard minimi in un tempo definito (a questo dovevano servire tutti quei compiti in classe...).

Se pero' la pratica fatta finora non vi e' bastata, allora vi possono aiutare il controllo, la preparazione e l'atteggiamento.

* "il controllo riduce la paura"

La prova prevede delle fasi aleatorie (lo svolgimento degli esercizi) e delle fasi che si ripetono sempre uguali (l'appello, il prendere posto con un documento e fare login, la lettura del testo degli esercizi, la consegna via mail, la chiamata per discutere quanto fatto): essere pronti e sapere cosa fare in ciascuna di queste fasi puo' aiutare non poco a gestire lo stress.

Esempi banali: perche' chi si presenta all'esame spesso non ha tutti i documenti che potrebbero essergli richiesti (es. libretto con foto, o tesserino + documento di identita')? Perche' la consegna via mail viene vissuta come una cosa cosi' difficile da fare?

Avere con se' tutti i documenti ed essersi esercitati nella consegna via mail sicuramente eviterebbe incertezze in queste fasi, e quindi ridurrebbe il carico di stress complessivo.

* "Sono pronto all'esame?"

In ogni prova di esame e' ovviamente insita una componente di incertezza, ma in generale, se la preparazione e' sufficiente, la prova viene superata.

E' necessario quindi capire se si e' sufficientemente pronti per superare l'esame; per fare questo bisogna essere in grado di misurare la propria performance.

Nel caso di una prova come questa, si puo' p.es. provare a svolgere completamente al computer, partendo da zero, un compito fra quelli passati (anche se ne conoscete la soluzione), senza pause e misurando il tempo impiegato: e' evidente che se ci si mette piu' del doppio del tempo consentito per totalizzare 30 punti, sara' ben difficile che durante la prova reale si ottengano risultati brillanti...

In alternativa, ci si puo' fermare allo scadere del tempo normalmente assegnato.

La verifica non si deve limitare alla misura dei tempi, puo' essere utile anche riprodurre l'ambiente e le condizioni di esame (p.es. fare la prova con i computer del laboratorio).

La difficolta' qui e' riuscire a mantenere una sufficiente disciplina per evitare facili scorciatoie ("questo lo so fare, passo al prossimo"), ed essere dotati di sufficiente onesta' intellettuale nel giudicare il proprio lavoro ("l'esercizio richiede 3 parametri e produce una lista di file, il mio script ne accetta uno e produce una lista di file e directory, va bene

lo stesso": al prof. non andra' bene per niente...).

Di solito, una o due prove di esame cosi' fatte sono piu' che sufficienti per valutare la propria preparazione in modo obiettivo.

* "Qual e' l'atteggiamento giusto?"

E' evidente che se vi recate all'esame per la centesima volta con l'angoscia di prendere 18 per passarlo, confidando nella vostra buona stella, siete fritti... Soprattutto perche' dopo due o tre tentativi l'intera faccenda vi risultera' insopportabile.

Meglio quindi accettare il fatto che per superare questa prova e' necessario lavorare sodo, e prepararsi al meglio delle proprie possibilita': alcune dritte su come farlo sono riportate sopra.

Il fatto di sapere come gestire le varie fasi dell'esame, e di aver misurato la vostra preparazione, vi aiuterà piu' di quanto immaginate.