

Little X Manual



di Olivio Mariano

Versione 3.0



Il Little X Manual è distribuito in versione non stampabile. Per ricevere una versione completamente abilitata alla stampa sarà necessario inviare una e-mail di richiesta all' autore: jobs@libero.it



in collaborazione con



L'autore non si assumerà in alcun caso la responsabilità per eventuali danni diretti o indiretti riguardanti l'utilizzo proprio o improprio del presente, o qualora il suo uso violi gli accordi sottoscritti con Apple. Scritto a soli fini informativi.

Prefazione

Finalmente, dopo un pò di tempo, eccoci alla versione 3.0 del Little X Manual. Prima di ogni cosa ci tengo a scusarmi con tutti i lettori che, per mia mancanza, non hanno ricevuto una versione stampabile del piccolo manuale. I crescenti impegni non mi hanno permesso di dare continuità alla distribuzione delle copie stampabili né di effettuare un regolare aggiornamento di questa piccola guida.

Visto il discreto successo delle scorse edizioni, il Little X Manual cercherà di approfondire gli argomenti finora esposti. Come di consueto, ci limiteremo a dare spunto a chi vuole capirci qualcosa del terminale senza, per questo, dover affrontare libri ben più complessi.

Chi volesse poi cimentarsi con l'applicazione delle conoscenze acquisite potrà rivolgersi alla "Guida alla personalizzazione di Mac OS X" di Filippo Antonica (filippo_zeus@tin.it) un ottimo manuale per scoprire alcune tra le più interessanti caratteristiche di Mac OS X.

Sperando di essere, ancora una volta con questo manuale, di aiuto alla comunità Mac...

Olivio Mariano

Perchè Unix ?



Mac OS X si fonda su Unix, ormai lo sanno anche le pietre! Eppure, in molti casi, ancora non ci è chiaro perchè Apple abbia fatto una scelta così rivoluzionaria. Rivoluzionaria perchè, di fatto, il buon vecchio Mac OS 9 è logicamente e concettualmente diverso da quello che da qualche anno è diventato il sistema operativo ufficiale di casa Apple.

Possiamo dire che le motivazioni che hanno spinto Apple ad una scelta così radicale sono due. Una di tipo economico e una di tipo concettuale.

1. Nel 1996 Apple non era l'azienda che conosciamo. Era un'azienda in crisi che basava le sue macchine su un sistema operativo efficiente ma obsoleto tecnologicamente. Dopo aver abortito il compiuto progetto Copland Apple si trovò a scegliere tra l'acquisto di due software house per la concezione del suo nuovo sistema operativo. Be Inc. e NeXT Inc. La scelta cadde su NeXT proprio perchè il suo sistema operativo NeXTStep poggiava su fondamenta solide quali quelle di Unix.

2. La difficoltà o la facilità di utilizzo di un sistema operativo non ha senso se fundamentalmente non è stabile e poco evoluto. Non che Mac OS Classic fosse instabile, ma era obsoleto rispetto ai concorrenti Wintel. Unix, in questo senso, ha svecchiato l'interfaccia Mac OS donandole potenza, flessibilità, robustezza sotto il profilo tecnologico e sposando appieno la filosofia user-friendly che aveva contraddistinto i calcolatori Macintosh fino ad oggi.

Mac OS X : le basi di un sistema operativo



Per capire come classificare Mac OS X e la nostra piattaforma è necessario fare alcune premesse che renderanno più chiara la catalogazione del sistema operativo targato Apple. Iniziamo con il dire che tutti I computer digitali moderni sono concettualmente molto simili. Il fatto che abbiano diverse funzionalità, costi e prestazioni li caratterizza in:

- *Supercomputers*
- *Mainframes and minicomputers*
- *Personal computers*
- *Real-time computers*

Generalmente la complessità di un sistema operativo è quasi sempre legata alla complessità dell'elaboratore su cui verrà installato. Mac OS X è basato su un sistema operativo provenientedal mondo dei mainframe e dei minicomputer. Se vogliamo ,dunque, capire le differenze fondamentali tra il Mac OS Classic e la nuova generazione di Mac OS è necessario capire le filosofie che hanno guidato la distinzione tra personal computers e mainframe-microcomputer.

- *Mainframes & Minicomputers*

Unix fu compilato per la prima volta nel 1969. Sono trascorsi solo 15 anni da quando il primo Mac è apparso sul mercato. Capite quindi che la nascita del *computer for the rest of us* è avvenuta molto dopo la concezione del sistema operativo Unix.

Negli anni 60 / 70 l'elaboratore era inteso come *mainframes* atto ad essere usato da centinaia di persone contemporaneamente tramite terminali collegati ad un unico grande cervello che spesso occupava un intero piano delle grandi aziende cui forniva servizio.

I *minicomputer* erano delle versioni miniaturizzate dei *mainframes*. Costavano di meno e potevano supportare un massimo di venti utenti terminale. Questo tipo di sistemi erano ideali per organizzazioni di piccola-media grandezza e comunque non reggevano il confronto con i *mainframes*.

Da tutto ciò possiamo capire perchè si parla tanto di "multiutenza" . Unix nasce con il preciso intento di rendere possibile la cooperazione dei vari utenti che hanno accesso al sistema. Se molti utenti hanno accesso al sistema, nessuno di questi può occupare l'intera potenza processore per scopi che riguardano il singolo.

Questo implica che un sistema multiutente non possa mai andare in crash, poichè il lavoro di più utenti che traggono profitto dalla stessa macchina non può essere interrotto dal crashing dell'applicazione di un singolo utente.

Un sistema di questo tipo ha dunque 3 principali vantaggi:

- la multiutenza (ciascun utente possiede un spazio di lavoro dedicato al quale gli altri utenti non possono accedere)
- il multitasking (la possibilità di far operare il processore senza che un particolare utente -e dunque un determinato processo- occupi l'intera potenza macchina)
- la memoria protetta (più processi operanti sulla macchina non interferiscono)

no tra loro evitando di coinvolgere l'operatività dell'intero elaboratore se uno di questi va in crash)

- Personal computers

I personal computer proprio perchè "personal" e dedicati all'utenza consumer usavano sistemi operativi molto semplici che non avevano bisogno di molta potenza per operare, nè necessitavano di un sistema client-server (e dunque neanche di multiutenza e multitasking) per operare. Ovviamente tutto ciò era dovuto al fatto che queste tecnologie avevano un altissimo costo di produzione e sviluppo, cosa che non avrebbe permesso ai produttori di computers "personal" di allargare la loro fascia di vendita.

Cosa è Unix ?



Quello che è avvenuto negli ultimi anni ormai è storia. L'abbassamento dei costi di sviluppo e l'aumentata potenza processore hanno permesso che Unix (nelle sue varie distribuzioni) venisse integrato nei Personal Computers senza perdere le varie caratteristiche che ne avevano fatto un punto saldo nelle architetture client-server tipiche dei mainframes. Unix è, dunque, un sistema operativo. Il compito di un sistema operativo è quello di "orchestrare" le varie parti di un computer - il processore, la memoria, le tastiere, i monitors - affinchè compiano processi utili all'utente. Il sistema operativo è ciò che controlla tutto quello che avviene sull'elaboratore. Ogni cosa che l'utente chiede al proprio elaboratore viene assegnata al sistema operativo che deve eseguire il comando con il minimo apporto di errori. Il sistema operativo che si adatta di più a questa definizione, come avrete già capito, è Unix. Unix fu creato con l'intento di essere un sistema operativo *multiuten-*

te, multitasking, concepito espressamente per i programmatori. La filosofia che portò avanti lo sviluppo di Unix era quella di creare un sistema operativo di base semplice al quale potessero aggiungersi flessibilmente altre componenti atti a rendere il sistema operativo al passo coi tempi.

Il sistema operativo Unix comprende tre parti: il *Kernel*, i *programmi di utilità* e i *documenti di configurazione del sistema*.

- Kernel

Il kernel è il *core* di Unix. Il kernel è principalmente un grosso programma caricato nella memoria quando la macchina viene accesa, controlla l'allocazione delle risorse hardware da quel momento in avanti. Il *kernel* conosce quali risorse hardware sono disponibili e possiede i programmi necessari per "parlare" con le varie periferiche.

- I programmi di utilità

Questi piccoli programmi includono semplici utilità come "cp" -che copia i documenti- e altre più complesse come lo shell che permette all'utente di impartire comandi al sistema operativo.

- Controlli di configurazione

I documenti di configurazione sono letti dal *kernel* e alcune delle utilità standard. Il kernel Unix e le utilità sono programmi flessibili, e alcuni aspetti delle loro caratteristiche possono essere controllati cambiando i documenti di configurazione. Tipico esempio di un file di configurazione è il documento di sistema "fstab" che comunica al kernel dove trovare i files presenti sul disco rigido. Altro esempio è il documento di configurazione del *system log* che comunica al *kernel* come effettuare la registrazione dei vari eventi e

errori che si possono riscontrare durante la sessione di lavoro.

Metodi di accesso ad un sistema operativo Unix



Esistono diverse metodologie di accesso ad un sistema operativo Unix. Per far questo è necessario sempre e comunque iniziare una sessione di lavoro effettuando il login all'interno del sistema. Il metodo principale per iniziare una sessione di login su una macchina Unix è attraverso un terminale che include quasi sempre una tastiera e un monitor. Quando il terminale stabilisce una connessione con il sistema Unix, il kernel fa partire un processo chiamato *tty* per accettare l'input dal terminale e visualizzare l'output sul terminale. Quando il processo *tty* è creato è necessario che gli sia detto con che tipo di terminale ha a che fare. Se il processo *tty* non ricevesse questo tipo di informazioni non potrebbe comunicare correttamente con il terminale e di conseguenza l'utente potrebbe ottenere risultati non voluti.

- Console

Ogni sistema operativo Unix possiede una console principale connessa direttamente con la macchina su cui è installata. La console è un particolare tipo di terminale che viene riconosciuto quando il sistema è in funzione. Alcune operazioni relative al sistema possono essere attuate da console. E' per questo che è molto spesso accessibile solo dagli operatori del sistema e dagli amministratori.

- Dumb terminals

Alcuni terminali vengono denominati *muti* poichè possiedono risorse sufficienti solo per mandare caratteri come input al sistema e ricevere caratteri come output dal sistema. Molto spesso i "personal computer" possiedono pro-

grammi di emulazione terminale in modo da potersi connettere ad un sistema Unix in questa modalità.

I terminali "muti" possono essere connessi direttamente ad una macchina Unix, in remoto, attraverso un modem, un terminale server o altre connessioni network.

- Smart terminals

I terminali cosiddetti "smart", come l'X terminal, possono interagire con il sistema ad un livello più alto rispetto ai terminali "dumb". Questi hanno sufficiente memoria per provvedere un adeguato supporto alle interfacce grafiche. L'interazione tra un terminale di questo tipo e Unix può avvenire attraverso semplici caratteri quali icone, finestre, menu e azioni attraverso il mouse.

Terminale e shells



Il Terminal rappresenta l'interfaccia "primitiva" di OS X e di qualunque OS Unix-based. Di fatto è stata la prima in senso assoluto. Essendo la più anziana è quella che lavora sin dalle prime build di Unix e quindi è sicuramente quella che richiede minor impegno da parte della CPU grazie a un'interazione macchina-utente che si avvale esclusivamente del testo. Il fatto di non dover necessariamente caricare un'interfaccia grafica tipo punta e clicca diventa estremamente vantaggioso per macchine che non dispongono di cospicue risorse hardware.

L'applicazione *Terminal* (/Applications/Utilities/Terminal) rappresenta così una sorta di finestra sul "vero & puro" Unix. In questo modo è possibile inviare comandi al Subsystem BSD che ne riceve le istruzioni diretta-

mente. Con queste caratteristiche il Terminal diventa uno strumento molto potente di cui un amministratore può disporre. Se se ne hanno le capacità è teoricamente possibile fare di tutto via terminale, anche cancellare i file che regolano l'accesso al sistema. Comunque lo si utilizzi il Terminal difficilmente influirà sugli eventi dell'ambiente grafico e viceversa, proprio perchè si occupa esclusivamente dell'interazione utente-subsystem e non dell'interfaccia grafica, che ,in qualche modo, potrebbe influire sulle prestazioni della macchina utilizzata.

Il prompt



Prima di spiegare il funzionamento di qualsiasi comando della shell tcsh (default in Mac OS X) è necessario che si conosca bene l'importanza di ciò che ci appare a schermo al primo avvio del terminale. Ciò che ci troviamo di fronte all'avvio dell'applicazione *Terminal.app* è il *prompt*:

```
Last login: Tue Mar  4 06:19:35 on console
Welcome to Darwin!
[PowerBook:~] jobs%
```



La parte tra parentesi quadre indica il nome della macchina. Questa informazione risulta importantissima per chi ha intenzione di cimentarsi con il controllo remoto dell'elaboratore. Ci informa, di fatto, su quale macchina stiamo operando. In successione dopo i due punti troveremo la directory di utilizzo attuale (in gergo: *working directory*). Di default Mac OS X imposta come directory iniziale la *home directory* indicata dalla tilde. Fuori dalle parentesi quadre appare il nome utente attualmente operante sul terminale. Tutto ciò che si scrive sul terminale appare sulla destra del nome dell'utilizzatore subito dopo il caratterere "percento".

- Uso dei tasti freccia

In genere sono usati per riprendere i comandi già dati (su e giù). Freccia sinistra e freccia destra invece spostano il cursore lungo la linea di comando per rendere più semplice la correzione degli errori.

Logging in e logging out



Per assicurare la massima privacy a ciascun utente che ha accesso all'elaboratore le macchine Unix implementano un sistema per l'accesso separato di ciascun utente. La cosiddetta multiutenza permette di separare le risorse di ciascuna macchina e dunque rendere più efficiente e intelligente l'esecuzione dei vari processi consentendo oltretutto di amministrare in perfetta sicurezza la propria parte di sistema.

Per far ciò è necessario che il sistema operativo vi "riconosca" e dunque vi si chiederà di effettuare un'operazione di *login*. Vi starere chiedendo a cosa serve questa operazione dato che all'avvio del Terminale il prompt ci dice già da subito con quale nome siamo stati loggati. Questo succede perchè di default Mac OS X ha abilitato il login automatico che sarà necessario disabilitare se si vuole usufruire della multiutenza. A login automatico attivo o meno la fase di login sarà effettuata comunque nell'ambiente grafico e dunque all'avvio del terminale saremmo già autenticati. L'operazione di login via terminale (e duque in modalità testuale) si può effettuare, scegliendo, "Altri..." nella finestra di login e scrivendo nel campo destinato al nome utente ">console". A questo punto potermmo effettuare la procedura di login descritta.

- Logging in

Autenticarsi con un sistema operativo Unix richiede due informazioni fonda-

mentali. Un nome utente e una password. Quando si accede al terminale di Unix , il sistema operativo richiede queste due informazioni attraverso il prompt, che in questo caso apparirà così:

Login:



Dopo aver scritto il proprio nome utente e premuto il tasto invio il sistema vi richiederà una password. In questo caso quando scriverete il sistema non provvederà a visualizzare la password digitata.

- Il nome utente

La cosiddetta *username* è assegnata dalla persona (quindi un amministratore) che crea il vostro account. Mac OS X indicherà come username di default il vostro *short name*. La vostra *username* sarà unica e indicherà al sistema operativo come identificarvi.

- La password

Quando il vostro account sarà creato, anche una password sarà assegnata al nuovo utente. In questo caso è utile sapere come cambiare la propria password trovandosi di fronte alla linea di comando. Dopo essersi loggati è necessario digitare:

```
% Passwd
```



Il sistema vi chiederà di indicare la vecchia password e successivamente la nuova richiedendone poi la conferma. E' molto importante che si scelga una "buona" password in modo tale da salvaguardare l'accesso alla propria porzione di sistema. E' necessario per questo seguire alcune regole fondamentali:

- *Non usare nessuna parte del proprio nome o il nome di qualcun altro. Nè usare l'anagramma di alcun nome*
- *Non usare un numero facilmente reperibile tipo il proprio numero di telefono, il proprio indirizzo...*
- *Non usare alcuna parola reperibile in qualunque dizionario italiano o straniero*
- *Non usare tutte le stesse lettere, o una sequenza banale di queste*
- *L'ideale è l'utilizzo anche di caratteri maiuscoli e minuscoli insieme. L'importante è che si usino al massimo sei caratteri.*
- *Se si hanno diversi account su diverse macchine non usare sempre la stessa password su ogni macchina.*

- **Logging out**

Quando si è terminata la sessione di lavoro è necessario "deautenticarsi" digitando:

```
% Exit
```



Dopo aver lasciato il proprio terminale è necessario assicurarsi che venga visualizzata la prompt di login affinché si abbia l'assoluta certezza di essere stati effettivamente deautenticati dal sistema. Se alcuni processi non sono stati terminati prima del login out il sistema richiederà che questi vengano terminati. La shell di Mac OS X usa anche il comando

```
% Logout
```



Per effettuare la deautenticazione.

E' sempre buona norma cancellare ciò che è rimasto visualizzato sul terminale prima del logout. Potrebbero essere rimaste informazioni importanti tipo alcune informazioni a proposito di voi o a proposito del lavoro effettua-

to sul sistema. In questo caso sarà opportuno digitare il comando:

% Clear



Tipi di login



Il login che si compie all'avvio della macchina è quindi essenziale per la sicurezza dell'intero sistema. Su Mac OS X abbiamo tre tipi di login che corrispondono a tre particolari set di privilegi.

- *Superuser* - Il superutente è denominato anche "root". Il carattere loggato con questo nome ha privilegi assoluti su tutti i files e cartelle, comprese quelle del system che possono essere tranquillamente cancellate da questo tipo di utente.
- *Administrator* - L'amministratore è l'utente che viene creato al primo avvio di OS X . Questo utente ha il privilegio di modificare le impostazioni del sistema, ma gli è impossibile cancellare e modificare gli script di sistema che sono monopolio del superutente. L'utente amministratore è stato creato da Apple per evitare che l'utente alle prime armi possa inavvertitamente mettere mano ai files di sistema.
- *User* - L'utente "ospite" non ha alcun diritto sulle impostazioni della macchina. In definitiva deve sempre richiedere l'autorizzazione all'Administrator per un qualsiasi cambiamento sulle impostazioni predefinite.

Il superutente



Il cosiddetto super utente (detto anche *root*) è il carattere che può tutto in Unix. E' utile conoscerlo poichè permette la modifica di parametri nei documenti di sistema altrimenti impossibile con l'account creato da Mac OS X al primo avvio. Per abilitarlo sarà necessario utilizzare l'applicazione NetInfoManager (vedi in /Applications/Utilities/NetInfoManager). Utilizzando questa applicazione possiamo autenticare e assegnare al root una password.

Vediamo la procedura di autenticazione:

- In Mac OS 10.2.x, apriamo NetInfo Manager e scegliamo dal menu "Security" l'opzione "Enable root user".
- In Mac OS 10.x.x scegliamo dal menu "Domain" l'opzione "Security". Dal sottomenù scegliamo "Enable root user".
- Ci verrà chiesta la password da amministratore e poi quella da root.

A questo punto l'utente root è abilitato in tutte le sue funzioni.

Per richiamare questo tipo di utente via terminale sarà necessario utilizzare il comando su :

```
[PowerBook:~] jobs% su  
Password:  
[PowerBook:/Users/jobs] jobs#
```



- Utilizzare il comando `sudo`

Si tratta di un'utilità di sistema. "sudo" sta per "substitute-user do" e permette di accedere ai privilegi dell'utente "root" immediatamente. Per usare questo comando è sufficiente scrivere:

```
[PowerBook:~] jobs% sudo nome_comando
```



Il sistema richiederà una password di amministratore. Scritta la password di amministratore il sistema effettuerà il comando come se l'utente attivo fosse il super utente. Ricordiamo che con questo comando non è necessario abilitare l'utente root.

Abilitare il login di sicurezza open-firmware



Anche usando tutti questi accorgimenti vi domanderete per quale motivo Mac OS X è considerato un sistema operativo estremamente sicuro, se ,avendo Mac OS 9 installato, chiunque può accedere ai vostri documenti partendo, attraverso il tasto opzione, dalla partizione di Mac OS Classic o addirittura resettare le password con il cd di Mac OS X. Apple ha posto rimedio a questo inconveniente con l'ultima release del firmware (4.1.7 o 4.1.8), disponibile per i modelli di ultima generazione di Mac. Attraverso questo aggiornamento è possibile specificare una password per l'OpenFirmware.

Attenzione! Apple non supporta questa funzionalità, (come viene specificato nel TIL n°106292). In effetti, il solo modo per eliminare questa protezione, è di portare il proprio Mac presso un centro di assistenza autorizzato fuori qualunque tipo di garanzia (anche AppleCare). Anche il reset della PRAM (opzione+command+P+R) non risolverebbe il problema.

La protezione tramite password si effettua secondo tre metodi diversi:

- *Modalità sicurezza assoluta (full)* - che chiede la password ad ogni avvio, prima della scelta della partizione di boot.
- *Modalità sicurezza normale (command)* - che permette di partire senza alcuna password sulla partizione scelta, senza permettere alcun altro comando a livello di firmware (quindi alcuna possibilità di cambiare la partizione di boot senza la password)
- *Modalità di non-sicurezza (none)* - che non chiede alcuna password (è la modalità di default)

Consiglio personalmente la modalità "command" che vi permette un reboot della macchina in caso di problemi, senza aver bisogno di specificare una password ogni volta all'avvio, ma che impedisce ai non-amministratori di cambiare la partizione di avvio. In effetti, sotto Mac OS X, gli amministratori possono sempre selezionare una partizione di avvio senza sapere quale sia la password dell'OpenFirmware. Ecco la procedura da adottare per modificare i propri parametri Open-Firmware:

- *Partire sotto OpenFirmware* - Per avviare con OpenFirmware, spegnete la vostra macchina, successivamente riaccendetela tenendo premuto *option+command+O+F*. Vi ritroverete di lì a poco in modalità linea di comando con una tastiera di tipo QWERTY.
- *Specificare la password* - Scrivete il comando password e scrivete due volte consecutivamente la password.

• *Specificare la modalità di sicurezza - Scrivete il comando*

```
setenv security-mode full
```

per passare in modalità *full*

```
setenv security-mode command
```

per passare in modalità *command*

o ancora

```
setenv security-mode none
```

per disattivare tutti i tipi di protezione.

Basterà registrare e riavviare per verificare le modifiche effettuate.

Scrivete il comando:

```
reset-all
```

La shell Unix



La shell è il programma più importante in un sistema Unix. La shell è l'interfaccia con il sistema Unix, una sorta di traduttore che interpreta i comandi dell'utente e li invia al kernel. Possiamo definire la shell in questo modo:

1) La shell è un tipo di programma chiamato *interprete*. Un interprete opera

In maniera molto semplice: accetta il comando, lo interpreta, lo esegue e successivamente attende il prossimo comando. La shell visualizza il prompt lasciando intendere che attende il prossimo comando. La shell riconosce un numero limitato di comandi ed è per questo che è necessario scrivere i comandi in modo tale che li comprenda. Ogni comando consiste nel suo nome seguito dall'opzione preferita o dall'argomento del comando (sempre se desiderati). Il nome del comando, le opzioni e gli argomenti sono separati da uno spazio.

2) La shell è il programma che il kernel fa partire per primo, e in quanto tale è un *processo* come lo può essere qualunque applicazione installata sulla macchina in questione. La particolarità di tutto ciò sta nel fatto che il kernel può eseguire lo stesso programma (dunque anche la stessa shell) per ogni utente loggato.

Di fatto su ciascun sistema Unix è possibile aver installati diversi tipi di interpreti (quindi diverse modalità di esecuzione di un comando). Tra questi ricordiamo *sh*, *bash* (default sotto Linux), *csch*, *tcsh* (default sotto MacOS X), *zsh*. Questi *interpreti* vengono denominati comunemente shells in ambiente Unix. Fondamentalmente tutte le shell fanno lo stesso identico lavoro (magari con diversa sintassi) ma alcune sono più evolute di altre, in sostanza perchè sono state compilate per estendere le capacità delle shells più anziane.

Qualunque sia la shell utilizzata è importante ricordare che tutte le shells sono "case sensitive" e quindi non ci possiamo permettere di usare un comando con lettere maiuscole o minuscole a nostro piacimento. *LS*, *ls*, *Ls*. *LS* sono quattro comandi diversi sia che si usi *sh*, *csch*, *tcsh*... Le capacità di per-

sonalizzazione , di scripting possono distinguere le varie shells e il passaggio da una shell all'altra è necessario solo se l'esecuzione di una particolare azione non è supportata dalla shell in esecuzione.

Convenzioni e utilizzo



Bene, ora che sappiamo tutte queste belle cose dobbiamo incominciare a capire come ragiona la shell. Generalmente un comando Unix si presenta sempre sotto questa forma:

```
% Nomecomando [-options]
```



Il nome del comando è il nome del programma che la shell dovrà eseguire. L'opzione ci viene in genere suggerita dalla shell stessa e altera la funzionalità del comando in base al compito che l'utente vuole che esegua. L'argomento possono essere dei documenti, delle cartelle o anche programmi.

Provando a scrivere per esempio:

```
[PowerBook:~] jobs% cp
```



La nostra shell ci verrà in aiuto indicandoci quali sono le opzioni possibili per il comando.

```
[PowerBook:~] jobs% cp
usage: cp [-R [-H | -L | -P]] [-f | -i] [-p] src target
       cp [-R [-H | -L | -P]] [-f | -i] [-p] src1 ... srcN directory
```



Nell'utilizzare qualunque comando della shell può capitare di incappare in

errori. Niente paura. Se si vuole abortire l'esecuzione di un comando sarà sufficiente usare la combinazione di tasti *ctrl + c*

E' importante che nello scrivere i comandi se ne conoscano anche i limiti. Sintatticamente parlando Unix riconosce alcuni caratteri del tutto particolari chiamati anche *meta characters* come direttive del comando. Questo tipo di caratteri sono sempre riconosciuti dalla shell dovunque appaiano nella command line, anche se non preceduti da uno spazio. Proprio per questo è importante non usare questo tipo di caratteri come nome dei documenti con cui si lavora. Diventerebbe estremamente difficile "combattere" con la shell affinché capisca che un particolare *meta character* è una direttiva di comando ma fa parte del nome di un file. I caratteri cosiddetti "meta" sono principalmente questi:

`\ / < > ! $ % ^ & * | { } [] " ' ` ~ ;`

- Caratteri speciali

Il carattere speciale per eccellenza è lo spazio. Interpretare lo spazio come nel sistema operativo grafico porta sotto il terminal sempre ad un errore, specie nell'indicare una directory o un file che ne contengono parecchi.

Scrivere:

```
% cd nuova_cartella
```



difficilmente porterà nella directory nuova cartella. Per far interpretare alla shell il comando corretto è necessario scrivere:

```
% cd "nuova cartella"
```



oppure

```
% cd nuova\ cartella
```



Altri caratteri speciali molto utilizzati:

- *barra verticale* (`|`): reindirizzamento, reinstradamento (piping); di solito si usa per "instradare" l'output di un comando verso l'input di un altro comando. Per esempio:

```
# ps -ax
```



mostra una lista di processi attivi

```
# ps -ax | grep netinfo
```



`ps` viene eseguito esclusivamente in funzione di `netinfo`. In pratica filtra `ps` in modo che venga eseguito in funzione di `netinfo` stampandolo a video.

-*maggiore* (`>`): ha la stessa funzione della barra verticale ma il reinstradamento viene effettuato invece che a video verso un file.

Scrivere

```
# ps -ax >lista.txt
```



non mostrerà nulla a video poichè l'output viene trascritto nel file `lista.txt` nella directory corrente.

Usare gli editor di testo



Molto spesso usare il terminale richiede la conoscenza di un editor di testo. L'editor di testo che si presta ad un facile apprendimento più di altri è "pico". Per aprire un file di testo con pico sarà necessario scrivere:

```
% pico nome_file
```



In questo modo ci sarà possibile editare, ad esempio, il file `crontab` nella cartella `/etc`. Il comando per compiere tutto ciò apparirà, dunque, così:

```
[PowerBook:/private/etc] jobs% sudo pico crontab
```



Aperto il file, la finestra terminale dovrebbe apparire così:

```
Terminal — tcsh (tty1)
UV PICO(tm) 2.5      File: /etc/crontab      Modified

# /etc/crontab
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log
#
#minute hour   mday   month   wday   who   command
#
#*/5 * * * * root /usr/libexec/atrun
#
# Run daily/weekly/monthly jobs.
15 3 * * * root periodic daily
30 4 * * 6 root periodic weekly
30 5 1 * * root periodic monthly

⌘ Get Help  ⌘ WriteOut  ⌘ Read File  ⌘ Prev Pg  ⌘ Cut Text  ⌘ Cur Pos
⌘ Exit     ⌘ Justify   ⌘ Where is  ⌘ Next Pg  ⌘ UnCut Text ⌘ To Spell
```

Come possiamo notare l'interfaccia è molto semplice e ci permette di

muoverci con l'ausilio dei tasti freccia. Nella parte bassa della schermata pico ci elenca i vari comandi disponibili per l'editing del testo. Il carattere "^" rappresenta il tasto control. In questo modo per passare alla pagina successiva seguiremo il suggerimento di pico che ci indica il comando `ctrl + V`. Se vogliamo avere un breve riassunto dei vari comandi disponibili con pico possiamo usare il comando `ctrl + G`.

Nel nostro caso modificando il documento `crontab` (magari cambiando gli orari per attuare particolari comandi) utilizzeremo il comando `ctrl + X`. A questo punto la parte inferiore della finestra dovrebbe apparire così:



Pico ci chiede o meno il salvataggio delle modifiche effettuate. Premeremo "y" per yes e "n" per no. Il procedimento è lo stesso per qualunque file di sistema e configurazione, ovviamente non dimenticando mai di autenticarci come "root" se, come in questo caso (`crontab`), si tratta di documenti accessibili unicamente dagli amministratori di sistema.

Comandi utili per l'apprendimento



- `man`

E' forse uno dei comandi più usati per chi si avvicina per la prima volta al terminale Unix. E' dunque una convenzione conoscerne l'esistenza. Questo comando da infatti accesso alla documentazione in linea ("man" sta per manual). Scrivendo:

```
% man [capitolo] nome_del_comando
```



Se nome-del-comando è un comando valido (naturalmente la sua man page deve esistere), viene mostrata la documentazione relativa.

Ricordiamo che le man pages non sono solo adibite alla descrizione dei comandi, ma anche alla descrizione di molti files di configurazione. Se si scrive:

```
% man [capitolo] rc
```



Si otterrà la documentazione in linea del file di booting "rc".

In qualunque caso il valore "capitolo" è un numero che indica in quale parte della man page si vuole ricercare l'informazione voluta. Omettendo questo valore il comando man mostrerà direttamente la prima pagina del manuale.

Il "valore capitolo" è importante poichè in ambito Unix ci si trova spesso davanti a frasi del tipo per dettagli vedere nomadelcomando(6), dove il valore tra parentesi corrisponde al capitolo delle man pages da consultare.

Come "regole" di consultazione ricordiamo che per andare avanti nello scorrimento del testo delle man pages è sufficiente tener premuto il tasto invio mentre per "sfogliare" le pages si può premere la barra spaziatrice.

Chi prende consiglio dalle *man pages* molto spesso desidera salvare su disco e tenere sempre a portata di mano il manuale ricavato.

In questo caso si pone un problema. Il testo delle man pages è formattato in maniera tutta particolare. Non possiamo dunque fare un semplice copia e incolla poichè il testo incollato sarebbe in definitiva poco leggibile.

Occorre oltre che filtrare l'output di man attraverso il comando "col" anche instradarlo in un file di testo specifico. Ad esempio:

```
% man cp | col -b >testomanuale.txt
```



Il file `testomanuale.txt` conterrà le informazioni desiderate correttamente formattate.

•`whatis`

Dopo l'introduzione del comando `man` ci sarà utile apprendere l'uso del comando "whatis". Quest'ultimo infatti riporta informazioni sintetiche su un comando, quali le man pages esistenti e il suo funzionamento in generale. Il comando `whatis` crea il suo database partendo da man pages già esistenti e disponibili sul computer.

•`apropos`

In correlazione con `whatis`, riporta tutte le ricorrenze del testo inserito. Scrivere

```
% apropos cp
```



troverà tutte le ricorrenze di `cp` nel database di `whatis`. Il comando risulta molto comodo quando si vuole verificare l'esistenza di quel comando in più man pages (in diversi capitoli, ad esempio).

La struttura del filesystem

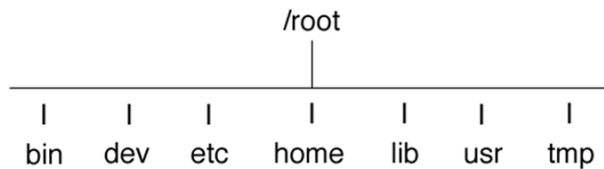


Tutte i dati inseriti in un computer a fondamento Unix sono custoditi nel *filesystem*. Tutte le volte che iniziamo una sessione di login in una shell, la shell ci colloca da qualche parte nel *filesystem*.

Il luogo in cui veniamo collocati nel filesystem è chiamato *working directory*.

Il filesystem di Unix è dunque di tipo gerarchico simile ad un albero. L'albero

è piantato nel terreno in un luogo chiamato *root* (radice) chiamata "/". Ogni elemento che troviamo in questo catalogo gerarchico è o una cartella o un documento.



Per specificare un particolare documento o cartella è necessario che si specifichi anche un percorso per giungerci. Il percorso partirà sempre dalla root del catalogo gerarchico per arrivare al file o cartella voluta. Ogni directory nella sequenza sarà seguita da "/".

Ricordiamo che il simbolo "/" indica anche la root directory. Non confondiamo dunque il carattere "/" usato ad inizio sequenza per indicare la root directory e il carattere "/" usato solamente per separare le cartelle del catalogo gerarchico.

- Cartelle "." e ".."

Il singolo punto indica la directory corrente mentre i due punti indicano la directory che contiene la directory in cui si sta lavorando. Scrivere:

```
% ./pwd
```



significa eseguire il programma `pwd` escludendo con `./` tutte le directory all'infuori di quella in cui si sta operando. Questo comando è molto importante se sappiamo che sulla nostra macchina risiedono programmi con lo stesso nome. Potrebbe capitare di lavorare su un programma che risiede in tutt'al-

tra directory. E' dunque buona norma specificare sempre la directory nella quale si sa che risiede il programma da eseguire.

- Il tasto tab

Per rendere la navigazione delle cartelle più veloce e flessibile la shell tcsh usa il tasto tab per autocompletare quello che si sta scrivendo. In altre parole scrivendo parte del nome di una directory o file e poi premere tab significa far completare sintatticamente il nome. Se la directory che si vuole non coincide in alcun modo con la porzione di nome già scritta la shell emetterà un beep di errore. In ogni caso se il nome da completare è una directory, il nome verrà completato con una barra alla fine indicando di fatto la presenza di un file o cartella al suo interno.

Muoversi nel filesystem



In Unix il proprio "luogo di lavoro" è chiamato *current working directory*. Se si vuole sapere dove ci si trova è necessario digitare:

```
% pwd
```



che significa *print working directory*. Quando si ha intenzione di cambiare la propria directory di lavoro è necessario usare il comando:

```
cd nome_directory
```



che significa *change directory* seguito dal nome della cartella in cui si vuole arrivare. L'argomento può anche essere un percorso che definisce la directory nella quale si vuole lavorare.

Scrivere semplicemente

```
% cd
```



nel prompt vuol dire chiedere alla shell di reindirizzarci direttamente nella nostra home-directory. Scrivere:

```
cd ..
```



significa chiedere alla shell di scalare di un livello la gerarchia delle directory. Quando si è nella cartella voluta possiamo chiedere al terminale di dirci cosa vi è dentro scrivendo:

```
% ls
```



che significa *list*. In questo caso possiamo chiedere alla shell altre informazioni o modalità di elencazione attraverso una particolare variabile di cui possiamo trovare descrizione nelle man pages scrivendo:

```
% man ls
```



Per esempio scrivendo:

```
% ls -lR /lib/l*
```



l'opzione "I" e "R" sono invocate insieme. L'opzione "I" richiede un *long output* e l'opzione "R" chiede a ls di listare le directory scendendo la gerarchia. L'ultima parte del comando " /lib/l*" chiede al comando di listare documenti

e cartelle cominciati per "l" nella cartella /lib . Il carattere * significa *qualunque carattere*, dunque tutte le cartelle cominciati per "l" e con qualunque seguito di caratteri.

Privilegi di accesso



Nella scorsa edizione del manuale avevo inserito una sezione sui privilegi di accesso che a molti è parsa un pò criptica. Diciamo intanto che dall'introduzione di Mac OS X 10.2.x "Jaguar" il tutto può essere eseguito attraverso l'interfaccia grafica affidandosi alla finestra informazioni di un documento o cartella. Ma se volessimo ostinarci a farlo via terminale ?

Partiamo con il dire che Unix affida ad ogni file o cartella informazioni relative al suo proprietario e ai privilegi di accesso. Il proprietario è definito come colui che può cambiare i privilegi intesi in Unix come:

- Read (r--)

Per un documento, il privilegio "lettura" permette di leggere il contenuto del documento. Nel caso di una cartella ne permette l'esplorazione.

- Write (-w-)

Per un documento, il privilegio "scrittura" permette di modificare il contenuto del documento. Nel caso di una cartella il privilegio di scrittura permette di alterarne il contenuto aggiungendo o eliminando documenti o sottocartelle.

- Execute (--x)

Per un documento il privilegio "esegui" permette la sua esecuzione se si tratta di un programma o di uno script. Questo privilegio è dunque irrilevante per i documenti non eseguibili. Nel caso di una cartella il privilegio "esegui" permette di eseguire i file legati a quella specifica cartella. In questo caso il privilegio di esecuzione limita i due privilegi di lettura scrittura

Quando si hanno tutti e tre i privilegi di accesso la sigla sarà "rwx" e via via per le varie combinazioni delle tre lettere.

Unix permette anche che gli utenti vengano definiti in gruppi, in modo da rendere la gestione dei privilegi più semplice e immediata.

Le entità presenti nell'uso dei privilegi di accesso sono dunque:

- *Proprietario*

Il proprietario (*owner*) è quasi sempre l'utente che ha creato il file. Quasi tutti i documenti e cartelle nella propria Home Directory avranno come proprietario l'utente loggato in quel momento.

- *Gruppo*

Gli utenti *admin* su Mac OS X sono membri di gruppi chiamati "admin" e "wheel", mentre il super-utente (root - da loggare con il comando "su") è membro del gruppi chiamati *admin* e *staff*. Quasi tutti i documenti e le cartelle sono assegnate a uno di questi tre gruppi.

- *Altri*

Altri (*Others*) si riferisce a tutti quegli utenti che non sono nè utenti nè fanno parte di alcun gruppo.

Modificare i privilegi di accesso



Per conoscere i privilegi di accesso ad un documento o cartella possiamo usare il comando "ls" (descritto in un'altra sezione di questo manuale) con l'opzione "-l".

Per esempio:

```
% ls -l /etc/passwd
```



dovrebbe dare questo tipo di output:

```
-rw-r--r-- 1 root wheel 722 Jul 14 2002 /etc/passwd
```

I primi 10 caratteri ci forniscono informazioni a proposito dei privilegi di accesso. Vediamo come.

Il primo carattere (in questo caso "-") indica il tipo di documento.

Convenzionalmente:

- d sta per *directory*
- s sta per *special file*
- - sta per *regular file*

rw-

I prossimi tre caratteri (nel nostro caso "rw-") descrivono i privilegi relativi al suo possessore. Nel nostro caso potrà dunque leggere e scrivere il file ma non eseguirlo.

r--

Questi caratteri ("r--") descrivono i privilegi per tutti gli utenti appartenenti allo stesso gruppo del proprietario. Nel nostro caso possibilità di lettura ma non di scrittura nè di esecuzione.

r--

I restanti caratteri descrivono i privilegi per tutti gli altri utenti. Nel nostro caso lettura ma non scrittura nè esecuzione.

Bene, ora che sappiamo come leggere i privilegi di accesso ci possiamo permettere effettuare dei cambiamenti su essi. Il comando che ci permette di fare ciò è

% chmod



Il comando chmod richiede che gli venga specificato il nuovo privilegio da assegnare e il documento o cartella al quale assegnarlo. Per settare il privilegio è necessario che si usi la notazione indicata "rwx" per specificare il tipo di privilegio e la cosiddetta "ugo notation" per specificare dove è necessario applicarlo

Per definire il tipo di cambiamento voluto è necessario usare il segno "+" per aggiungere un privilegio, il segno "-" per rimuovere un privilegio e il segno "=" per assegnare il privilegio direttamente.

Ad esempio, volendo cambiare i privilegi al documento .shrc nella nostra cartella personale (*home directory*) possiamo usare questa stringa:

```
% chmod g=rw- ~/.shrc
```



In questo esempio stiamo specificando un privilegio di lettura e scrittura per il gruppo senza possibilità di esecuzione.

Possiamo combinare privilegi multipli nei nostri comandi separando ogni operazione con una virgola, ovviamente non usando alcuno spazio. Ad esempio:

```
% chmod u=rwx,go=r-x lollo
```



setterà i privilegi "rwx" per il proprietario del documento "lollo", mentre assegnerà al gruppo privilegi "r-x"

Esplorare il contenuto di un documento



•Il comando "file"

I file di testo usati da Unix sono di due tipi. File di testo di tipo "diretto" e di tipo "indiretto" (richiedenti l'ausilio dell'elaboratore per la loro interpretazione). Quando ci troviamo a lavorare con un documento, un comando importante per comprendere con quale dei due tipi di file stiamo interagendo è:

```
% file/percorso/file/
```



Per esempio:

```
% file /bin/sh
```



dovrebbe darci questo tipo di output:

```
% /bin/sh: Mach-O executable ppc
```

In questo caso la shell ci informa che il file in questione (il file "sh" stesso) è eseguibile e contenente istruzioni di tipo binario, quindi eseguibili direttamente dall'elaboratore.

•Il comando "cat"

Il comando "cat" concatena i documenti inviandoli sullo schermo. Si possono specificare , dunque, uno o più documenti come argomenti. In alcuni casi il comando cat ci permetterà di formattare il testo.

Ad esempio:

```
% cat /etc/inetd.conf
```



Visualizzerà a schermo il contenuto del file di configurazione "inetd.conf"

•Il comando more

Il comando "more" mostra a schermo un file di testo una pagina per volta, adattando il testo alla finestra del terminale. Premendo il tasto "a capo" possiamo scorrere il testo una linea per volta oppure ,premendo la barra spaziatrice una pagina alla volta. In ogni momento possiamo uscire dal testo premendo il tasto "q".

Ad esempio, scrivendo:

```
% more /etc/rc
```



ci verrà visualizzato lo script rc pagina per pagina.

- I comandi head e tail

Il comando "head" ci permette di visualizzare la parte iniziale di un file specificando il numero di linee da visualizzare. Se non lo si specifica il comando sceglierà di default 10 linee.

Scrivere:

```
% head -15 /etc/rc
```



significa mostrare a schermo le prime 15 linee del file /etc/rc

Il comando tail compie esattamente lo stesso compito del comando head visualizzando ,però, le linee finali del documento in questione

Scrivere:

```
% tail /etc/rc
```



mostrerà le ultime 10 linee del file /etc/rc. In questo caso non abbiamo però specificato il numero di linee. Il documento, dunque, ci mostrerà, come di default le ultime 10 linee del documento.

Copiare documenti e cartelle



Il comando per la copia di cartelle o documenti é cp. La sintassi per l'uso di questo comando è:

```
% cp file_sorgente file_destinazione
```



In qualunque caso il comando cp ,prima di creare una copia del file voluto, osserverà se vi è già un documento o una cartella con la stessa destinazione. Se esiste un file sovrascriverà il file già esistente. Se esiste una cartella la shell metterà la copia in quella cartella e nominerà la nuova copia come l'originale.

Spostare e rinominare documenti e cartelle



Il comando per spostare i documenti e le cartelle in Unix è "mv". La sintassi è somigliante a quella di cp:

```
% mv file_cartella cartella
```



Con lo stesso comando è possibile cambiare il nome di una cartella o un file. Scrivere:

```
% mv pippo.doc paperino.doc
```



significa cambiare il nome del file da pippo.doc in paperino.doc. Questo può avvenire esclusivamente se il luogo in cui avviene il cambiamento è sempre lo

stesso e non ne viene specificato altro.

Rimuovere documenti e cartelle



Il comando usato per cancellare i documenti è rm. La sintassi di utilizzo è:

```
% rm nome_file
```



Nell'uso di questo comando si possono includere diversi documenti separandoli con uno spazio.

Scrivendo:

```
% rm pippo.doc paperino.doc pluto.doc
```



si chiederà alla shell la rimozione dei file elencati.

Per la cancellazione di una cartella è possibile l'uso di due comandi:

```
% rmdir nome_cartella
```



oppure

```
rm -r nome_cartella
```



Nel primo caso il comando non funzionerà se la cartella non sarà completamente vuota. Nel secondo caso il comando rm con opzione -r prima cancellerà il contenuto della cartella e poi la cartella stessa

Creare una cartella



Il comando per la creazione di una cartella vuota è `mkdir`. La sintassi usata è:

```
% mkdir nome_cartella
```



Per creare la cartella voluta in un luogo specifico è necessario che prima del nome della nuova cartella venga specificato il percorso di posizionamento. Se questo non viene specificato la shell darà per scontato che la cartella deve essere creata nella propria `working directory`.

Cercare documenti e cartelle



Esiste in Unix un comando complementare a Sherlock. Immaginate di voler ritrovare tutti i documenti appartenenti all'utente Nina che hanno più di 7 giorni. Con Sherlock l'impresa rischia di diventare abbastanza ardua! Con il comando `find` la ricerca si rivelerà di una facilità estrema. Scriviamo nel terminale:

```
% find / -user Nina +mtime 6 -print
```



Il comando significa: ricerca (`find`), a partire dalla root (`/`), i documenti che appartengono all'utente Nina (`-user Nina`), la cui data di ultima modifica è superiore ai 7 giorni (`-6` e `0` per oggi - si parla di 24 ore a partire dalla data corrente) e stampa il risultato a schermo (`-print`).

Il comando risulta comunque molto flessibile. Immaginiamo ,ad esempio, un programma che crea documenti di log molto grandi e che noi vogliamo comunque conservare nell'arco di sette giorni.

```
% find /var/tmp -name '*.log' +mtime 6 -exec rm - {} \
```



Il comando significa: ricerca (find), a partire da /var/tmp, i documenti che si chiamano nomefile.log (*.log) e la cui data di ultima modifica è superiore a sette giorni (+mtime 6). In più, richiediamo di eseguire (-exec) il comando rm sui documenti che si sono trovati precedentemente (-). Le parentesi {} servono a passare gli argomenti al secondo comando (in questo caso rm); infine (\) termina il comando.

Metodi di accesso via rete



I sistemi Unix furono concepiti nei primi anni per funzionare solo attraverso il supporto di una rete. Non per niente i sistemi Unix erano utilizzati prevalentemente nelle accademie negli ambienti di ricerca. Una particolare implementazione fu utilizzata dalla DARPA (*Department of Defense's Advanced Research Projects Administration*) che di fatto gettò le basi per il futuro internet,

• FTP

Il protocollo FTP (File Transfer Protocol) fornisce all'utente un metodo semplice per il trasferimento dei documenti da un sistema Unix. L'accesso via FTP ad una macchina Unix può avvenire attraverso il login di un utente registrato oppure può essere anonimo. Una sessione di lavoro FPT prevede un particolare e limitato set di comandi con il quale è possibile manipolare i docu-

menti da trasferire.

Per usare il protocollo di trasferimento ftp in Mac OS X è necessario scrivere:

```
% ftp 10.0.0.1
```



L'host da contattare in questo caso è un numero di ip ma potrebbe essere anche un dominio. Usando questo comando possiamo interpellare il server ftp su altre macchine Mac OS X o Unix . Questo è facilmente attivabile dalle preferenze di sistema spuntando l'opzione "Attiva l'accesso FTP" nelle Preferenze Network.

Ricordiamo comunque che i suggerimenti qui presenti sono da applicare esclusivamente a Mac OS X e non a Mac OS X Server, che ha altre modalità per l'attivazione del server FTP.

Attivato il server FTP attraverso l'interfaccia grafica saremo in grado di far connettere tramite questo protocollo solo gli utenti creati sulla propria macchina. In effetti ,per ragioni di sicurezza, alcuni accounts non sono autorizzati a connettersi via FTP, anche se esistono e sono funzionanti . Gli accounts non autorizzati sono listati nel file

```
% /etc/ftpusers
```



IL file sarà accessibile tramite l'applicazione pico, o qualunque altro editor di tipo esadecimale.

Rimane comunque una sorta di anomalia nell'uitlizzo del server FTP sotto OS

X. Quando ci si connette a un server FTP in genere si accede tramite l'account "anonymous". Come attivare questo tipo di account sul nostro Mac ?

Per far ciò sarà necessario:

- scegliere una cartella "root" per l'account "anonymous"
- creare e parametrare una cartella "root"
- creare un utente FTP

La scelta della cartella "root" dell'account "anonymous" è importante proprio perchè è necessario che gli utenti "anonymous" non possano esplorare liberamente il contenuto del proprio disco.

Quindi creiamo la cartella

```
/Library/FTPRoot/
```



In seguito sarà necessario attribuire dei privilegi a questa cartella per bloccare l'utente "anonymous" in caso dovesse cercare di modificare o cancellare i files sul disco. Come mostrato in figura, digitiamo:

```
[PowerBook:~] jobs% su  
Password:  
[PowerBook:/Users/jobs] jobs# mkdir /Library/FTPRoot/  
[PowerBook:/Users/jobs] jobs# chmod 555 /Library/FTPRoot  
[PowerBook:/Users/jobs] jobs#
```



Per creare un utente FTP, è necessario scegliere un numero utente (e eventualmente un gruppo) FTP. Ad esempio scegliamo come UID "666". Per verificare che l'UID sia libero (e quindi disponibile per la creazione e la modifica) digitiamo questo comando:

```
[PowerBook:/Users/jobs] jobs# nidump passwd / | cut -f3 -d: | sort -n  
-2  
0  
1  
25  
70  
74  
75  
99  
501  
502  
[PowerBook:/Users/jobs] jobs#
```

Il risultato ottenuto stamperà a schermo gli UID già utilizzati. Se verificiamo la non esistenza di "666" nella lista, allora lo possiamo utilizzare, se invece appare nella lista sarà necessario cambiarlo e sostituirlo al posto di 666 nel comando seguente:

```
echo "ftp*:666:666::0:0:FTP User:/Library/FTPRoot:/dev/null" |sudo niload passwd
```



Per verificare il funzionamento dell'account *anonymous*, connettiamoci come *anonymous* alla nostra macchina.

Note

- Il protocollo FTP non è un protocollo sicuro: le password vengono trasferite sulla rete in chiaro. Per evitare ciò sarà necessario utilizzare sftp. Per utilizzare sftp non avremo bisogno di attivare FTP, ma è necessario attivare l'accesso SSH (nella Sharing Preferences, spuntare "Allow remote login") e utilizzare un client sftp (su Mac OS X, il comando sftp è un client sftp)
- Con ftpd, solo gli utenti root possono connettersi

- Con Mac OS X Server il server FTP è un wu-ftpd, anche se ftpd è presente
- Per utilizzare un server FTP le manipolazioni delle impostazioni sono abbastanza semplici.

Installato un programma server FTP sarà necessario editare il file `/etc/inetd.conf` e modificare la linea seguente corrispondente al servizio FTP.

```
% ftp stream tcp nowait root /usr/libexec/tcpd ftpd -l
```



Rimpiazziamo "ftpd -l" con il comando per lanciare il server FTP (leggere le istruzioni del server usato per sapere quale sia il comando)

La procedura è la stessa in caso si voglia cambiare l'accesso al server ftpd.

• Telnet

Telnet è concepito per gli utenti che vogliono effettuare un login su una macchina connessa ad un network. Effettuata la sessione di telnet viene iniziata la normale procedura di login.

In Mac OS X il controllo remoto via telnet è anche possibile via ssh altro protocollo più sicuro e veloce. In entrambi i casi la modalità di funzionamento sarà questa:

```
[PowerBook:~] jobs% telnet 10.0.0.1  
Trying 10.0.0.1...
```



In questo caso telnet cerca di connettersi all'host 10.0.0.1 informadoci di un'eventuale avvenuta comunicazione.

Nota: qualunque protocollo si utilizzi (*ftp, telnet, ssh...*) la navigazione all'interno del file system della macchina remota è perfettamente uguale a quella che si effettua sulla propria (dunque implica l'uso dei comandi `cd`, `ls`, `rm`,

mv...)

- http

l'http è il protocollo per l'interscambio di pagine HTML nel World Wide Web. I server http che forniscono il servizio sono quasi sempre accessibili pubblicamente. In alcuni casi, l'accesso ai documenti attraverso server http richiede una qualche forma di autenticazione.

- Https

Una variante dell'http. La "s" finale sta per "secure". Questo perchè le connessioni http sono considerate estremamente vulnerabili. Le connessioni effettuate attraverso il protocollo https comprendono un particolare schema di encrypting per le informazioni scambiate.