Introduzione a Linux

Carlo Pinciroli

19 novembre 2003

Indice

1 Che cos'é Linux

Linux è un sistema operativo derivato da UNIX, e perciò conserva molte delle caratteristiche di quest'ultimo, tra cui le più importanti sono:

La multiutenza: più utenti possono accedere al sistema (che può essere formato da un terminale o da una rete di terminali) indipendentemente l'uno dall'altro; in altri termini l'attività di un utente non può compromettere l'attività di un altro utente (ad esempio l'utente "gianni" non può cancellare i dati di "pinotto" né modificarli, senza il consenso di "pinotto"). Per garantire tutto questo, il sistema operativo prevede dei veri e propri diritti (di lettura, scrittura ed esecuzione) da applicare a file e directory: in questo modo chi non possiede i diritti necessari non può intervenire sul file o sulla directory che vorrebbe. Nessun utente normale può modificare i file di configurazione e di avvio del sistema, per evidenti ragioni di sicurezza: è dunque necessaria l'esistenza di un utente speciale, denominato root (in inglese "radice"), che ha diritti illimitati sul sitema: l'utente root funge quindi da amministratore. Una conseguenza diretta di tutto ciò è la necessità di identificare l'utente al lavoro, al fine di definire i suoi diritti sul sistema: questa identificazione viene effettuata all'atto del login, senza il quale non si può avere accesso alle risorse del sistema. Durante il login vengono richiesti nome utente e password.

Il multitasking: da task, in inglese "compito". È la possibilità di eseguire più programmi contemporaneamente, senza che un programma possa influenzarne un altro: il blocco di un prorgamma in Linux (e in UNIX, naturalmente) non pregiudica il funzionamento di tutto il sistema come di norma accade in Windows quando appaiono le famose schermate blu. Per questo Linux è considerato un sistema stabile.

Nonostante storicamente Linux interagisse con l'utente per lo più attraverso la console (l'analogo del "Prompt di MS-DOS" di Windows), esiste anche un ambiente grafico, simile a quello Microsoft, denominato X Window. Può avere diversi aspetti grafici e diversi modi di essere utilizzato, perchè diversi sono i gestori delle finestre (Window Manager in inglese), tra i quali spiccano per fama e semplicità d'uso KDE e GNOME.

Una delle più famose e, al tempo stesso, positive caratteristiche di Linux è la libertà di distribuire i sorgenti del sistema operativo (ovvero il sistema operativo non ancora eseguibile, quindi modificabile a piacimento dall'utente al fine di porre migliorie o correggere eventuali errori), e quindi di copiare sul proprio computer Linux semplicemente scaricandolo da Internet. Dal momento però che la grandezza media di un download di questo tipo è piuttosto consistente, e che

la scelta delle parti da scaricare e poi installare è vasta e piuttosto difficoltosa, alcune aziende hanno pensato di fornire su CD-ROM il sistema operativo già pronto da installare con le parti del sistema necessarie, i programmi di uso più comune e una procedura di installazione e configurazione semplificata. Queste distribuzioni su CD-ROM (i cui nomi sono, per citarne alcune: RedHat, Mandrake, SuSe, Debian, Slackware...) sono facilmente reperibili, a bassissimo prezzo (spesso meno di 5 euro) e differiscono fra di loro per l'hardware supportato, la scelta dei programmi e la posizione di certi file di configurazione del sistema.

2 Concetti di base sul funzionamento di Linux

Il filesystem è il complesso dei meccanismi che gestiscono file e directory del sistema: attraverso di esso, è possibile creare, modificare, cancellare e spostare file e directory. Cosìcome Windows possiede un filesystem, anche Linux possiede il proprio. In Linux però il concetto di file è piuttosto particolare, perché è molto più astratto che in Windows. Infatti si usa dire che in Linux "tutto è un file", ovvero qualunque parte fisica o logica del computer (che sia un disco, una scheda sonora o di rete, un modem, uno schermo, ecc.) viene gestita dal sistema come se fosse un insieme di file. Ad esempio, per stampare un documento, in effetti Linux scrive su un file che rappresenta la stampante; è poi il "sottobosco" del sistema che si occupa di tradurre questa brutale scrittura in segnali comprensibili alla periferica. In altre parole, Linux vede tutto come se fosse un file. Per poter gestire una periferica, bisogna quindi far sìche l'insieme dei file relativi alla periferica stessa faccia parte della struttura delle directory di Linux: questa operazione di aggiunta di file a quelli di Linux è detta montaggio (in inglese mounting). Quindi, se una cosa non è montata, Linux non la vede, e quindi non può essere utilizzata. Un esempio evidente di tutto questo si ottiene quando si cerca di tradurre un indirizzo di un file sotto Windows (ad es. `C:\WINDOWS\WIN.INI`) in uno per Linux. Tanto per cominciare, sotto Linux non esiste l'espressione A: o C:, perché le varie unità a disco fanno parte della struttura delle directory, se sono già montate. Il montaggio di un'unità a disco fa sìche l'intero contenuto del disco appaia come una sottodirectory, quindi ad esempio se l'hard disk C: è stato montato nella directory `/mnt/winc`, l'indirizzo in formato Linux sarà `/mnt/winc/windows/win.ini` Attenzione: Linux è case sensitive quando tratta i nomi di file, quindi `WIN.INI` e `win.ini` sono due file differenti.

La struttura delle directory di Linux è abbastanza standard (alcune sottodirectory cambiano da distribuzione a distribuzione)

- `/home` : qui vengono messi i file personali degli utenti. Ogni utente ha una sua sottodirectory in cui può fare ciò che vuole: ad esempio "gianni" ha `/home/gianni` e "pinotto" ha `/home/pinotto.` L'analogo Windows è più o meno `C:\Documenti`.
- `/root` : la directory personale di "root"
- `/etc` : contiene molti file di configurazione del sistema e di vari programmi. Un pò come `C:\Windows\Command` o `C:\Windows\System`.
- `/usr` : contiene i file dei programmi installati, i manuali e molto altro. Più o meno come la `C:\Programmi`.

- `/dev` : contiene tutti i file che rappresentano una periferica, ad esempio
  - `/dev/hda` è l'hard disk primario (il C: di windows, di solito)
  - `/dev/hdb` è il drive secondario, slave di `/dev/hda` (quindi o un hard disk o un cdrom)
  - `/dev/hda1` indica la prima partizione di C:
  - `/dev/hda3` è la terza partizione di C:
  - `/dev/ttsy0` è la porta COM1
  - `/dev/ttsy1` è la porta COM2

  e così via.

3 Introduzione, struttura di base di X e avvio

X è il più famoso sistema grafico per Unix, e quindi anche per Linux. Sono sinonimi i termini X, X Window e X Window System, mentre la dicitura "X Windows" è errata.

La struttura di X è organizzata a strati, di cui il più basso rappresenta l'hardware del computer e il più alto le applicazioni. L'hardware (di solito solo tastiera + mouse + scheda video + schermo) è gestito da X attraverso un SERVER, ovvero un "servitore", che ha il compito di offrire le funzionalità grafiche utilizzate poi dai CLIENT. I client (in italiano clienti) sono infatti programmi che usano l'ambiente grafico comunicando con il server X e facendo uso di librerie chiamate XLIB. Al fine di garantire la comunicazione fra server e client sono state predisposte delle regole di linguaggio, raccolte in un protocollo detto con scarsa fantasia PROTOCOLLO X. Il rapporto esistente fra server e client di X ricalca quello che sussiste in ambiente di rete, ed infatti il server X fra le funzionalità garantite ha anche quella di permettere una connessione da remoto. Per chiarire con un'immagine il rapporto che sussiste fra server e client, si può pensare che il server sia un venditore che tiene aperto il proprio negozio in attesa di clienti; che i clienti si presentino o meno, il negozio rimane aperto, ed appena un cliente si presenta viene subito servito se è possibile farlo. Il protocollo è in un certo senso la "lingua" (italiano, inglese...) che viene utilizzata per comunicare dal negoziante e dal cliente. Una collezione di serventi grafici molto famosa è XFree86: ogni servente grafico è specifico per un tipo di scheda video, ma non mancano serventi grafici per schede video generiche. I serventi grafici possono quindi essere considerati come delle specie di "driver", per dirla con una similitudine con Windows, ma come abbiamo visto i server fanno molto di più di quello che fanno i driver in ambiente Microsoft. I client sono invece i programmi, che siano un gioco, Microsoft Office, o altro.

Tra i programmi clienti fondamentali, una particolare menzione merita il gestore delle finestre (Window Manager). È un programma che mette a disposizione le funzionalità che l'utente necessita per poter usare il sistema grafico, quindi ad esempio una barra come la "Avvio" di Windows, le caselline per ridimensionare/spostare/chiudere le finestre, e oggetti simili. Tra i più famosi Window Manager ricordiamo KDE, Gnome, fvwm con le varianti fvwm2, e fvwm95-2, twm, Afterstep e IceWM. È importante però sapere che è possibile utilizzare X anche senza un gestore delle finestre, anche se questo fa parte di argomenti che esulano dalla trattazione attuale.

Una delle differenze più sorprendenti che X ha nei confronti di Windows è data dal fatto che in X esistono degli schermi virtuali. Anche se lo schermo fisico (quello che si sta guardando) è uno soltanto, X permette di gestire più schermi e di passare da uno schermo ad un altro con la stessa facilità con cui si passa da un programma ad un altro. La quantità di schermi virtuali disponibili dipende dalle impostazioni che si sono adottate. Un'altra differenza con Windows, ma meno marcata, è la necessità di un mouse a tre tasti per X, quando invece Windows richiede solo un mouse a due tasti. Molte operazioni (fra cui quelle di copia-incolla) sono spesso legate al tasto centrale del mouse, e quindi non averlo costringe a configurare X in modo particolare, al fine di emulare il tasto centrale mancante tramite la pressione contemporanea dei due tasti del mouse.

È possibile impostare il boot di linux per avviare X all'avvio, nel qual caso il login che si presenta è di tipo grafico. Altrimenti, se si esegue del lavoro in ambiente testuale (console), l'avvio di X si ottiene col comando startx.

4 Programmi di uso comune

La quantità di applicativi per X è considerevole, e non è possibile né in questa sede né in futuro coprirne l'intera lista. Ci limiteremo a fornire le possibili alternative ai più famosi programmi per Windows offerti da X per Linux con KDE come window manager. Naturalmente anche Gnome o gli altri window manager offrono applicativi analoghi, alcuni dei quali sono comuni a tutti i window manager in quanto vengono forniti insieme a XFree86. Almeno per quanto riguarda KDE, l'utilizzo dei programmi è molto simile allo stile di Windows. Unica vera differenza, in Linux non esiste il "doppio clic".

4.1 Avvio di programmi

Le modalità sono le stesse di Windows: si può utlizzare la barra "K" oppure le icone sul desktop, ma senza fare uso del doppio clic!

4.2 Operazioni sulle finestre

Sulla barra del titolo a destra ci sono tre icone, che come in Windows permettono rispettivamente di ridurre a icona, massimizzare e chiudere la finestra. A sinistra del titolo invece troviamo prima l'icona del programma a cui la finestra si riferisce: cliccando su di esso compare un menù che permette di fare le stesse operazioni delle icone a destra. L'altra icona tra quella menzionata e il testo del titolo, se cliccata, fa sì che quella finestra compaia in tutti gli schermi virtuali.

4.3 Personalizzazione dell'ambiente

È possibile effettuarla tramite il KDE Control Center, navigando fra i menù a tendina.

4.4 Operazioni sui file

Konqueror può essere tranquillamente paragonato a "Esplora risorse" (o "Gestione risorse") di Windows e ne ricalca le modalità di utilizzo. Per un analogo di Winzip ci si può rivolgere a GnoZip. xterm è un terminale testuale emulato.

Quando si parlerà dei comandi della console, si vedrà l'utlità di poter accedere ad un terminale virtuale. Al momento ci limitiamo a paragonarlo al "Prompt di MS-DOS" di Windows. Una delle comodità di Windows in fase di installazione è data da InstallShield, e anche X offre delle funzionalità analoghe (ma non identiche),anche se diverse da distribuzione a distribuzione: nella RedHat e nella Mandrake sono offerti gli archivi RPM che si possono gestire con kpackage.

4.5 Lavoro d'ufficio

Se quello di cui si ha bisogno è Office, Linux offre i notevoli OpenOffice.org e KOffice. Il primo dei due è fra l'altro identico al sosia sotto Windows ed è completamente compatibile con esso. Anche le modalità di utilizzo e i nomi delle opzioni sono praticamente gli stessi.

4.6 Internet

L'Accesso remoto è "tradotto" in KPPP sotto Linux, il gestore della posta Outlook in KMail e i browser disponibili sono Netscape/Mozilla, Opera (disponibili sia per Windows che per Linux), Konqueror (come Explorer, per KDE) e Galeon (per Gnome).

4.7 Multimedia

Per quanto riguarda la musica e gli mp3, xmms è il parente Unix di Winamp; la grafica è invece maneggiabile con programmi come gimp (fotoritocco), gqview, blender (grafica 3d); anche OpenOffice.org offre programmi di questo genere.

5 Conclusioni

Per concludere, non è difficile convincersi che praticamente ogni cosa si possa fare con un programma a pagamento per Windows sia tranquillamente realizzabile in Linux con programmi Open Source. L'unica (relativa) falla di Linux è la scarsità di giochi, se paragonata alla considerevole quantità di videogames disponibili per la piattaforma Microsoft.

6 Cenni generali sul concetto di shell

Il programma più importante in un sistema operativo, dopo il kernel, è l'ambiente in cui è possibile interagire con l'hardware del computer. In realtà il dialogo con l'hardware è già garantito dal kernel, e a questo è dovuta la sua fondamentale importanza; quello che rimane all'utente è quindi il compito di impartire comandi al kernel. Questa attività è piuttosto complessa, e quindi si usa racchiudere il kernel stesso in una "conchiglia" (in inglese shell) con la quale l'utente interagisce effettivamente. In altre parole, la shell è l'ambiente che permette all'utente di interagire con il kernel mettendo a disposizione comandi comprensibili dall'uomo e interpretabili dal kernel. Una volta interpretati, i comandi vengono mandati all'hardware che si occupa di eseguirli. Una shell dunque può essere un sistema grafico, come Windows o X, oppure un ambiente

testuale come DOS o la console di Linux; concettualmente non ci sono differenze fra ambiente grafico e testuale in termini di possibilità di interazione con il sistema: entrambi gli ambienti permettono un'efficace utilizzo della macchina, anche se la difficoltà di utilizzo dell'ambiente testuale e la sua versatilità sono spesso molto maggiori.

7 La shell Bash

Bash è la shell standard di molti sistemi Unix fra cui anche Linux; è l'evoluzione della shell scritta da Stephen Bourne per Unix, chiamata appunto shell Bourne. Bash significa dunque Bourne Again SHell (in inglese "ancora shell Bourne"), ed associa le funzioni della shell Bourne alle funzioni di altre due shell piuttosto famose, la shell Korn (ksh) e la shell C (csh).

L'eseguibile della shell Bash è `/bin/bash` ed è paragonabile a `COMMAND.COM` degli ambienti Microsoft.

Può essere avviato in due modalità:

- la modalità interattiva, che permette all'utente di inserire comandi. Viene quindi visualizzato un invito a digitare detto "prompt dei comandi".
- la modalità non interattiva, in cui vengono eseguiti gli script di shell (come i file batch di DOS) e i comandi esterni; gli script verranno trattati più avanti. In questa modalità non viene visualizzato alcun prompt.

La shell in modalità interattiva, dunque, è ciò che si presenta quando abbiamo appena finito la procedura di login; sullo schermo appare un messaggio del tipo

```
[pippo@localhost etc]$ _
```

in cui il prompt è formato da tutti i caratteri fino a $ compreso, mentre il trattino è il cursore lampeggiante che indica dove si sta per scrivere; la linea visualizzata è quindi una "riga di comando". Il carattere $ si presenta quando l'utente al lavoro è un utente normale; se invece si sta lavorando come root, il carattere visualizzato è #.

Non appena avviata, Bash legge i file di configurazione `/etc/profile` e `/.bash_profile` (se quest'ultimo non esiste legge `/.bash_login`; nel caso nemmeno questo esista, prova con `/.profile`; altrimenti rinuncia alla lettura). I file menzionati contengono le impostazioni dell'ambiente di lavoro; ne parleremo più diffusamente tra qualche pagina, una volta introdotte le variabili.

7.1 Inserimento ed esecuzione dei comandi

Un comando altro non è che una serie di parole da scrivere al prompt; la pressione del tasto Invio fa sìche la shell legga quanto è stato inserito e lo intepreti; una volta intepretato, il comando viene eseguito.

I comandi che vengono eseguiti sono raccolti in una lista ordinata cronologicamente (detta "history dei comandi") che permette di ritrovarli facilmente e di non doverli cosìriscrivere.

Alcuni comandi sono forniti dalla shell stessa, e allora sono detti "comandi interni". Ma la maggior parte dei comandi disponibili non è interna, bensìè un

programma esterno alla shell che spesso risiede nella directory `/bin` o `/sbin`. Quando si eseguono i comandi esterni, Bash carica una nuova shell Bash in modo non interattivo ed è questa shell figlia ad eseguire il comando esterno.

La shell prevede dei tasti speciali per eseguire particolari e comode funzioni:

- Tab: premendo Tab, la shell cerca di completare quanto è stato scritto secondo lo schema seguente:
  - se la parola inizia con ~ , la interpreta come un nome di un utente e cerca un utente che inizi con le lettere digitate; se lo trova completa la parola col nome trovato e la visualizza sulla riga di comando;
  - se la parola inizia con $, la interpreta come un nome di una variabile e si comporta come sopra;
  - altrimenti interpreta la parola come un nome di file, e completa seguendo le regole succitate.
- Freccia su: richiama sulla linea l'ultimo comando digitato; premendo ancora freccia su, compare il penultimo e cosìvia.
- Freccia giù: richiama il comando successivo nella lista a quello visualizzato.
- Freccia a destra / sinistra: permette di spostarsi a destra/sinistra tra i caratteri del comando attivo al fine di correggerlo più agevolmente.
- Ins: fa passare dalla modalità di inserimento a quella di sovrascrittura.
- Inizio / Fine: sposta il cursore all'inizio/fine della riga di comando

Il modo più semplice di eseguire un comando è di scrivere il suo nome e premere Invio, ad esempio:

```
[pippo@localhost etc]$ pwd [-> Invio]
/home/pippo
```

questo comando visualizza il nome completo della directory in cui si sta lavorando al momento (pwd significa print working directory).

7.2 Parametri

pwd non richiede particolari informazioni per poter svolgere il proprio lavoro, quindi è possibile eseguirlo nel modo visto. Ma ci sono comandi che invece necessitano di informazioni aggiuntive che l'utente può specificare facendo seguire al nome del comando una lista di altre stringhe (dette parametri o argomenti) separate l'una dall'altra da spazi. Ad esempio il comando rm (remove) cancella i file; eseguendo tale comando senza specificare argomenti:

```
[pippo@localhost etc]$ rm [-> Invio]
rm: too few arguments
```

si riceve un messaggio di errore che esprime la necessità di ulteriori informazioni. Il modo corretto di eseguirlo è

```
[pippo@localhost etc]$ rm pippo.txt [-> Invio]
```

in questo modo si cancella il file pippo.txt dalla directory corrente. Una cosa da sapere è che un file, una volta cancellato, non è più recuperabile; il vecchio DOS offriva l'utility UNDELETE che permetteva di recuperare in qualche caso i file rimossi, ma su Linux la cancellazione non prevede questa possibilità. Una ragione in più per lavorare come utente normale e non come root, è appunto la certezza di non cancellare file importanti senza i quali il sistema sarebbe compromesso.

In generale, quindi, la struttura di un comando è la seguente:

```
nomecomando parametro1 parametro2 ... parametroN
```

Abbiamo visto che un parametro può essere un nome di un file, ma i parametri possono essere utilizzati per specificare quale opzione di un comando si vuole utilizzare. Ad esempio, rm prevede un'opzione per attivare la richiesta di conferma di rimozione di un file al fine di prevenire cancellazioni indesiderate. Il comando con l'opzione appare così:

```
[pippo@localhost etc]$ rm -i pippo.txt [-> Invio]
rm: remove 'pippo.txt' ? _
```

l'opzione -i (abbreviazione di interactive) fa sìche compaia un messaggio di conferma che aspetta una y o una n come risposta per procedere o annullare l'operazione.

Dall'esempio visto, si notano due fatti:

- le opzioni precedono i nomi di file
- le opzioni si distinguono dai file per la presenza di un `-` iniziale

nessuno dei due fatti menzionati è però una regola fissa, si tratta soltanto di una convenzione.

`rm` prevede anche un'opzione `-r` che permette di cancellare i file specificati nella directory corrente e anche nelle sottodirectory. Volendo eseguire `rm` con entrambe le opzioni `-i` e `-r` è possibile raggrupparle come segue:

```
[pippo@localhost etc]$ rm -ir pippo.txt [-> Invio]
rm: remove 'pippo.txt' ? _
```

Poniamo ora il caso di essere nella directory `/home/pippo`, e di voler spostare il file `trippa.gnam` che si trova nella directory `/home/pippo/tantafame`, per metterlo nella directory corrente. Un modo per farlo è il seguente:

```
[pippo@localhost pippo]$ mv /home/pippo/tantafame/trippa.gnam \
/home/pippo [-> Invio]
```

il comando `mv` (move) sposta il file specificato come primo argomento nella directory specificata come secondo argomento. La sintassi usata è un po' pesante, perchè abbiamo specificato l'intero percorso del file `trippa.gnam` e della directory corrente. Ma se ci troviamo in `/home/pippo`, è possibile sottintendere questa parte del percorso, scrivendo:

```
[pippo@localhost pippo]$ mv tantafame/trippa.gnam /home/pippo \
[-> Invio]
```

è da notare che il primo argomento non comincia per /, quindi viene interpretato da Bash come una sottodirectory di `/home/pippo`. Sarebbe comodo poter sottintendere anche `/home/pippo` nel secondo argomento, però una scrittura del tipo

```
[pippo@localhost pippo]$ mv /home/pippo/tantafame/trippa.gnam \
[-> Invio]
```

genera un errore in quanto mv vuole che la destinazione dello spostamento sia specificata. Allora si può utilizzare il carattere . che è il nome di un file che rappresenta la directory corrente. Ovvero, si può scrivere

```
[pippo@localhost pippo]$ mv /home/pippo/tantafame/trippa.gnam . \
[-> Invio]
```

e stavolta tutto funziona, perché viene intepretato da mv nel modo esatto. Se . è la directory corrente, .. è invece un file che punta alla directory madre di quella corrente. Nel nostro esempio, il comando

```
[pippo@localhost pippo]$ mv /home/pippo/tantafame/trippa.gnam \
.. [-> Invio]
```

sposta il file trippa.gnam dalla directory `/home/pippo/tantafame` alla directory `/home`. mv è usato anche per cambiare nome ai file (come il comando REN di DOS):

```
[pippo@localhost tantafame]$ mv trippa.gnam cosciotto.slurp \
[-> Invio]
```

questo comando, eseguito nella directory `/home/pippo/tantafame`, cambia nome al file `trippa.gnam` facendolo diventare `cosciotto.slurp`.

7.3 Caratteri speciali

Nel caso in cui si volessero cancellare tutti i file di una directory, con le conoscenze attuali dovremmo eseguire rm specificando come parametri i nomi di tutti i file; la scomodità della procedura è evidente, e per questo la shell Bash prevede dei caratteri speciali che rendono molto più agevole la selezione di liste di file. I caratteri speciali sono i seguenti: * (indica un insieme qualunque di caratteri), ? (è un segnaposto per un carattere qualunque), [...] (definisce un insieme di caratteri) e ~ (è il nome della directory home).

Alcuni esempi chiariranno l'uso dei caratteri speciali. Supponiamo che la directory corrente contenga i file `pippo.txt`, `pluto.txt`, `paperino.jpg`, `lippo.dat` e `gippo.gif`. Per cancellare tutti i file che cominciano per p si può scrivere

```
[pippo@localhost etc]$ rm p* [-> Invio]
```

La shell Bash interpreta p* come p seguito da una qualunque stringa, e i file `pippo.txt`, `pluto.txt` e `paperino.jpg` rispondono a questi requisiti. Quindi la shell, prima di mandare il comando in esecuzione, sostituisce p* con i nomi dei tre file indicati, trasformando il comando in

```
rm pippo.txt pluto.txt paperino.jpg
```

Il comando è ora pronto per essere eseguito. La sostituzione dei caratteri speciali non viene notificata dalla shell, cioé è fatta senza visualizzare alcun messaggio, quindi è bene controllare con attenzione che la stringa inserita generi la giusta lista di file.

Per cancellare tutti i file di una directory, procedere come segue:

```
[pippo@localhost etc]$ rm * [-> Invio]
```

qui * sta ad indicare una stringa qualsiasi, quindi qualunque nome di file va bene e allora vengono cancellati tutti. Se si vuol cancellare i file `gippo.gif`, `pippo.txt` e `lippo.dat`, si può eseguire il comando

```
[pippo@localhost etc]$ rm ?ippo* [-> Invio]
```

in cui il carattere ? viene interpretato dalla shell come un qualsiasi carattere. Se invece si vuol cancellare solo `gippo.gif` e `lippo.dat`, ma non `pippo.txt`, si può utilizzare la forma

```
[pippo@localhost etc]$ rm [g,l]ippo* [-> Invio]
```

in cui [g,l] indica che la shell deve scegliere solo fra g e l la sostituzione di carattere. Sono possibili selezioni multiple come [a-f,z] che indica l'insieme dei caratteri a, b, c, d, e, f, z.

Infine veniamo al carattere ~: se è da solo viene sostituito dalla shell con il nome completo della directory home, oppure se è seguito da caratteri viene sostituito con la home directory dell'utente il cui nome è quello che segue ~. Ad esempio, se in un sistema ci sono due utenti chiamati gianni e pinotto, e l'utente al lavoro è gianni, allora ~ viene sostituito con `/home/gianni` e ~pinotto viene sostituito con `/home/pinotto`.

7.4 Protezione dei caratteri speciali

Può capitare di dover lavorare su un file il cui nome contiene dei caratteri speciali, ad esempio `gi*gi`, ma come sappiamo il carattere * viene trasformato da Bash. In realtà, la trasformazione va a buon fine solo se esistono file che cominciano e finiscono per "gi"; se nessun file risponde a questi requisiti, Bash rinuncia alla trasformazione e lascia la stringa gi*gi invariata. Perciò, se l'unico file che risponde ai requisiti di cominciare e finire per "gi" è `gi*gi`, non ci sono equivoci: il file a cui ci si riferisce è quello desiderato. Ma nel caso in cui in una directory si trovino i file `gi*gi` e `giangi`, il comando

```
[pippo@localhost etc]$ rm gi*gi [-> Invio]
```

cancellerà solo `giangi`, perché la sostituzione di * è andata a buon fine. Una volta cancellato questo file l'equivoco scompare, ma se ogni volta che si presenta una situazione ambigua l'unica soluzione fosse di fare piazza pulita dei file che "coprono" `gi*gi`, la comodità della shell verrebbe meno.

Per risolvere le ambiguità che coinvolgono i caratteri speciali è possibile "proteggerli" dalla trasformazione tramite un altro carattere speciale (sembra quasi un circolo vizioso, ma poi le cose funzionano), ovvero il carattere *backslash*. Per proteggere * basta quindi farlo precedere da *backslash* in questa maniera:

```
[pippo@localhost etc]$ rm gi\*gi [-> Invio]
```

In questo caso, la shell Bash non trasforma * ma lo lascia invariato, ed elimina la *backslash*; il comando diventa dunque

```
rm gi*gi
```

proprio come volevamo. E per cancellare un file di nome `gi\gi`? Basta proteggere \ facendolo precedere da un altro \, in questo modo:

```
[pippo@localhost etc]$ rm gi\\gi [-> Invio]
```

e l'intepretazione che Bash ne dà è

```
rm gi\gi
```

A ben guardare, anche il carattere " " (spazio) è speciale per la shell Bash: è infatti il carattere che separa fra loro gli argomenti di un comando. Può capitare però di dover lavorare con file il cui nome contiene uno spazio, ma naturalmente non si può digitare

```
[pippo@localhost etc]$ rm ciao ciao come stai io bene [-> Invio]
```

sperando che serva a cancellare il file `ciao ciao come stai io bene`. In questo caso la protezione che si può fare è la seguente:

```
[pippo@localhost etc]$ rm ciao\ ciao\ come\ stai\ io\ bene \
[-> Invio]
```

oppure, più intuitivamente e comodamente

```
[pippo@localhost etc]$ rm "ciao ciao come stai io bene" \
[-> Invio]
```

Nel caso in cui, infine, il file da cancellare fosse '`mi chiamo "giangi"`', il comando opportuno è

```
[pippo@localhost etc]$ rm "mi chiamo \"giangi\"" [-> Invio]
```

Se un comando è troppo lungo per stare su un'unica linea, è possibile andare alla linea successiva digitando *backslash* seguito da Invio. Questa procedura non dovrebbe stupire: il carattere *backslash* toglie ai caratteri speciali il loro significato speciale: quindi anche il tasto Invio, se preceduto da *backslash* viene privato della funzione di marcatore di fine comando. Un esempio:

```
[pippo@localhost etc]$ rm nelmezzodelcammin\ [-> Invio]
dinostravita [-> Invio]
```

questo cancella il file `nelmezzodelcammindinostravita`.

7.5 Variabili di shell e di ambiente

Alcune informazioni che caratterizzano la sessione di lavoro (come la directory corrente, l'utente al lavoro, ...) sono immagazzinate in memoria sotto forma di variabili. Una variabile è una specie di "scatola" nella quale è possibile conservare un valore, sia numerico che testuale. Per distinguere fra loro le "scatole", ovvero le variabili, si è pensato di dotarle di un nome. Le variabili di shell hanno validità solo nella shell con cui si sta lavorando. Ma sappiamo che quando

si esegue uno script o un comando esterno viene aperta una nuova shell in modo non interattivo; perciò questa nuova shell non vedrà le variabili della shell madre. Però spesso si vuole che le variabili della shell su cui si lavora siano disponibili anche alle altre shell figlie aperte successivamente; allora è necessario trasformarle in variabili di ambiente.

Le variabili di shell più importanti sono:

- `PATH`: questa variabile contiene una lista di directory separate da : e serve alla shell per sapere dove trovare i comandi digitati. Quando si digita un comando, se il comando è interno l'esecuzione non richiede la lettura di `PATH`, ma se è esterno, la shell vuole sapere dove andare a cercarlo; allora legge la variabile PATH e prova ordinatamente a trovare il comando nella prima directory della lista, poi se non lo trova prova nella seconda e così via. Se il comando non viene trovato in nessuna delle directory, allora la shell emette un messaggio di errore (command not found). `PATH` di Linux è quindi come `PATH` di DOS.
- `HOME`: contiene l'indirizzo della directory home dell'utente al lavoro (`/home/gennaro` ad esempio)
- `PS1`: contiene il formato del prompt dei comandi, ovvero dell'invito a digitare che di solito appare nella forma `[pippo@localhost etc]$` Modificando questa variabile, cambia l'aspetto del prompt; è simile alla variabile `PROMPT` di MS-DOS.
- `HISTSIZE`: il numero massimo di comandi da conservare nella history (di solito 500 o 1000)
- `HISTFILE`: il file in cui scrivere la history (di solito `/.bash_history`)
- `PWD`: la directory corrente
- `OLDPWD`: l'ultima directory visitata prima di andare in quella attuale
- `BASH`: il percorso dell'eseguibile della shell Bash, ovvero `/bin/bash`
- `LOGNAME`: il nome con cui si è eseguito il login
- `USER`: il nome dell'utente al lavoro

Per dichiarare una variabile di shell di nome `PIPPO` e dargli un contenuto iniziale (ovvero in gergo inizializzarla), digitare

```
[pippo@localhost pippo]$ PIPPO="uncane" [-> Invio]
```

Per farla diventare un variabile di ambiente si utilizza il comando export:

```
[pippo@localhost pippo]$ export PIPPO [-> Invio]
```

Ma si può fare tutto in una volta, anziché in due passaggi:

```
[pippo@localhost pippo]$ export PIPPO="uncane" [-> Invio]
```

Per visualizzare il contenuto di una variabile, usare il comando echo in questo modo:

```
[pippo@localhost pippo]$ echo $PIPPO [-> Invio]
uncane
```

come si nota, per far capire alla shell che PIPPO non è il nome di un file ma il nome di una variabile si fa precedere PIPPO da un $.

Se sulla riga di comando scriviamo

```
[pippo@localhost pippo]$ echo $PI [-> Tab]
```

seguito dalla pressione del tasto Tab, la shell completa la riga visualizzando

```
[pippo@localhost pippo]$ echo $PIPPO
```

cioè proprio come fa coi nomi di file.

Per consultare la lista di tutte le variabili (che siano di shell o di ambiente) si può usare il comando set senza parametri, e per eliminare una variabile si deve digitare

```
[pippo@localhost pippo]$ unset PIPPO
```

Adesso possiamo riprendere brevemente il discorso sul contenuto di `/etc/profile` e `/.bash_profile`: questi file contengono una serie di dichiarazioni di variabili di ambiente e alcuni comandi per configurare l'ambiente di lavoro. `/etc/profile` ha una configurazione che si applica a tutti gli utenti; `/.bash_profile` invece contiene la configurazione personalizzata dell'utente.

7.6 Alias

Chi utilizza il DOS, per visualizzare la lista dei file di una directory utilizza il comando dir; su Linux, questo comando non si chiama dir ma ls, e per chi è abituato a digitare dir questo può essere uno spiacevole inconveniente. È allora possibile definire un nuovo nome (detto alias) per il comando ls col comando alias:

```
[pippo@localhost etc]$ alias dir="ls -l" [-> Invio]
```

l'opzione -l (long listing) di ls fa sì che la lista visualizzata sia provvista di molti dettagli sui file, più o meno come la lista presentata da dir. Quando impartiamo il comando dir alla shell, la prima cosa che viene fatta è la sostituzione di dir con ls -l, e poi vengono eseguite tutte le sostituzioni dei caratteri speciali. Ovvero:

```
[pippo@localhost etc]$ dir p* [-> Invio]
```

diventa

```
ls -l pippo.jpg pluto.gif
```

Per eliminare un alias, usare il comando unalias:

```
[pippo@localhost etc]$ unalias dir [-> Invio]
```

Se si vuole far sì che un alias sia impostato automaticamente dopo il login di Linux, bisogna semplicemente aggiungere il comando relativo nel file `/.bash_profile`; non è consigliabile aggiungerlo al file `/etc/profile` perché quest'ultimo contiene i dettagli di configurazione validi per tutti gli utenti.

7.7 Pipe e redirezione

Da quello che abbiamo visto finora, possiamo dire che ogni comando ha questa struttura concettuale di base:

- riceve dell'input (attraverso i parametri o attraverso domande effettuate in fase di esecuzione, come la richiesta di conferma di rm -i)
- emette dell'output (ad esempio presenta la lista dei file di una directory o mostra dei messaggi di errore)

Di solito, dunque, la tastiera è l'input e lo schermo è l'output; in gergo informatico, potremmo dire che la tastiera è lo standard input e lo schermo è lo standard output. Sappiamo che su Linux "tutto è un file", quindi non dovremmo stupirci di trovare dei file associati proprio allo standard input e allo standard output: e infatti, guardando nella directory `/dev` troviamo i tre file `/dev/stdin`, `/dev/stdout` e `/dev/stderr`.

I primi due sono proprio i file che si riferiscono allo standard input e allo standard output; il terzo invece è lo standard error. Per convenzione, i comandi hanno due tipi di output: quello normale che passa attraverso lo standard output, e quello relativo ai messaggi di errore che passa invece per lo standard error.

Quando viene eseguito, dunque, un comando legge caratteri dal file stdin, e scrive caratteri sul file stdout o stderr; ma nulla vieta di dirottare il flusso di caratteri verso il file stdout e di dirigerlo ad un file che vogliamo noi, perché sempre di file si tratta. È una procedura che almeno in teoria non dovrebbe generare errori.

L'operazione di dirottare il flusso di caratteri in entrata o in uscita da un comando si chiama appunto redirezione, e si effettua usando dei caratteri speciali. Ad esempio, se vogliamo salvare su file la lista delle variabili di shell e di ambiente, dobbiamo redirezionare l'output di set in questo modo:

```
[pippo@localhost pippo]$ set > lista.txt [-> Invio]
```

La lista delle variabili non viene più visualizzata, ma nella directory è stato creato un file lista.txt che contiene proprio quello che volevamo. Il carattere `>` indica a set che non deve scrivere su sdtout, ma su `lista.txt`; `>` costringe set a sovrascrivere il file `lista.txt`, ma è possibile anche far sìche i caratteri vadano ad accodarsi alla fine del file `lista.txt` usando `>>`. Se invece vogliamo redirezionare lo standard error, dobbiamo utilizzare `2> nomefile`, dove il 2 sta ad indicare il secondo canale di output che è proprio quello di stderr. Se vogliamo redirezionare entrambi i flussi di output ad un unico file, possiamo utlizzare `&>`. Per quanto riguarda lo standard input, il carattere speciale da utilizzare per la redirezione è `<`. Ad esempio, è possibile ordinare il contenuto di un file col comando sort come segue:

```
[pippo@localhost pippo]$ sort < elenco.txt [-> Invio]
abaco
babbione
zuzzurellone
```

In DOS, per generare la lista ordinata dei file di una directory esiste l'opzione /O del comando DIR. Il suo analogo Linux è il comando ls, come abbiamo già

visto, che però non possiede questa opzione. Si potrebbe ovviare al problema usando il comando sort in questo modo:

```
[pippo@localhost pippo]$ ls -l > elenco.txt [-> Invio]

[pippo@localhost pippo]$ sort < elenco.txt [-> Invio]
```

il primo comando crea un file di nome `elenco.txt` che contiene la lista dei file disordinata, il secondo comando visualizza sullo schermo la lista ordinata. È una soluzione un po' troppo scomoda per risolvere un problema così frequente, e infatti Bash mette a disposizione una scorciatoia per effettuare questa operazione: la pipe.

In inglese pipe significa "tubatura, condotto"; questo concetto è proprio quello che serve per risolvere il probema di prima. Infatti noi vogliamo che l'output disordinato di ls venga letto da sort per essere poi ordinato. Ovvero, vorremmo fare una "tubatura" che porta il flusso di caratteri (non d'acqua...) da ls in sort. La soluzione manuale già vista, cioé quella di creare un file intermedio (`elenco.txt`) è un modo per creare questo condotto, ma le pipe eseguono questa operazione automaticamente. In pratica, si scrive:

```
[pippo@localhost pippo]$ ls | sort [-> Invio]
```

dove il carattere `|` rappresenta proprio un tubo stilizzato. Questo comando effettua tutte le operazioni che prima facevamo a mano, ovvero prende l'output di ls e lo scrive su un file di tipo speciale (detto tipo pipe, appunto) gestito con una politica a coda, cioè il primo dato scritto sul file è il primo dato che viene letto poi da sort: come la coda alla posta! Una volta completata la creazione del file intermedio, sort legge tutti i dati e li ordina. Tutto questo accade senza che l'utente abbia alcun messaggio sullo schermo, perché le pipe sono silenziose. Finito l'ordinamento, finalmente la lista viene visualizzata sullo schermo.

Nel caso in cui la lista dei file sia troppo lunga per stare tutta in una schermata, sarebbe desiderabile poterla leggere una schermata alla volta. ls non prevede nemmeno questa possibilità, ma le pipe ci vengono ancora una volta in aiuto. Per dividere l'output di ls in pagine, di solito si utilizzano i comandi more o less: il primo aspetta la pressione di un tasto per visualizzare la pagina successiva, il secondo è più evoluto e permette anche di scorrere avanti e indietro di una riga o una pagina alla volta. Per ottenere una lista pagina dopo pagina, basta allora digitare:

```
[pippo@localhost pippo]$ ls | more [-> Invio]
```

oppure

```
[pippo@localhost pippo]$ ls | less [-> Invio]
```

8 Lista di alcuni comandi di frequente utilizzo

Help in linea (per visualizzare un testo esplicativo sui comandi)

```
man nomecomando
info nomecomando
nomecomando --help
```

Operazioni su directory

`cd nomedir`: (change directory) cambia la directory corrente
`mkdir nomedir`: (make directory) crea una directory
`rmdir nomedir`: (remove directory) cancella una directory solo se è vuota
`pwd`: (print working directory) visualizza il nome della directory corrente

Operazioni su file

`cp sorgente destinazione`: (copy) copia i file da `sorgente` a `destinazione`
`rm nomefile`: (remove) cancella i file
`mv sorgente destinazione`: (move) sposta i file da `sorgente` a `destinazione`
`chmod permessi nomefile`: (change mode) cambia i permessi di un file

Compressione e decompressione di file

`tar`: per lavorare con i file .tar
`gzip`: per lavorare con i file .gz
`zip`: per lavorare con i file .zip

Visualizzazione dei file (soprattutto nomi o contenuto)

`ls`: (list) visualizza la lista dei file di una directory
`find -name "nomefile"`: cerca un file nel filesystem
`less nomefile`: mostra il contenuto di un file una pagina alla volta
`more nomefile`: mostra il contenuto di un file una pagina alla volta
`sort`: ordina i dati che riceve

Informazioni sui dischi

`df`: (disk free) mostra la quantità di spazio libero nelle varie partizioni montate
`du nomedir`: (disk used) mostra la quantità di spazio occupato da una directory
`mount filesystem nomedir`: monta il filesystem nella directory specificata

Gestione degli utenti

`adduser nomeutente`: aggiunge un utente
`passwd nomeutente`: cambia la password di un utente
`su nomeutente`: (substitute user) cambia l'utente attivo

Comandi particolari

`echo messaggio`: visualizza un messaggio
`alias nuovonomecomando="comando"`: dà un altro nome al comando specificato
`unalias nomealias`: elimina un alias
`set nomevariabile="valore"`: imposta il valore per la variabile specificata
`unset nomevariabile`: elimina una variabile
`export variabile`: trasforma una variabile di shell in una di ambiente

9 Approfondimenti sulla shell

Per maggiori informazioni sui comandi, esiste in Bash un ottimo help in inglese che funziona come il comando HELP di DOS: è `man`. Per utilizzarlo digitare

```
[pippo@localhost pippo]$ man nomecomando [-> Invio]
```

È scritto in lingua inglese, ma è la fonte principale di informazioni per chi vuol conoscere a fondo i comandi della shell.

Un ulteriore lettura caldamente consigliata è l'HOWTO DOS2Linux, scritto sia in italiano che in inglese, reperibile al sito www.pluto.linux.it alla voce documentazione.

10 Bibliografia di uso generale

Sono molte le fonti da cui trarre informazioni su Linux, e in questa sede riporterò le più famose.

10.1 Help forniti con Linux

da console ci sono "man" e "info" dei quali si è visto l'utilizzo, in KDE ci sono la Guida di KDE e I Suggerimenti di Kandalf

10.2 Internet

- www.pluto.linux.it (alla voce Documentazione, si possono trovare gli HOWTO in italiano e Appunti di Informatica Libera)
- www.linux.com (per tutte le utenze, fornisce anche gli HOWTO)
- www.linuxplanet.com
- www.gnu.org (per avere informazioni su cos'è il software libero)
- www.linuxnewbie.com (per principianti)
- www.mandrakesoft.com (è il sito della distribuzione Mandrake, la più adatta a chi si avvicina al mondo Linux. Vedere al link Documentazione)

A GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if

there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in

the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.