Manuale di laboratorio per gli studenti di astronomia

\- - -

Versione 1.1

Diego Zuccato <diego.zuccato@unibo.it>

22 marzo 2011

Indice

Capitolo 1

Introduzione

Questo testo vuole essere una guida *semiseria* e *minimale* per guidare lo studente in quello strano mondo che sono i laboratori del Dipartimento, rispondendo a dubbi e domande ricorrenti e proponendo approcci efficaci a problemi apparentemente ardui (per esempio: "come impostare un documento complesso come una tesi in modo che sia comodo lavorarci").

Non vuole essere, invece, un sostituto del *regolamento per l'uso dei laboratori*, di cui dovrete prendere visione: è affisso proprio per essere sempre disponibile.

Non mi risulta che qui in ufficio siano mai stati mangiati degli studenti [1], quindi se avete qualche dubbio/problema/perplessità potete venire a chiedere senza timore.

Le vostre segnalazioni sono anche molto importanti per poter ripristinare il funzionamento delle macchine in caso di guasto. Tempo permettendo, di tanto in tanto si fa un giro a controllare se tutte le macchine si accendono, ma se il problema è più subdolo (per esempio una macchina non stampa dove dovrebbe) è difficile che ce ne accorgiamo e magari un problema risolvibile in 2 minuti rimane per 6 mesi solo perché nessuno lo ha segnalato. Non siate timidi!

Si ringraziano fin d'ora tutti gli utenti che, con consigli, esperienze e correzioni, aiuteranno a migliorare questo manuale.

La versione più recente di questo manuale è disponibile all'indirizzo
`http://software.bo.astro.it/documentazione/manuale_lab.pdf`

[1] Anche se, ogni tanto, notando mouse e tastiere staccati o portatili collegati al posto dei fissi, la tentazione di tagliare qualche manina c'è...

Capitolo 2

Problemi ricorrenti

2.1 Accesso

Intanto un chiarimento fondamentale: tutti i computer di un laboratorio sono equivalenti per l'accesso ai vostri dati. Possono esserci lievi differenze di potenza di calcolo o nel layout di tastiera, ma su tutti troverete i vostri dati. Quindi non disperatevi se il computer dove avete lavorato la volta precedente è in uso da parte di qualcun altro: sedetevi ad uno libero ed i vostri dati saranno lì ad aspettare che li maciniate come si deve.

2.1.1 Accensione/Login

A differenza di quello che succede sul PC di casa, quando arrivate in laboratorio ed accendete un computer, vi viene richiesto di identificarvi. Per farlo, è necessario usare le proprie *credenziali istituzionali* (fornitevi al momento dell'iscrizione come *nome.cognomeN@studio.unibo.it*) nel formato `studenti\nome.cognomeN` (senza @studio.unibo.it). La password è quella che avrete provveduto ad impostare sul sito `https://www.dsa.unibo.it`.

Vi ricordo che la password è strettamente personale e non va comunicata a nessun altro (tecnici compresi!). Chi ha la vostra password può agire a nome vostro e sarete voi a risponderne!

Talvolta le credenziali non vengono accettate appena si presenta la maschera di login. In questo caso, aspettate un minuto poi riprovate. Se ancora non vengono accettate, riavviate (cliccando l'apposita icona!) ed attendete un paio di minuti da quando compare la maschera di inserimento[1]. Raramente è necessario un secondo riavvio. Se però il problema persiste, deve proprio intervenire un tecnico.

[1] Sembra strano che un problema sparisca "senza fare nulla", ma in questo caso si tratta, normalmente, di un problema di mancanza di sincronizzazione dell'orologio interno col server: se il computer è stato spento a lungo, potrebbe esserci una differenza eccessiva, che viene "recuperata" da NTP mentre voi aspettate.

Una volta entrati, avrete accesso ad un vostro spazio "personale"[2] (la "home" locale[3], accessibile come `/home/local/STUDENTI/nome.cognomeN/` o, più brevemente, `~`), all'eventuale home "remota"[4] (accessibile in `/home/remote/STUDENTI/nome.cognomeN/`) e ad eventuali diretory condivise ("di gruppo", sotto `/home/local/STUDENTI/__GRUPPI__/`*`corso`*`/`*`NN`*`/` [5]).

In ognuna delle vostre home non dovreste tenere più dati del necessario: lo spazio (almeno quello sui dischi...) non è infinito. Verranno effettuati controlli automatici per rilevare occupazioni abnormi. Un limite ragionevole è 200MiB[6]. Nel caso abbiate bisogno di tenere dei file temporanei, conviene utilizzare `/tmp`. Questa cartella, infatti, è sul disco locale del computer su cui state lavorando, quindi risulta molto più veloce della home che invece deve essere raggiunta via rete. Inoltre viene automaticamente ripulita all'avvio e non penalizza gli altri utenti.

Mentre possiamo "chiudere un occhio" su campioni di dati astronomici (per es. file .fits molto ingombranti) purché non impediscano ad altri di lavorare, non verrà in alcun caso tollerata la presenza di files mp3 o di altro materiale comunque coperto da copyright.

2.1.2 Avviso di creazione home

Talvolta succede(va) che al momento del login comparisse una finestra con l'avviso di creazione della home.

Se non è il primo login in quel laboratorio, contattate immediatamente un amministratore senza *dare l'OK!* Significa infatti che il sistema non è stato in grado di trovare la vostra home e sta per ricrearla. Ovviamente vuota!

Se si è trattato di un problema temporaneo, al primo login che effettuerete dopo la risoluzione non troverete più i dati su cui avete lavorato tra la creazione della nuova home e la risoluzione del problema.

Oramai non dovrebbe più succedere, ma problemi di rete sono sempre possibili.

---

[2] Nel senso di 'non condiviso con altri', non nel senso che ci possiate mettere quel che vi pare!

[3] La vostra home si trova in ogni caso su un server, ed è quindi "remota", ma nel path viene considerata 'locale al laboratorio in cui vi trovate'

[4] Viene creata al primo accesso nell'altro laboratorio: se siete in Ranzani 11, la home remota (che si trova su un server in Ranzani 1) la potrete vedere solo dopo aver effettuato almeno un accesso dal laboratorio di Ranzani 1. E viceversa. Se non ci sono problemi di rete, potete sempre vedere i vostri dati sia locali che remoti. Ma in caso di problemi, vedrete solo quelli nella home locale! Ovviamente l'accesso ai dati locali è più veloce rispetto all'accesso a dati remoti.

[5] Solo da Ranzani 11. Da Ranzani 1 potrete accedere alle cartelle condivise solo da `/home/remote/STUDENTI/__GRUPPI__/`*`corso`*`/`*`NN`*`/`.

[6] Se il comando `du -m -s ~` stampa un valore superiore a 200, è sicuramente il caso di iniziare a fare pulizia

2.1.3 Sessioni incustodite

Troppo spesso vedo sessioni di lavoro lasciate aperte senza "proprietario" nelle vicinanze. Mi è addirittura capitato di trovare al lunedì una sessione lasciata aperta il venerdì!

Ricordatevi che nei laboratori dei vostri cugini ad Informatica, chi trova una sessione aperta ed incustodita *è esplicitamente autorizzato* a fare danni. Compresa la cancellazione totale dei dati dell'utente incauto. Dovremmo fare così anche qui?

Se proprio dovete allontanarvi per qualche minuto (la "pausa caffè"), ricordatevi di bloccare la sessione (quell'iconcina in basso a destra a forma di lucchetto serve proprio a questo). Se invece vi dovete allontanare per parecchio tempo, effettuate il logout e lasciate libero il computer per un altro utente.

2.1.4 Logout/Spegnimento

È assolutamente da evitare lo spegnimento "brutale" staccando la spina o tenendo premuto 4 secondi il tasto di accensione.

Bisogna uscire usando l'apposita icona o la voce nel menù principale. Se vi viene segnalato che c'è un altro utente collegato (e normalmente si tratta di un amministratore), *"non è simpatico"* procedere comunque con lo spegnimento. In certi casi la macchina potrebbe non riavviarsi!

2.1.5 Permessi su file e directory

In Unix, i permessi di accesso ad un oggetto sono differenziati per il proprietario, il gruppo e gli altri.

Il proprietario è colui che ha creato il file e che, normalmente, ne ha il pieno controllo.

Ogni utente può appartenere a parecchi gruppi (visualizzabili con `id`). Il primo gruppo di appartenenza (nel vostro caso `STUDENTI\domain_users`) è anche quello che viene assegnato di default ai file appena creati. Se volete evitare di dovervi ricordare di cambiare i permessi ai file quando lavorate sulla cartella di gruppo, potete ricordarvi di usare `newgrp corso_NN`.

I permessi "elementari" (nonché i più usati) sono:

read permette la lettura

write permette la scrittura

exec permette l'esecuzione

Attenzione al significato dei permessi nel caso delle directory: 'read' vuol dire che se ne può leggere il contenuto, 'exec' che ci si può accedere.
Potreste così avere una cartella accessibile ma non leggibile. Non è assurdo

come sembra: la vostra home non dovrebbe essere leggibile da altri che da voi, ma potreste avere una sottodirectory leggibile da chiunque. In effetti, succede regolarmente quando Apache, un server web, deve rendere disponibile la vostra `~/public_html` : Apache "gira" come normale utente non privilegiato. Se la home non fosse 'eseguibile', questo non sarebbe possibile. Però non è bello che chiunque possa vedere che file avete: per questo, normalmente, non è leggibile. Un altro esempio con cui probabilmente avrete a che fare è la "dropbox": una cartella in cui chiunque può 'depositare' file ma solo il proprietario può prelevarli. In pratica una "cassetta postale", spesso usata per la consegna di progetti senza intasare la mail del docente. Se avete capito come funzionano i permessi non dovreste avere problemi a capire come va settata.

Molti amministratori, spesso, rendono anche `/home` (o, nel nostro caso, `/home/local/*` e `/home/remote/*` non leggibili. La privacy ne guadagna (solo un amministratore può sapere se e quando vi siete collegati), ma l'usabilità ne risente: il completamento automatico non può funzionare con le directory non leggibili.

Il comando per cambiare i permessi è `chmod`. Per cambiare proprietario e gruppo si usano, rispettivamente, `chown` e `chgrp`. Attenzione se assegnate un nuovo proprietario! Potreste poi non riuscire più ad accedere al file! Per ulteriori informazioni rimando al par. 4.3.4.

Particolare cura va posta nella gestione dei permessi nelle cartelle di gruppo: ricordatevi di rendere i file accessibili ai vostri compagni impostando il gruppo corretto!

Altra cosa a cui fare *molta* attenzione è l'avere cartelle scrivibili da chiunque. È una pessima pratica: imparate ad usare correttamente i permessi e non vi servirà mai rendere qualcosa scrivibile da chiunque[7]. Probabilmente verrà aggiunto un test automatico per individuare e segnalare eventuali cartelle o file o+w.

È invece inutile dare estensione .exe ai file che compilate: in Unix quello che conta è l'attributo 'executable', non l'estensione. Normalmente (per la regola aurea che meno si scrive meno si rischiano errori), gli eseguibili sono senza estensione.

È possibile "nascondere" un file semplicemente facendone iniziare il nome con un punto. Non è una misura di sicurezza (`ls -a` visualizza anche i file nascosti), ma risulta utile per tenere pulita la lista file. Generalmente viene usato per i file di configurazione.

2.2 Alcune tastiere non scrivono bene!

Se quando premete le virgolette o gli apici non compare nulla, niente paura: si tratta di tastiere americane internazionali.

[7] A meno di non dover creare una dropbox.

In queste tastiere non compaiono le lettere accentate ed altri utili caratteri, ma è possibile ottenerli ugualmente premendo prima l'accento e poi il carattere da accentare. Se quello che volete ottenere è il solo accento, è sufficiente digitare l'accento seguito dallo spazio. Questo permette di ottenere anche simboli che non trovereste su una tastiera italiana (altrimenti come potreste scrivere correttamente Schrödinger? Con la tastiera internazionale basta che scriviate `Schr"odinger`).

Personalmente le preferisco a quelle italiane, soprattutto quando programmo, ma anche per la scrittura normale risultano molto comode.

2.3 Ci sono aggiornamenti: devo farli?

Beh, sarebbe carino almeno dare un'occhiata a quanti sono. Se non sono tanti e non hai troppa fretta, ti conviene farli. Se invece sono tanti è meglio se lasci che li facciamo noi tecnici... Di solito gli aggiornamenti grossi possono portare problemi, magari banali ma risolvibili solo da un amministratore.

È importante tenere il computer aggiornato, e proprio per questo viene data la possibilità a tutti gli utenti di applicarli.

Capitolo 3

LaTeX, questo sconosciuto

Ogni volta che c'è da creare un testo non banale, LaTeX è un'ottima scelta. Infatti permette di scrivere abbastanza "di getto", ma già dando una struttura alla propria opera.

Direte che lo fa anche Word, ma... le formule? Qualcosa tipo:

$$a^n = \overbrace{a \times a \times \cdots \times a}^{n \text{ volte}}$$

riuscite a renderla "presentabile"?

Non per niente LaTeX è lo standard con cui vengono creati gli articoli scientifici che leggerete tanto spesso. Meglio mettersi a studiarlo ora, risparmiando tanto tempo che poi servirà per altre cose (un paio d'ore passate ad imparare LaTeX si ripagano la prima volta che vi doveste trovare a modificare leggermente la struttura di un documento all'ultimo secondo... come succede regolarmente con la tesi).

Consiglio di usarlo anche per creare le slide per la presentazione della tesi: le formule saranno più leggibili ed il risultato finale più 'professionale'. Di più, al riguardo, nel cap. 5.

Se poi si aggiunge la semplicità di creare la bibliografia, mantenere coerenti i riferimenti incrociati ad altre parti del testo anche se si inseriscono o (ri)muovono interi capitoli, non c'è storia. E questo è solo l'inizio.

Per iniziare semplicemente ad usare LaTeX potete usare l'ottimo testo "Impara LaTeX (... e mettilo da parte)", liberamente scaricabile da
`ftp://ftp.pluto.linux.it/pub/pluto/ildp/misc/impara_latex/`
È anche disponibile ilforum del GuIT[1], dove potrete trovare utenti molto preparati e disponibili.

Se preferite iniziare in modo ancora più 'soft', potete utilizzare `kile`, un ottimo frontend grafico per i vostri documenti. Anche se probabilmente, una volta presa dimestichezza col formato, tornerete ad usare un editor più snello. 'Quale' editor dovrete deciderlo per conto vostro: se volete assistere ad una

[1] Gruppo Utilizzatori Italiani TeX e LaTeX, `http://www.guit.sssup.it/` .

vera e propria *guerra santa*, provate a chiedere in un'aula di informatici se sia meglio `vi` o `emacs`... Ma prima di farlo non dimenticatevi di indossare un ottimo giubbotto antiproiettile. Il mio consiglio è di provare vari editor, imparando a conoscerli quel tanto che basta per capire se vi ci potete trovare bene o no, e quindi focalizzarvi su uno per imparare ad usarlo al meglio.

Una volta imparate le basi, vi verrà voglia di automatizzare la generazione di documenti complessi, magari rigenerando solo il minimo indispensabile. Però servono vari comandi già nel caso semplice, e se ci si scorda di rigenerare il dvi prima di creare la versione finale della tesi? O, errore ancora più subdolo, ci si scorda la compilazione multipla per sistemare indici, bibliografia, liste di tabelle e figure? Beh, in questo caso vi conviene continuare a leggere, prestando molta attenzione all'uso dei makefile!

Capitolo 4

Tecniche di programmazione

4.1 I Makefile

Se con programmi banali avete messo tutto in un unico file, non potete fare lo stesso con progetti complessi. Oltretutto un unico file ha tante limitazioni:

- ci può lavorare una sola persona per volta;
- è difficile arrivare velocemente al punto che interessa;
- un piccolo errore (come un carattere digitato per sbaglio) può influenzare parti di codice ritenute stabili;

Questi (ed innumerevoli altri) problemi sono ben noti a chiunque abbia programmato in C, dove la separazione di un programma in varie unità di compilazione è spesso una necessità. Unix (e quindi anche Linux) è scritto in gran parte in C. È mai possibile che nessuno abbia studiato un modo elegante per risolvere questi problemi? Sarebbe veramente incredibile.

Ed infatti c'è un comando apposito: `make`. Il suo compito è, chiaramente, di "costruire" un programma partendo dai suoi componenti. Per una trattazione basilare (ma sufficiente per la maggior parte degli usi) consiglio `info make`.

Per chi è piuttosto pigro, includo anche il makefile usato per creare questo manuale. Se però provata a copiarlo ed incollarlo senza capirne almeno la sintassi, *probabilmente* non funzionerà...

```
1 TEX=latex
2
3 .PHONY: clean
4
5 manuale_lab.pdf: manuale_lab.dvi
6         dvipdf $<
7
```

```
8  %.dvi: %.tex %.toc Makefile
9          $(TEX) $<
10
11 %.toc: %.tex
12         $(TEX) $<
13
14 clean:
15         rm -f *.toc *.dvi *.aux *.log
```

Per rigenerare il manuale mi basta scrivere `make` ed il tutto viene aggiornato. E per ripulire tutto, alla fine, un bel `make clean` lascerà solo il sorgente ed il pdf.

L'uso di makefile per LaTeX non è particolarmente fondamentale (viene comunque ricompilato *tutto* il documento). Ma se nel documento dovete includere delle immagini generate da altri tool di analisi a partire da insiemi di dati e vostri programmi, ecco che potete iniziare a rendervi conto di quanto vi possa semplificare (e velocizzare) il lavoro un bel makefile.
Poniamo che il vostro documento contenga 3 immagini generate con un tool di analisi. E poniamo che per generare una di tali immagini siano necessarie 2 ore di calcolo. La prima compilazione, ovviamente, impiegherà 6 ore, e questo difficilmente si può evitare[1]. Ma una volta fatto tutto, vi accorgete che una delle immagini deve essere rigenerata perché avete usato un set di dati sbagliato.

Ops! Ecco che le varie soluzioni si differenziano:

Metodo manuale: Riscrivete prima il comando per generare la seconda immagine e poi quello per rigenerare il documento, sperando di non sbagliare qualcosa;

Script di shell: Rilanciate la generazione di *tutto*, comprese le due immagini che andavano bene, e perdete inutilmente 4 ore;

Script di shell (2): Prima di rilanciare lo script, lo editate per rimuovere la generazione delle immagini che vanno bene, lo lanciate ed al termine vi dovete ricordare di rimetterlo a posto;

Makefile: Lanciate *make* e tutto viene aggiornato automaticamente, senza dimenticanze e lasciandovi liberi di pensare al significato dei risultati ottenuti piuttosto che a *come* li avete ottenuti.

Quindi: a voi la scelta!

[1] Beh, potreste usare 3 PC per generare in parallelo le tre immagini iniziali... Ma solo se il laboratorio ha abbastanza postazioni libere!

4.2 Moduli

Un metodo molto potente si è sempre rivelato il *"divide et impera"*, ovvero "dividi e comanda". La programmazione non fa eccezione. Oltre ad essere più "maneggevoli", tanti file piccoli suddivisi per funzione risultano anche più chiari e permettono di isolare meglio gli immancabili errori.

Anche lavorando in Fortran potete "spezzare" il vostro programma in varie *unità funzionali*, compilarle usando lo switch `-c` ("compile only") e senza usare lo switch `-o outfile` . Verrà generato un file con lo stesso basename del vostro .f90 ma con estensione .o ("object"). I vari .o così ottenuti andranno poi uniti in un eseguibile. Se usate un makefile è semplicissimo:

```
FSOURCES := a.f90 b.f90
FOBJECTS := $(FSOURCES:.o=.f90)

ab: $FOBJECTS
        gfortran -o $@ $<

$FOBJECTS : %.o: %.f90
        gfortran -c $<
```

Questo compilerà `a.f90` in `a.o` e `b.f90` in `b.o`, quindi li unirà per formare l'eseguibile `ab`. Se vi servisse aggiungere un ulteriore file al progetto, basterà aggiungerlo alla variabile FSOURCES, senza toccare altro. Ed ovviamente rilanciare `make`. Modificando `a.f90` verranno rigenerati solo `a.o` ed `ab`, senza toccare `b.o` (o altri eventuali moduli).

4.3 Lavorare da linea di comando

Sembrerà arcaico in un'era di GUI 3D, ma molto spesso la linea di comando è l'interfaccia più efficiente con cui lavorare. Intanto non è necessario spostare le mani dalla tastiera al mouse, e già questo può far risparmiare parecchio tempo. E poi rende possibile lavorare attraverso connessioni lente, con ritardo elevato o a banda stretta. Un po' la differenza tra Word e LaTeX... solo esasperata.

Ci sono casi di elaborazioni *rallentate di 3 volte* nel caso si usasse l'interfaccia grafica al posto della linea di comando (nella fattispecie il render di un filmato che impiegava 9 ore lanciato da GUI, se lanciato da linea di comando ne impiegava si e no 3). Controllare se il mouse si è mosso, calcolare la percentuale di avanzamento ed aggiornare la barra, controllare se è stato premuto il tasto 'Annulla', ... può portare via parecchio tempo, specialmente se tutto viene fatto all'interno di un ciclo eseguito milioni di volte.

Oltretutto, che piaccia o no, normalmente da linea di comando si hanno a disposizione molte più funzioni che non da GUI, ed è anche molto semplice

automatizzare serie di operazioni (avete mai pensato di rinominare tutti e 500 i file di una cartella e delle sue sottocartelle, sostituendo gli spazi con underscore, rendendo maiuscola solo la prima lettera e minuscole le altre? È abbastanza semplice creare un comando[2] che lo faccia in un attimo tutte le volte che sia necessario, ma è molto difficile automatizzare una serie di azioni analoga in un file manager grafico).

Nella miriade di comandi disponibili, cerco di riassumere quelli che vi potranno risultare più utili e di uso più frequente. Ovviamente non riporto qui la documentazione completa, dato che potrete ottenerla direttamente mentre lavorate (vedi al par. 4.3.3).

4.3.1 Completamento automatico

Una funzione molto comoda (e che permette di evitare errori) è il *completamento automatico*: se in una directory avete `unFileConUnNomeLungo.f90` e `unAltroFileConUnNomeAncoraPiuLungo.f90`, per compilare il primo basta che scriviate `gfortran -o mioProgramma unF<tab>` (dove <tab> significa premere il tasto di tabulazione, quello con la doppia freccia sopra a Caps-Lock) per ritrovarvi con il nome completo del file desiderato. Se più di un file fosse iniziato per 'unF', il completamento avrebbe inserito solo la parte comune a tutti, aspettando poi un ulteriore carattere (per disambiguare) ed una nuova pressione di <tab> per terminare il completamento. Premendo <tab> due volte vengono visualizzati tutti i possibili completamenti.

Ricordatevi che gli operatori Unix ritengono che scrivere poco voglia dire lavorare più velocemente e più efficientemente. Tanto che, per molti comandi, è previsto il completamento automatico anche delle opzioni.

4.3.2 Le variabili

La shell ha la possibilità di memorizzare dati in variabili. Tali variabili non sono però tipizzate: quando vengono richiamate, il valore viene interpretato a seconda del contesto, dopo una sostituzione letterale.

Darò per sottointeso l'uso di Bash come shell predefinita. Nel caso ne usiate una diversa, consultate la relativa documentazione per i dettagli.

Quando volete assegnare un valore ad una variabile, è sufficiente scrivere `nome = valore` . Per utilizzare il valore assegnato, basta scrivere `$nome` o `${nome}` come in `echo $nome` .

Per eseguire conti, si usa `x = $(($x + 1))` . Attenzione sempre alla sostituzione letterale: se prima aveste scritto `x = a` potreste avere delle sorprese.

[2] In realtà una composizione di comandi: `find` e `sed`, in particolare. Il comando finale viene lasciato come esercizio: se non si è in grado di scriverlo, risulta quasi incomprensibile se si tenta di leggerlo.

È perfino possibile assegnare l'output di un programma ad una variabile usando `variabile = `comando`` o `variabile = $(comando)` .

Per la shell gli spazi separano i parametri. È quindi necessario prestare parecchia attenzione per evitare errori. Normalmente, è bene effettuare la sostituzione di variabili all'interno di ¨. In altri casi può risultare conveniente usare \ come carattere di escape quando c'è da inserire un unico spazio.

Le variabili vengono definite per la shell corrente. Quindi, se scrivete uno script di shell, le variabili che definite al suo interno non sono disponibili nella shell che usate per lanciarlo, a meno che non le esportiate con la direttiva `export variabile` .

4.3.3 Ottenere informazioni sui comandi

Per ogni comando, potete avere accesso alla documentazione (generalmente piuttosto esaustiva) semplicemente col comando `info`.

Un ottimo modo per iniziare a dare un'occhiata è utilizzare `info` per ottenere informazioni su come utilizzarlo: `info info`

È un'ottima idea, quasi *vitale* dare un'occhiata con `info` alla sintassi ed alle funzioni di ogni comando che si intende usare.

4.3.4 Comandi di shell molto utili

ls

Lista i file nella directory indicata. Li può filtrare in molti modi e risulta anche uno dei comandi più utilizzati singolarmente.

In mandriva è definito `ll` come alias per `ls -l`.

find

Permette di trovare i file con determinate caratteristiche: più nuovi di una certa data, più grandi di una certa dimensione, con un nome che rispetta una certa espressione, ecc.

Oltre a stamparli, può anche eseguire azioni sui file trovati.

screen

Permette di lasciare in esecuzione un processo anche dopo che si è eseguito il logout e di riprenderne il controllo quando ci si riconnette. Funziona solo coi comandi di shell. termina quando termina il comando lanciato, quindi se volete poter vedere eventuali messaggi di uscita è necessario che *prima* lanciate screen, *poi*, dalla nuova console, il comando che vi interessa.

Ottimo se dovete lanciare un'elaborazione piuttosto lunga. Ovviamente nel frattempo la macchina non dovrà venire spenta, quindi conviene lasciare un avviso a chi potrebbe utilizzarla nel frattempo (un bel foglio sulla tastiera

con scritto "Elaborazione in corso, si prega di non spegnere" dovrebbe essere sufficiente).

È da preferire rispetto al lasciare la sessione lockata perché:

- permette a chi magari deve solo controllare la posta di utilizzare la macchina e
- una sessione lockata a lungo potrebbe essere una semplice dimenticanza o, peggio, un modo di "tenersi" una macchina che non si sta usando: in entrambi i casi un utente che noti una situazione del genere in penuria di macchine libere può chiedere ad un amministratore di terminare la sessione lockata, con conseguente *perdita dei dati non salvati*.

Il comando `screen` "rompe" la simmetria delle macchine del laboratorio: per riprendere il controllo di quel processo è *necessario* ricollegarsi a quella macchina.

Se la macchina è in uso da un altro utente e dovete semplicemente controllare se l'elaborazione è terminata, chiedete all'utente attuale se vi può cedere un attimo la postazione. Non è necessario che si disconnetta: premete Ctrl-Alt-F1 per accedere ad una console testuale, fate login (ricordatevi che, a differenza del login grafico, quando inserite la password *non compare nulla*) e lanciate `screen -r` per riprendere il controllo della sessione. Una volta data una *veloce* occhiata, potete disconnettervi nuovamente, eventualmente uscendo dalla sessione di screen se il job è terminato. Per tornare all'ambiente grafico, basta premere Alt-F7 o Alt-F8.

ln

Crea un collegamento ad un file o una directory.

Il collegamento simbolico è un semplice "puntatore" ad un altro file o directory. Potreste per esempio voler accedere alla cartella del vostro gruppo di 'Elementi di Programmazione 1' da 'Documenti'. In questo caso potreste semplicemente scrivere qualcosa come

```
ln -s /home/local/__GRUPPI__/ep1/NN/ ~/Documenti/ep1
```

ed andando in Documenti/ep1 vi ritroverete nella cartella del vostro gruppo. Un link simbolico può puntare a qualsiasi cosa, anche a 'nulla' (un esempio di symlink che non punta a nulla è il lockfile di Firefox: viene usato un link per tenere traccia del processo attivo di Firefox).

Il collegamento fisico (hard link) è più difficile da comprendere. In pratica viene creato un altro nome per *lo stesso* file. Con `ls` l'unica indicazione che potete avere per identificare se un file è un hard link è l'indicazione del numero di link maggiore di 1. Modificando un hard link modificate il file originale!

Un hard link deve sempre puntare ad un file: non sono permessi hard link nè a directory (permetterebbero di creare loop) nè a file inesistenti. Inoltre un hard link può puntare solo file sullo stesso mountpoint.

Al 99.9% userete esclusivamente link simbolici.

Sui symlink non vanno impostati i permessi: vengono usati automaticamente quelli del file "puntato".

mc

Il Midnight Commander è un file manager che probabilmente oramai usano solo i dinosauri come il sottoscritto. Rende disponibili due pannelli per visualizzare in parallelo due directory e fornisce un'interfaccia veloce a molti comandi molto usati. Permette anche di accedere ad un sito ftp o ssh come se fosse una directory locale.

chown/chgrp/chmod

Questi comandi, anticipati al par. 2.1.5, modificano i permessi di accesso a file e directory. Generalmente `chgrp` viene usato poco, in favore di `chown :gruppo file`.

`chmod` invece si usa spesso. Particolarmente comoda è la forma mnemonica dei permessi: `chmod ugo+rx public_html` rende public_html leggibile ed accessibile (+**rx**) da chiunque (**u**ser, **g**roup, **o**thers). Se però dovete impostare una maschera esatta, è meglio la forma ottale: `chmod 0755 public_html` – apparentemente l'effetto è lo stesso del comando mnemonico, ma la maschera ottale sta anche specificando che il gruppo ed il resto del mondo *non* hanno accesso in scrittura. Con la forma mnemonica sarebbe stato necessario usare un secondo comando: `chmod go-w public_html` per essere sicuri di avere i permessi corretti.

4.3.5 Manipolare l'input e l'output dei programmi

La logica che rende apparentemente ostico l'uso della shell è che i comandi sembrano fare "poco". In realtà questo li rende i "mattoni" ideali (selezionati in quarant'anni di utilizzo!) per ottenere risultati complessi.

Alla base di questa possibilità di combinazione c'è la pipe (letteralmente "tubo"). Un programma per la shell, normalmente, prende il suo input da una pipe, lo elabora e lo riversa in un'altra pipe. Se, da bravi idraulici, si attacca una pipe di uscita ad una di ingresso, si ottiene un comando composito.

Per fare in modo che l'output di un comando venga usato come input per un altro, basta interporre tra i due il segno `|` (si chiama proprio pipe, la barra verticale che, sulla tastiera italiana, si trova in alto a sinistra, sopra a \).

Se si vuole che l'input di un comando sia un file, si usa la notazione `comando < file.in`. Se l'output del comando deve essere rediretto in un file, si usa la notazione `comando > file.out`.

more/less

Questi comandi servono alla fine della pipe, poiché interagiscono con l'utente paginando l'output e permettendo lo scorrimento (solo in avanti nel caso di `more`, anche all'indietro e con possibilità di ricerca nel caso di `less`).

sort

Ordina il suo input.

Attenzione se avete dei numeri allineati a sinistra: normalmente l'ordinamento è lessicografico, quindi 11 viene prima di 2...

head/tail

Estrae le prime (o le ultime) righe dell'input e le passa in output.

Se si specifica un numero negativo come parametro per -n, passa in output *tutto tranne* le prime (o le ultime) righe.

id/whoami

Ottiene informazioni sull'utente corrente.

`whoami` indica solo il nome di login dell'utente, mentre `id` indica anche tutti i gruppi di appartenenza (e può richiedere parecchi secondi!).

grep

Filtra, usando un'*espressione regolare*[3] o una semplice stringa, la sua pipe di input. Sulla pipe di output finiscono le sole linee che soddisfano il criterio di ricerca. Se viene usato al termine della catena, generalmente colora la parte di testo che ha soddisfatto il criterio di ricerca.

sed

Ovvero "Script EDitor". Nell'uso più comune permette di sostituire una regexp (o semplicemente una serie di caratteri) con un'altra.

Se volete sostituire tutte le occorrenze di 'Paperino' con 'Topolino' in `storia.txt` salvando la nuova versione in `altrastoria.txt`, basta scrivere `sed s/Paperino/Topolino/g <storia.txt >altrastoria.txt`

[3]Brevemente regexp (REGular EXPression). Scrivere regexp è quasi un'arte... Possono essere semplici o molto complesse, a seconda di ciò che si vuole ottenere. Se quel che leggete nelle pagine di `info` non vi è sufficiente, si possono trovare in rete interi trattati.

Capitolo 5

Tesi e relativa presentazione

Come già accennato nel cap. 3, consiglio di usare LaTeX anche per scrivere sia la tesi che la presentazione.

È invece fortemente sconsigliato l'uso di PowerPoint o di OO Presenter, poiché rischiate di perdervi in una marea di funzioni inutili in un pdf (animazioni, in particolare) e molto spesso il risultato finale lascia a desiderare (presenza di righe bianche, bordi ineliminabili, formule e grafici "sgranati", ecc.).

In particolare, con PowerPoint ci sono stati problemi di tutti i generi, compreso un file che a seconda del PC veniva visualizzato con lo sfondo, senza, o con solo una parte! Altri problemi comuni sono la mancanza di un font, o l'uso di immagini eccessivamente pesanti (assolutamente inutili: il dettaglio a video non aumenta, ma il tempo per passare da una pagina all'altra si!). La risoluzione in proiezione è, attualmente, di 1024x768: inutile impaginare per un 1920x1080!

Un'altra cosa molto importante (più per voi che per noi) è la puntualità nella consegna della presentazione: visto che i problemi sono sempre possibili, vogliamo andare in aula a provare la proiezione ed essere sicuri che non ci siano sorprese. Nel caso qualcosa non funzionasse, vogliamo darvi anche il tempo di apportare le eventuali correzioni che si rendessero necessarie.

Quindi i consigli sono:

- puntualità: la scadenza viene anche indicata sul televisore all'ingresso, ed è il termine ultimo perché una presentazione venga provata sul proiettore; *non* verranno *mai* accettate sostituzioni il giorno della presentazione se non autorizzate dal presidente della commissione di laurea

- il file da consegnare *deve* avere nome `Cognome Nome.pdf` e, se non è consegnato via mail, deve contenere l'indicazione dell'indirizzo email a cui inviare eventuali segnalazioni

- nel frontespizio riportate tutti i dati (titolo della tesi, nome e cognome, *data della presentazione*, nome del relatore e dell'eventuale co-relatore[1]
- durante la presentazione, le slide sono solo una traccia: non dovete semplicemente leggerle! Se una slide non può essere letta in 4-5 secondi vuol dire che avete scritto troppo
- cercate di tenere le cose semplici: sfondi multicolori con gradienti strani rendono la lettura più difficoltosa e distraggono dal contenuto
- il (poco) contrasto del proiettore "appiattisce" le sfumature e può rendere illeggibile un'immagine già di per sè poco contrastata
- evitate assolutamente gradienti chiaro-scuro, a meno di non voler usare un colore molto contrastante per il testo, e potrebbe risultare comunque poco leggibile
- ricordatevi il tempo che avete a disposizione: verrete avvisati un minuto prima del termine ed interrotti a tempo scaduto
- ricordate ad amici e parenti di rispettare l'edificio storico in cui vi trovate, e tenete presente che oltre a venirvi addebitate eventuali spese di pulizia, la vostra laurea deve ancora andare alla firma del Rettore...

[1] *Non* "correlatore", come tanti si ostinano a scrivere!