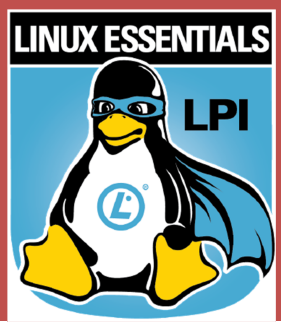


LPI Linux Essentials

Guida alla certificazione

Edizioni ithum srl

Copyright (c) 2013 – Ithum Srl

È garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation; con le Sezioni Non Modificabili (Tutte), con i Testi Copertina (Tutti), e con i Testi di Retro Copertina (Tutti). Una copia della licenza è acclusa nella sezione intitolata "Licenza per Documentazione Libera GNU".

ISBN: 9788866850090

Gli autori

Questo libro è il risultato di un lavoro di gruppo da parte di persone che hanno partecipato alla Beta mondiale dell'esame di certificazione Linux Essentials a Roma e Orvieto in 29 e il 30 maggio 2012. Per primi hanno fatto pratica diretta con la nuova certificazione di LPI e, tramite l'associazione ICT Academy, hanno hanno prestato il loro tempo e la loro buona volontà alla creazione di questi materiali.

Di seguito la lista (in ordine alfabetico) di tutti coloro che hanno partecipato a questo progetto con il loro attivo contributo scrivendo capitoli o con un paziente lavoro di revisione e integrazione dei contenuti:

- Simone Ciccarone
- Salvatore Cristofaro
- Edoardo D'Atri
- Marco De Luca
- Antonio Doldo
- Alberto Luzzi
- Enrico Mainero
- Piero Mazzotta
- Emanuela Nicolosi
- Emanuele Petrucci
- Andrea Pigliacelli
- Massimiliano Pippi
- Alessandro Profiti
- Andrea Ranaldi
- Luca Serpietri
- Francesco Settembre
- Luca Silvani
- Alessio Tomellieri
- Sofien Vannutelli
- Stefano Za

Dato che il libro è stato scritto da mani diverse è possibile che, nonostante le revisioni, al suo interno siano presenti delle difformità o che alcune parti siano mancanti.

Questo può anche essere dovuto al fatto che gli autori non hanno usato tutti la stessa distribuzione di Linux e che non esiste ancora un indice dettagliato dei contenuti della certificazione. All'interno dell'Appendice 1 del presente libro è possibile verificare il

syllabus attuale degli argomenti della certificazione. La struttura di questo libro cerca di seguire, per quanto possibile, questo schema.

Invitiamo tutti i lettori che verifichino delle inesattezze o vogliano contribuire a quest'opera a segnalarlo direttamente a:

Ithum Srl

Piazza Vincenzo Mangano 28, 00177 Roma (RM)

Telefono: 06 21 58 915

Emai: lpi@ithum.it

Website: http://www.ithum.it

ISBN: 9788866850090

Sommario

ISBN: 9788866850090

ithum®

Parte 1:
La comunità Linux e le carriere lavorative nell'Open Source

ithum®

1 Evoluzione di Linux e diffusione dei sistemi operativi

1.1 Breve storia del computer

Prima di entrare nel dettaglio di cosa sia un computer, ecco qualche citazione nota che ben introdurrà il lettore nel mondo informatico:

"Originariamente si pensava che se ci fossero stati cinque o sei enormi computer [negli USA], nascosti nei laboratori di ricerca, essi sarebbero stati sufficienti a soddisfare le necessità dell'intera nazione"

Howard H. Aiken, 1952

Howard Aiken fu un pioniere dei moderni computer e il progettista del primo computer della IBM, il "Harvard Mark I".

I primi computer in senso moderno furono costruiti durante la seconda guerra mondiale per permettere di decifrare messaggi in codice o per eseguire calcoli complessi. Erano molto grandi fisicamente, difficili da usare e inclini all'errore - componenti elettronici come i transistor o i circuiti integrati non era ancora stati inventati. Il risultato della ricerca informatica in quegli anni e in quelli immediatamente successivi furono delle assunzioni di base che consentirono di definire il termine “computer”:

- Un computer elabora *dati* secondo una sequenza di istruzioni (*programma*) eseguite automaticamente
- I programmi devono consentire l'esecuzione condizionale e i cicli
- Deve essere possibile cambiare o rimpiazzare il programma eseguito dal computer.

Ad esempio, molti dispositivi tecnologici (dalle televisioni alle macchine fotografiche, dalle lavatrici alle automobili) contengono piccole unità di controllo programmabili, quasi come se fossero dei piccoli computer. Tuttavia, non consideriamo questi dispositivi come dei “computer” perché eseguono solo programmi fissi e non modificabili. Al contrario, una calcolatrice può essere usata per “elaborare dati” ma non è un “computer” perché non lo fa automaticamente: qualcuno deve pigiare i tasti.

Nei primi anni '50, i computer erano dispositivi altamente specializzati per lo più utilizzati all'interno di istituti di ricerca.

Le serie televisive di fantascienza del tempo mostravano enormi sale piene di file di armadi contenenti misteriosi ingranaggi rotanti. In meno di 70 anni, questa immagine è cambiata radicalmente.

"Non c'è nessuna ragione per nessuno di avere un computer nella propria casa"

Ken Olsen, 1977

Ken Olsen era il CEO di un'altra industria di computer, Digital Equipment Corporation (DEC), che spingeva il mercato, durante tutti gli anni ’70, verso lo sviluppo di "piccoli" computer.

Gli anni '70 videro l'avvento dei "computer casalinghi". Questi non possono essere comparati ai PC di oggi - bisognava saldare insieme i pezzi (cosa che sarebbe fisicamente impossibile oggi), e raramente avevano una tastiera ragionevole né tantomeno un display. Erano piuttosto un passatempo per appassionati, al pari di un set di trenini elettrici, anche perché in tutta onestà non erano realmente utili a nulla. Tuttavia, erano definiti "computer" perché liberamente programmabili, anche se i programmi dovevano essere laboriosamente inseriti a mano o caricati da audiocassette.

Solo durante la fine degli anni '70 e '80, i "computer casalinghi" si trasformarono in kit di dispositivi pronti all'uso (nomi come "Apple II" o il "Commodore 64" potrebbero essere familiari ai lettori non giovanissimi) e iniziarono a comparire anche negli uffici.

Il primo PC IBM fu introdotto nel 1981, e la Apple produsse il primo "Macintosh" nel 1984. Il resto, come si dice, è storia, ma non si dovrebbe tuttavia dimenticare che nel mondo dei computer non esistono solo Mac e PC. I computer grandi quanto una sala, chiamati *mainframe*, sono ancora in circolazione - anche se tendono a diventare sempre più rari. Tuttavia, i principi non sono cambiati dai tempi di Howard Aiken: i computer sono sempre dispositivi che elaborano automaticamente dati secondo programmi intercambiabili che possono contenere espressioni condizionali e cicli.

E questi principi di base non cambieranno mai.

1.2 Hardware di un computer

Cogliamo l'occasione per dare un'occhiata ai componenti interni di un personal computer (PC, o, per meglio dire, “PC IBM-compatibile”).

Il processore (o “CPU”, Central Processing Unit) è il nucleo del computer. E' il componente che effettua l'elaborazione dei dati attraverso calcoli in sistema binario. I processori di oggi contengono più “nuclei”, il che vuol dire che i componenti principali del processore sono moltiplicati e possono operare indipendentemente – questo aumenta la velocità di elaborazione del computer e quindi le sue prestazioni.

I PC normalmente contengono processori prodotti da Intel o AMD. I tablet e gli smartphone generalmente usano processori ARM, che non sono ugualmente potenti ma molto più efficienti in termini di consumo di energia.

La memoria di un computer è chiamata RAM (Random Access Memory, “memoria ad accesso casuale”). Essa contiene non solo i dati in corso di elaborazione, ma anche il codice del programma in esecuzione. I computer di oggi sono normalmente equipaggiati con 1 gigabyte di RAM o più.

La scheda grafica consente di mostrare a video l'output dell'elaborazione del computer. Oggi anche gli smartphone più semplici hanno un'ottima grafica, e i PC comuni contengono hardware grafico che negli anni '90 sarebbe costato quanto l'equivalente di un'auto sportiva o di un piccolo appartamento. Le schede grafiche moderne contengono un processore dedicato, che spesso è più veloce della CPU del computer, ma ha ovviamente degli scopi molto più ristretti.

La scheda madre è tipicamente un grosso pezzo di plastica rettangolare sul quale sono saldati la CPU, la RAM e la scheda grafica di un computer. Essa comprende anche i connettori per i dischi fissi, per la stampante, la tastiera, il mouse, i cavi di rete e i dispositivi elettronici necessari per controllare questi connettori. Le schede madri per computer sono disponibili in tutte le forme e colori – dai piccoli computer silenziosi usati come videoregistratori in salotto fino ai mainframe che necessitano di molto spazio per la RAM e i processori.

L'alimentatore di un computer serve a fornire a esso l'energia elettrica necessaria per lavorare.

La maggior parte dell'energia elettrica fornita dall'alimentatore al computer viene successivamente trasformata in calore, il che rende molto importante il sistema di raffreddamento. Per questo i computer contengono almeno una ventola che, girando, soffia aria fresca sui componenti elettronici del computer prevenendone il surriscaldamento.

I dischi fissi, al contrario della RAM, sono usati per memorizzare dati non in uso in un certo momento. Il motivo di ciò sta nel fatto che i dischi fissi possono memorizzare molti più dati rispetto alla RAM: infatti, la loro capacità si misura in terabytes piuttosto che in gigabytes.

Tradizionalmente, gli hard disk sono costituiti da piatti rotanti rivestiti di materiale magnetico. Le testine possono magnetizzare questo materiale in diversi punti e successivamente rileggere i dati scritti con questo metodo. Questo rende i dischi fissi molto sensibili a urti e cadute: infatti, se una testina viene a contatto con il piatto mentre il disco è in rotazione, lo distrugge.

L'ultima moda nel campo dei dispositivi di salvataggio dati è costituita dai dischi a stato solido (Solid State Disk, SSD), che invece di usare piatti rotanti utilizzano memorie flash. Le memorie flash sono come la RAM, con la differenza che possono conservare il loro contenuto anche senza elettricità. I SSD sono più veloci dei dischi magnetici ma anche molto più costose in termini di capacità. Si rivelano pertanto molto interessanti per i computer portatili perché non contengono parti mobili, non consumano molta energia e non subiscono malfunzionamenti in seguito a urti e cadute.

ISBN: 9788866850090

Ci sono vari modi per connettere un disco fisso (magnetico o SSD) a un computer. Il più comune è il “serial ATA” (SATA). I computer più datati usano “parallel ATA”, anche chiamato “IDE”. I server usano dischi SCSI o SAS (“serially attached SCSI”). Per quanto riguarda i dischi esterni, si usano connettori universal serial bus (USB) oppure eSATA.

Oltre ai dischi fissi, i PC tipicamente supportano dispositivi di lettura e scrittura su dischi ottici come CD, DVD o Blu-ray. I dischi ottici sono tipicamente usati per copiarci sopra e distribuire software e contenuti digitali. La loro importanza, tuttavia, è messa in ombra dal crescente utilizzo di Internet come mezzo di distribuzione.

Il display è il dispositivo di output usato per visualizzare l'output dell'elaborazione del computer. Al giorno d'oggi sono spariti i vecchi monitor a tubo catodico (CRT) e si usano schermi basati sui cristalli liquidi (LCD, liquid crystal display). Gli schermi LCD sono più sottili e hanno minore emissione di radiazioni rispetto ai vecchi monitr. Sono disponibili in tutte le misure, dagli smartphone ai maxischermi da muro. La qualità di uno schermo è determinata dalla sua risoluzione che si misura in *pixel*. Molti computer supportano più di uno schermo.

Esistono poi una serie di periferiche aggiuntive rispetto all'elenco fornito sino a ora: stampanti, scanner, fotocamere, modem e così via.

1.3 Software di un computer

Il software, in un computer, è importante come il suo hardware. Esistono tre categorie di software:

- Il firmware è memorizzato sulla scheda madre del computer e non può essere rimpiazzato o modificato facilmente. Viene usato per far sì che un computer venga a trovarsi in un determinato stato dopo l'accensione. Sui PC questo software è chiamato BIOS (Basic Input / Output System).
- Il sistema operativo consente al computer di essere utilizzato, gestendone le risorse hardware (RAM, disco fisso, processore) e l'accesso alle altre periferiche. Il software centrale del sistema operativo, che ha il compito di fornire ai processi in esecuzione sul computer un accesso sicuro e controllato all'hardware, è chiamato *kernel*. Il sistema operativo spesso è corredato di un'interfaccia utente grafica (Graphical User Interface – GUI) che determina il *look and feel* del computer rispetto agli utenti.

Tipicamente, quando si parla di computer, la maggior parte della gente tende ad associarli a Microsoft Windows. Questo deriva dal fatto che al giorno d'oggi la maggior parte dei PC venduti contiene Windows come sistema operativo preinstallato – il che non è un male, perché consente agli utenti di poter usare il computer appena acquistato senza doversi preoccupare di installare il sistema operativo. Il lato negativo, tuttavia, è che questo approccio rende la vita difficile a sistemi operativi alternativi come Linux.

Il sistema operativo Microsoft Windows di oggi è un discendente di “Windows NT”, che costituì il tentativo di Microsoft di adeguare il suo sistema operativo agli standard degli anni '90 (versioni precedenti come “Windows 95” costituivano estensioni grafiche al sistema operativo del tempo, MS-DOS, ed erano abbastanza primitive già ai loro tempi). Dare un giudizio critico sul sistema operativo Windows non è appropriato in questa sede, quindi ci limiteremo a dire che fa all'incirca ciò che ci si aspetterebbe da un sistema operativo, fornisce un'interfaccia grafica e gestisce le periferiche esterne.

Il computer “Macintosh” fu lanciato da Apple nel 1984 e già da allora, come oggi, era equipaggiato con un sistema operativo chiamato “Mac OS”. Nel corso degli anni, Apple fece varie modifiche alla piattaforma e al sistema operativo, alcune delle quali decisamente radicali. La versione attuale, Mac OS X (inteso come numero 10 in lettere romane, e non come lettera “X”), si basa su un'infrastruttura non dissimile dal sistema operativo Linux.

La grande differenza tra Windows e OS X sta nel fatto che OS X è venduto esclusivamente con i computer Apple e non è possibile eseguirlo su un “normale” PC. Questo consente ad Apple di fornire con maggiore facilità un sistema più omogeneo. Windows, invece, deve essere poter eseguito su qualunque PC e supportare le più diverse tecnologie hardware che possono essere combinate in migliaia di modi diversi. Di conseguenza, gli utenti Windows spesso devono avere a che fare con problemi di incompatibilità che possono essere molto difficili da superare. D'altro canto, però, c'è molta maggiore possibilità di scelta dell'hardware per i computer basati su Windows, a prezzi meno esorbitanti rispetto a quelli dei computer Apple.

I programmi utente consentono di fare qualcosa di utile, come scrivere documenti, manipolare immagini, comporre musica, giocare ai videogiochi, navigare su Internet o sviluppare altro software. Questi programmi sono anche chiamati *applicazioni*. In aggiunta a essi esistono spesso delle *utility* fornite dal sistema operativo all'amministratore del sistema per modificare la configurazione del computer. Inoltre, i computer che fanno da server supportano spesso software che fornisce servizi (posta elettronica o navigazione nel Web) ad altri computer detti client.

1.4 Linux

Nell'estate del 1991, uno studente dell'Università di Helsinki di nome Linus Torvalds realizzò sul suo PC x386 un programma in grado di essere eseguito direttamente sull'hardware senza sistema operativo, che gli consentiva di accedere al sistema Unix dell'Università.

Unix è un sistema operativo, scritto principalmente in C con supporto al multitasking con prelazione (preemptive multitasking)1 creato nel 1969 presso i laboratori AT&T Bell. Unix

1 Nel preemptive multitasking, il kernel può assegnare porzioni di tempo della CPU tra i processi anche se i processi non rilasciano volontariamente il controllo di essa.

fu reso disponibile (in seguito a un decreto che impediva ad AT&T di vendere software) alle Università unitamente al suo codice sorgente per permetterne la compilazione su macchine differenti. Iniziò così ad essere usato e modificato, e ai tempi di Linus Torvalds era il sistema operativo più diffuso nell'ambito delle Università e dei centri di ricerca e sviluppo.

Il kernel realizzato da Torvalds era monolitico e derivato dal kernel Unix in modo da poter riutilizzare il software sviluppato per Unix su Linux.

Il 25 agosto del 1991 Linus annunciò il suo progetto al pubblico e ben presto molti volontari (non solo studenti e appassionati, ma anche aziende come IBM, Red Hat oppure Oracle) decisero di collaborare. A quei tempi il sistema non aveva ancora un nome: “Linux” era considerato troppo personalistico da Torvalds, il quale optò per il termine “Freax”. Quando però il sistema venne caricato sui server FTP dell'università, un suo collega lo rinominò “Linux”, che divenne finalmente il nome ufficiale del nuovo progetto.

Linux 0.99 venne rilasciato a Dicembre 1992, mentre la versione 2.0 venne rilasciata all'inizio del 1996 e introdusse funzionalità importanti quali il supporto per i multiprocessori e la capacità di caricare moduli kernel a runtime. Venne inoltre introdotto “Tux”, il pinguino-mascotte di Linux.

Linux oggi è molto cambiato rispetto agli anni '90: le migliorie continue sono state, e sono tuttora, introdotte grazie al contributo degli sviluppatori.

Nel luglio 2011, a vent'anni dalla nascita di Linux, Linus Torvalds ha deciso di rinominare la versione in preparazione (.2.6.40) in “Linux 3.0” che contiene oltre 14 milioni di righe di codice. Con l'avvento della versione 3.0, tuttavia, il progetto Linux non si è concluso; anzi, esso viene costantemente esteso e migliorato da centinaia di programmatori al mondo, molti dei quali rivestono ruoli importanti nell'industria informatica.

A differenza di Windows e OS X, Linux non è sostenuto da alcuna azienda che ha interessi economici nel far sì che abbia successo. Linux è “disponibile liberamente” e può essere usato da chiunque – anche con scopi commerciali – purché accetti le regole del gioco (dettagliate nel capitolo 3).

Questo aspetto, unito al fatto che attualmente Linux può essere eseguito non solo su PC ma anche su altre piattaforme (dai telefoni, agli smartphone, ai mainframe), rende Linux il sistema operativo più versatile nella storia dell'informatica moderna.

La versatilità di Linux lo rende inoltre il sistema operativo di riferimento per la *virtualizzazione* e il *cloud computing*.

La virtualizzazione consente di emulare su di un computer “fisico” molti computer “virtuali”, ciascuno con un proprio sistema operativo virtualizzato sul quale è possibile eseguire normalmente le applicazioni senza differenze rispetto a un sistema operativo nativo. Questo porta a grandi benefici: uso più efficiente delle risorse hardware, maggiore flessibilità, possibilità di migrare facilmente macchine virtuali da una macchina fisica all'altra.

Questo consente di costruire un'infrastruttura in grado di reagire facilmente e senza inconvenienti a situazioni di carico eccessivo o malfunzionamenti.

Strettamente legato alla virtualizzazione è il concetto di cloud computing. Il cloud computing identifica la fornitura, tramite Internet, del software come servizio (software as a service) e on demand, anziché come applicazione installata sul computer dell'utente.
I fornitori di servizi cloud computing consentono ai loro clienti di usare macchine virtuali attraverso Internet, chiedendo loro un compenso commisurato al reale utilizzo di esse. Questo approccio porta notevoli risparmi rispetto ai costi di manutenzione di un'infrastruttura per datacenter, in termini di costi per la messa in opera, di sistemi per la fornitura di energia, di forza lavoro addetta alla gestione di esso.

1.5 Hardware consigliato e accenno alle distribuzioni

In realtà, quando si dice “Linux” si intende il solo kernel del sistema operativo, cioè il programma che gestisce l'allocazione delle risorse verso applicazioni e utility. Dal momento che un sistema operativo risulterebbe inutile senza applicazioni, tipicamente un utente installa una distribuzione Linux (volgarmente detta “distro”), vale a dire un pacchetto che consiste nel kernel “Linux” e una selezione di applicazioni, utilità, documentazione e altro materiale utile. La cosa interessante è che la maggior parte delle distribuzioni Linux è “disponibile liberamente” senza alcun costo o a costo molto basso. Questo rende possibile equipaggiare il proprio computer con software il cui equivalente per Windows oppure OS X costerebbe centinaia di euro, e poterlo fare su più computer, senza violare alcuna restrizione di licenza.

Esiste una moltitudine di distribuzioni Linux che hanno diversi obiettivi e approcci e diversa struttura organizzativa. Alcune distribuzioni sono pubblicate da aziende e vendute in cambio di denaro, altre sono realizzate da gruppi di sviluppatori volontari.

E' possibile installare una distribuzione Linux in diversi modi:

- Come unico sistema operativo del computer;
- Insieme ad un altro sistema operativo, e decidere di volta in volta quale dei due far partire all'avvio del computer (dual boot);
- Come macchina virtuale usando un tool per virtualizzazione
    - VMWare (http://www.vmware.com)
    - Virtualbox (http://www.virtualbox.org)

Per installare Linux il computer deve soddisfare almeno i seguenti requisiti:

- **CPU** 400 MHz Pentium Pro o superiore
- **RAM minima** 512 MB – **consigliata** 1 GB o superiore
- **Spazio su Hard Disk** almeno 10 GB disponibili su disco non partizionato

Alcune distribuzioni possono lavorare su computer ancora meno potenti, altre possono richiedere un hardware a più alte prestazioni. Nel paragrafo successivo prendiamo in esame le distribuzioni Linux general-purpose più comuni.

1.6 Distribuzioni Linux più comuni

1.6.1 Red Hat

Red Hat[2] nacque nel 1993 come “ACC Corporation”, un'azienda di distribuzione di accessori per Linux e Unix. Nel 1995 il fondatore, Bob Young, comprò la compagnia di Marc Ewing, che nel 1994 aveva pubblicato una distribuzione chiamata “Red Hat Linux”, e ne cambiò il nome in “Red Hat Software”.

Tuttora, Red Hat è probabilmente la più grande azienda basata esclusivamente su Linux e software open source.

Inizialmente la distribuzione “Red Hat Linux” era indirizzata ai singoli utenti, mentre dal 2004 porta a un uso professionale in ambito aziendale (principalmente nei datacenter) con il nome di Red Hat Enterprise Linux (RHEL). RHEL è licenziato in base al numero dei server su cui è installato, anche se non è previsto pagamento per il software (rilasciato sotto licenza GPL e FOSS), ma solo per l'accesso agli aggiornamenti e ai servizi di supporto.

Fedora è una distribuzione, controllata principalmente da RedHat, usata soprattutto come ambiente di test per RHEL. Aggiornamenti e innovazioni che riguardano il software vengono prima provate su Fedora e tutto ciò che si rivela utile viene successivamente portato su RHEL. Fedora non è a pagamento, si può scaricare gratis e il relativo progetto è gestito da un comitato i cui membri sono in parte eletti dalla comunità di sviluppatori e in parte nominati da RedHat. Per molti utenti Fedora è interessante proprio per lo spirito di innovazione che la contraddistingue, anche se questo la rende non adatta all'uso da parte di utenti alle prime armi né all'installazione su server che devono garantire la massima affidabilità.

1.6.2 SUSE

L'azienda tedesca S.u.S.E. (abbreviazione dal tedesco di “Sviluppo di software e sistemi” - Software und System-Entwicklung”) fu fondata come gruppo di consulenza Unix nel 1992 a Norimberga, in Germania. Il sistema operativo SUSE nacque originariamente come traduzione in tedesco della distribuzione Slackware, e la sua prima versione fu rilasciata del 1994 con il nome di S.u.S.E Linux 1.0. La prima versione di SuSE che si differenziava da Slackware fu la 4.2 del 1996. SuSe in breve tempo divenne la principale distribuzione in lingua tedesca. Era possibile scegliere la versione “Personale” e “Professionale”: quest'ultima versione era più costosa e conteneva software maggiormente orientato ai server.

Nel Novembre 1992 SuSE fu venduta all'azienda americana Novell per 210 milioni di dollari. Nell'Aprile 2011 sia Novell che SuSe vennero acquisite da Attachmate.

[2] http://www.redhat.com

Come Red Hat, SUSE offre un “enterprise Linux” chiamato SUSE Linux Enterprise Server (SLES[3]). Oltre alla versione enterprise, c'è la versione SUSE Linux Enterprise Desktop (SLED) che è orientata all'uso sulle workstation di tipo desktop.

Anche SUSE ha realizzato una versione open della propria distribuzione, chiamata openSUSE[4] che viene generalmente resa scaricabile alcuni mesi dopo l'uscita su supporti ottici. A differenza di Red Hat, SUSE offre ancora una tipologia di package che contiene software proprietario.

1.6.3 *Debian*

A differenza delle distribuzioni Red Hat e SUSE che hanno alle spalle grandi aziende di sviluppo, il progetto Debian si fonda sulla collaborazione di oltre mille volontari il cui obiettivo è rendere disponibile una distribuzione Linux di alta qualità chiamata “Debian GNU/Linux”. Debian è distribuito esclusivamente tramite download da appositi server di distribuzione ed è basato su tre documenti:

- Le “linee guida del software libero Debian” (Debian Free Software Guidelines, DFSG) definiscono quale software è considerato libero (DFSG-free) all'interno del progetto. Questo è importante dal momento che solo software DFSG-free può fare parte della distribuzione Debian. Il progetto distribuisce anche software non libero, che viene tenuto rigidamente separato dal software DFSG-free nei server di distribuzione: il software non libero si trova nella cartella non-free, il software DFSG-free si troba nella cartella main.
- Il Social Contract decrive gli obiettivi del progetto
- La Debian Constitution descrive l'organizzazione del progetto.

In virtù della sua organizzazione, della libertà da interessi commerciali e della separazione tra software libero e non libero, Debian è un'ottima base per distribuzioni da essa derivanti.

1.6.4 Ubuntu

Una delle più importanti distribuzioni derivanti da Debian è Ubuntu, che è fornita dall'azienda inglese Canonical Ltd., fondata dall'imprenditore sudafricano Mark Shuttleword. Ubuntu è una parola della lingua Zulu il cui significato è “umanità verso gli altri”. L'obiettivo di Ubuntu è offrire una distribuzione Linux attuale, potente e facile da capire che venga aggiornata a intervalli regolari.

Gli aggiornamenti di ciascuna versione di Ubuntu vengono infatti pubblicati ogni sei mesi, e ogni due anni viene rilasciata una versione supportata a lungo termine per la quale la Canonical promette cinque anni di aggiornamenti.

Alcuni sviluppatori di Ubuntu partecipano attivamente anche al progetto Debian, il che garantisce lo scambio di innovazione tra i due progetti. Una grossa differenza sta nel fatto che a Ubuntu non sta strettamente a cuore l'aspetto della libertà del software:

[3] http://www.suse.com/products/serve/

[4] (http://www.opensuse.org)

quindi, se da un lato tutti i tool infrastrutturali di Debian sono disponili come software libero, per Ubuntu non è sempre così.

1.6.5 *Altre distribuzioni*

Esistono molte altre distribuzioni famose, come Mandriva[5] e Turbolinux[6] come principali competitor di Red Hat e SUSE, oppure Gentoo Linux[7], focalizzata sul codice sorgente.

Con il termine '**Embedded Linux**' ci si riferisce poi a un insieme di distribuzioni concepite per essere utilizzate su sistemi embedded. I sistemi embedded sono dispositivi hardware secializzati che rispondno ad uno scopo specifico. Le caratteristiche di tali dispositivi impongono dei vincoli molto severi al sistema operativo in termini di occupazione di memoria flash, spazio su disco e tempo di avvio.

Android è un esempio di Embedded Linux. Di fatto è un sistema operativo open source per dispositivi mobili, basato sul kernel 2.6 di Linux, il cui ambiente utente si fonda su una macchina virtuale ("Dalvik") realizzata e manutenuta da Google. Grazie alla sua versatilità, Android può essere utilizzato anche su una moltitudine di altri dispositivi come tablet, lettori di ebook, fotocamere, notebook, smart TV, orologi, cuffie e così via.

Recentemente hanno cominciato a diffondersi anche le cosiddette distribuzioni live, che non richiedono installazione e che si possano eseguire direttamente da un supporto rimovibile come CD, DVD o pen drive.

Domande:

1. Quale dei seguenti è un sistema operativo Embedded Linux?

A. Android B. SUSE C.CentOS

D. Debian F. Fedora

2. Il kernel Linux deriva dal kernel BSD?

A. Vero B. Falso

3. Che tipologia di multitasking utilizzano Linux e Unix?

A. Preemptive B. Multiuser

C. Cooperative D. Single-tasking

E. Single-user

4. Linux utilizza un kernel di tipo:

A. Modulare B. Monolitico

C. Multitask D. Shell

[5] http://www.mandriva.com/en/Linux

[6] http://www.turbolinux.com

[7] http://www.gentoo.com

Risposte:

1. A
2. B
3. A
4. B

ithum®

ISBN: 9788866850090

2 Le principali applicazioni compatibili con Linux

2.1 Introduzione

Linux è un sistema operativo potente ed elegante, ma sarebbe inutile senza programmi.

In questa sezione presentiamo una selezione dei software più importanti che si possono trovare su un computer equipaggiato con Linux.

2.2 Applicazioni per la produttività

La maggior parte dei computer viene utilizzata per le applicazioni da ufficio, che riguardano per esempio la scrittura, la presentazione di progetti e l'analisi statistica, la grafica vettoriale e il video editing. Inoltre, gli utenti trascorrono gran parte del loro tempo al computer per navigare su Internet, leggere o scrivere e-mail; quindi non c'è da meravigliarsi se esistono molti software che permettono di svolgere questi compiti.

La suite OpenOffice.org è stata per anni l'ammiraglia della comunità open source nelle applicazioni per ufficio. Tutto ebbe inizio anni fa con "StarOffice", poi acquisito da Sun Microsystem e reso disponibile come software libero.

OpenOffice.org contiene tutto quello che ci si aspetta da un pacchetto di applicazioni per ufficio: un programma per la scrittura di testi, un foglio elettronico, un programma per le presentazioni e una base di dati. Infine, ha la caratteristica di essere compatibile con i formati di file di Microsoft Office, ma dispone anche di formati nativi basati su XML.

LibreOffice è una nuova versione di OpenOffice.org realizzata da un gruppo di sviluppatori di OpenOffice dopo l'acquisto di Sun da parte di Oracle. Entrambi i progetti LibreOffice e OpenOffice.org sono attualmente mantenuti fianco a fianco (Oracle ha donato OpenOffice.org alla Apache Software Foundation), e tuttora non è chiaro se ci sarà un'unificazione dei due software.

Mozilla Firefox è ormai il browser web più diffuso al mondo e viene distribuito da Mozilla Foundation. Firefox è più sicuro ed efficiente rispetto al precedente browser più usato (Microsoft Internet Explorer), è leggero, veloce, facile da usare e conforme agli standard che regolano il World Wide Web. Inoltre è disponibile una vasta gamma di estensioni con le quali è possibile personalizzare Firefox secondo le proprie esigenze.

Mozilla SunBird è un progetto supportato dai sistemi operativi Linux e Windows, per la gestione dei calendari ed impegni, basato sul linguaggio Interfaccia Utente di Mozilla, XUL.

Mozilla Thunderbird è un progetto della Mozilla Foundation per la gestione della posta elettronica, e condivide grandi parti della sua infrastruttura di base con il browser Firefox - e come Firefox - offre all'utenza un grande bacino di estensioni per vari scopi.

Chromium è la variante open source del browser Google Chrome. Chrome, che ha recentemente iniziato a competere con Firefox, è un potente browser sicuro con varie estensioni e perfettamente integrato con i servizi web forniti da Google.

2.3 Tool di gestione pacchetti e repository

L'installazione di software su Linux può avvenire in quattro modi diversi:

- online, attraverso i rispettivi gestori di pacchetti di ogni distribuzione. E' il sistema più semplice, poiché un pacchetto software può avere una o più dipendenze da altri pacchetti software che devono essere scaricati e installati come prerequisito. Attraverso l'installazione online le dipendenze vengono automaticamente risolte e il sistema viene mantenuto pulito a lungo andare.
- installazione manuale dei pacchetti: è un sistema molto comodo per installare applicazioni non contenute nei gestori di pacchetti delle varie distribuzioni. Lo svantaggio è che non risolve automaticamente le dipendenze;
- installazione da codice sorgenti: prevede la compilazione del codice sorgente. E' sicuramente il metodo più complesso, non risolve le dipendenze e col passare del tempo può rendere difficile la manutenzione del sistema;
- attraverso un file binario eseguibile (.bin o .run), analogamente a come avviene su Windows: è il caso tipico dei videogiochi e di software proprietari. Possono essere ottimizzati per determinate distribuzioni (anche se poi funzionano su quasi tutte) e generalmente è necessario controllare le dipendenze (se non previsto dall'installer).

Ogni distribuzione ha il suo *tool di gestione pacchetti*, ma esistono dei concetti di fondo che si devono capire prima di utilizzarne uno:

1. Quando si vuole installare un pacchetto, se ne deve conoscere il nome preciso. Ad esempio, alcune distribuzioni chiamano Firefox mozilla-firefox, altri semplicemente firefox. Tutti i tool di gestione pacchetti consentono di fare ricerche all'interno dei pacchetti da loro conosciuti per verificare se un software è già installato o meno sul computer.
2. I tool di gestione recuperano i pacchetti da archivi online chiamati *repository*. Un repository è un archivio dedicato soltanto ad ospitare pacchetti solo per una distribuzione specifica, dal quale essi possono essere scaricati e installati su un computer. La maggior parte delle distribuzioni Linux ha molte repository in tutto il mondo che rispecchiano il repository principale (mirroring). Normalmente, per consentire al tool di gestione dei pacchetti di funzionare basta soltanto impostare l'indirizzo web del repository di interesse.

Non esiste un tool di gestore di pacchetti migliore degli altri, funzionano tutti e sono tutti in grado di fare più o meno le stesse cose. Ogni distribuzione ha il suo tool di gestione pacchetti, ma alcune distribuzioni, come Fedora, permettono di usare più tool. L'unico difetto di questo metodo è che si affida ad Internet per installare il software, in quanto i repository sono spesso archivi online. Dunque, se non c'è connettività a Internet, non si può usufruire di questa funzionalità. Tuttavia, è possibile impostare come repository il CD sul quale sono scritti i file di installazione della distribuzione.

Di seguito viene presentata una lista nella quale associa ciascuna distribuzione al proprio tool di gestore pacchetti:

- Mandriva → Urpmi
- Debian e Ubuntu/Kubuntu/Xubuntu → APT
- OpenSuse → Yast/Smart
- Fedora → Yum
- Slackware → Swaret
- Gentoo → emerge
- Red Hat → Yum

2.4 Grafica e strumenti multimediali

Grafica e multimedia sono state a lungo appannaggio dei computer Apple Macintosh. Attualmente Linux manca ancora di programmi che siano equivalenti a, per esempio, Adobe Photoshop, ma quelli disponibili non sono da sottovalutare.

The GIMP[8] è un programma per l'elaborazione di immagini ed il fotoritocco. Non è del tutto equivalente a Adobe Photoshop, ma è sicuramente utilizzabile per molti scopi e offre anche alcuni strumenti per realizzare grafica per il World Wide Web dei quali invece Adobe Photoshop non dispone.

Inkscape è un programma per la grafica vettoriale alternativo a Adobe Illustrator. L'obiettivo del progetto è quello di fornire un potente strumento grafico per la grafica non professionale, supportando una piena compatibilità con gli standard XML, SVG e CSS.

ImageMagick è una suite di software per la creazione, modifica e visualizzazione di immagini. La suite può essere utilizzata per la conversione delle immagini tra diversi formati (ne supporta oltre 100). Consente inoltre di applicare le più comuni trasformazioni (traslazione, riflessione, rotazione, ecc.), regolare i colori delle immagini, applicare ad esse effetti speciali o inserirvi testo, linee, poligoni, ellissi o curve di Bézier. ImageMagick consente, inoltre, la manipolazione delle immagini attraverso script. Questa funzionalità è enormemente vantaggiosa per i web server e per tutti gli altri ambienti nei quali la grafica deve essere elaborata senza mouse e monitor.

[8] Acronimo di GNU Image Manipulation Program.

aTunes è un software scritto in Java per l'ascolto e l'archiviazione di file audio. Supporta i formati mp3, ogg, wma, wav, flac e mp4, e consente anche di effettuare il ripping di CD audio.

Audacity è un editor di file audio. Consente la registrazione, la riproduzione, la modifica e il mixaggio di un file audio.

Blender non consente semplicemente di effettuare editing video, ma anche di realizzare grafica e animazione 3D. Le sue caratteristiche, che normalmente si possono trovare solo in programmi a pagamento, lo rendono uno strumento veloce e potente alla portata di tutti coloro che vogliono lavorare e/o divertirsi con una dimensione in più.

Cinelerra e altri programmi come KDEnlive oppure OpenShot sono programmi per il "montaggio video non-lineare" che consentono di acquisire video da videocamere digitali, ricevitori TV o webcam, modificarlo e pubblicare il risultato su Youtube o masterizzarlo su DVD.

2.5 Servizi per Internet e Networking

Internet è la più grande rete mondiale esistente, costituita da centinaia di milioni di computer collegati tra loro. Offre all'utente i più svariati servizi, i principali dei quali sono il web, la posta elettronica, i trasferimenti di file. Inoltre è utilizzata per le comunicazioni più disparate: private e pubbliche, lavorative e ricreative, scientifiche e commerciali.

Senza Linux, Internet non sarebbe riconoscibile: Google possiede centinaia di migliaia di server equipaggiati con Linux come sistema operativo, così come i sistemi di trading delle maggiori Borse del mondo (in Germania, a Londra, a New York) si avvalgono delle ottime prestazioni garantite esclusivamente da Linux. E' un dato di fatto che la maggior parte del software Internet sia stato prima sviluppato su Linux, e che gran parte della ricerca universitaria su queste aree venga svolta su piattaforma Linux.

Apache HTTP Server è il nome della piattaforma web server più popolare su Internet – più della metà di tutti i siti web utilizzano server Apache.

Esistono altri web server di buona fattura per Linux - come Lighttpd - ma Apache resta, al momento, il web server più utilizzato.

MySQL e PostgreSQL sono database server relazionali. MySQL è tipicamente usato per i siti web, mentre PostgreSQL è un database innovativo e ad alte prestazioni per tutti gli scopi.

Postfix è un server di posta sicuro ed estremamente potente, utile per qualsiasi ambiente, da quelli home office fino agli ISP di grandi dimensioni.

FileZilla Server consente di trasferire file attraverso il protocollo FTP. Supporta, oltre a FTP, anche i protocolli SFTP e FTP su SSL/TLS.

2.6 Software infrastrutturale

Un server Linux è decisamente utile anche all'interno di una Local Area Network (LAN): è affidabile, veloce e necessita di poca manutenzione, tanto che è possibile installarlo e poi dimenticarsene (a parte i regolari backup!).

Samba consente di trasformare una macchina Linux in un server verso i client Windows per la condivisione di dischi e stampanti sulla rete. Con il nuovo Samba 4, un server Linux può anche funzionare come Controller di dominio Active Directory.

NFS[9] è un file system distribuito per sistemi Linux e Unix. Consente la condivisione in rete di dischi tra un server Linux e client Linux / Unix. Linux supporta la versione 4 di NFS che hanno migliori prestazioni e sicurezza.

OpenLDAP è l'equivalente Linux del Lightweight Directory Active Program, utile in reti di medie e larghe dimensioni.

DNS e *DHCP* costituiscono la base di una qualsiasi architettura di rete. Con *BIND*, Linux supporta il server DNS di riferimento, e il server ISC DHCP può collaborare con BIND per fornire ai client parametri di rete come l'indirizzo IP anche in reti di grandi dimensioni. Dnsmasq è un Server DNS e DHCP facile da usare per reti di piccole dimensioni.

2.7 Linguaggi di Programmazione e Ambienti di Sviluppo

Fin dai suoi inizi, Linux è sempre stato un sistema operativo realizzato da sviluppatori per sviluppatori. Di conseguenza, sono disponibili per Linux i compilatori e gli interpreti per tutti i linguaggi di programmazione più importanti. La famiglia dei compilatori Linux, ad esempio, supporta C, C++, Objective C, Java, Fortran e Ada. Naturalmente sono supportati anche i più diffusi linguaggi di scripting come Perl, Python, Tcl / Tk, Ruby, Lua, PHP, e i meno comuni linguaggi come Lisp, Scheme, Haskell, Prolog e Ocaml.

Un ricco arsenale di editor e di strumenti ausiliari rende lo sviluppo software su Linux un vero piacere. E' infatti disponibile l'editor standard, *vi*, così come lo sono gli ambienti di sviluppo professionale come GNU Emacs o Eclipse.

Linux è utile anche come ambiente di sviluppo di "sistemi embedded", vale a dire piccoli computer in esecuzione all'interno degli elettrodomestici di consumo, che siano basati su Linux oppure utilizzino sistemi operativi specializzati. Su un computer equipaggiato con Linux si può installare facilmente un compilatore che generi codice macchine per, ad esempio, processori ARM.

Linux, infine, viene utilizzato anche per sviluppare applicazioni per smartphone basati sul sistema operativo Android, e tutti gli strumenti professionali atti a questo scopo vengono forniti gratuitamente da Google.

9 Acronimo di "Network File System"

2.7.1 Framework di sviluppo

Come è stato detto precedentemente, grazie a Linux, gli sviluppatori hanno potuto progettare ed implementare framework senza incontrare ostacoli.

Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework c'è sempre una serie di librerie di codice utilizzabili con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software, come ad esempio un Integrated Development Environment (IDE), un debugger, o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito.

GTK+[10] é un toolkit per la creazione di interfacce grafiche sviluppato in C che supporta nativamente l'ambiente grafico X Window System e Microsoft Windows 2000. Pur essendo stato progettato inizialmente come ausilio alla programmazione per il noto programma di grafica GIMP, ha acquisito popolarità scavalcando le vecchie librerie Motif e divenendo parte fondamentale dell'ambiente desktop di Linux chiamato GNOME.

Symfony è un framework per il linguaggio PHP creato nativamente per lo sviluppo di applicazioni web. Il suo scopo è quello di aiutare gli sviluppatori web a creare potenti applicazioni in breve tempo.

Apache Lucene è un framework appartenente al progetto Apache Software Foundation, la quale mette a disposizione un insieme di API scritte in Java funzionali all'implementazione di potenti (efficienti ed altamente scalabili) motori di ricerca full-text per applicazioni J2EE. Lucene è stata successivamente reimplementata in Perl, C++, Python, Ruby e PHP.

Weka, acronimo di "Waikato Environment for Knowledge Analysis", è un ambiente software interamente scritto in Java e mette a disposizione un framework per progettare ed implementare applicazioni nel campo dell'apprendimento automatico. E' stato sviluppato dall'Università di Waikato in Nuova Zelanda e l'acronimo che lo identifica curiosamente corrisponde al nome di un simpatico animale simile al Kiwi, presente solo nelle isole della Nuova Zelanda.

Apache OpenNLP è un framework basato sull'apprendimento automatico per l'elaborazione di testi in linguaggio naturale. Supporta le attività più comuni nell'ambito dell'elaborazione del linguaggio naturale.

Domande:

1. Quale tra i seguenti programmi viene usato come client di posta elettronica su Linux?

A. Mozilla Thunderbird B. Microsoft Outlook

10 Acronimo che sta per "GIMP ToolKit"

C. Mozilla Firefox D. GIMP

2. FilleZilla Server è un programma per trasferire file attraverso il protocollo FTP?

A. Vero B. Falso

Quale tra i seguenti programmi viene usato per elaborare le immagini?

A. GIMP B. Mozilla Thunderbird

C. Cinelerra D. Mozilla Sunbird

Quale tra i seguenti programmi non fa parte delle applicazioni per la produttività?

A. Mozilla Firefox B. LibreOffice

C. Blender

Su quale framework si appoggia GIMP?

A. GTK+ B. OpenCV C. Symfony

Quale framework viene usato per costruire applicazioni legate all'ambito dell'elaborazione del linguaggio naturale?

A. OpenCV B. OpenNLP

C. Weka D. JRE

Quale protocollo di rete permette di accedere ai file memorizzati su un altro sistema Linux?

A. SMTP B. NFS C. PHP

D. DNS E. DHCP

OpenOffice e LibreOffice sono due suite per la produttività molto simili.

A. Vero B. Falso

Risposte:

1. A
2. A
3. A
4. C
5. A
6. C
7. B
8. A

3 Software open source e tipi di Licenze

3.1 La proprietà intellettuale: copyright, marchi e brevetti

Il software, così come i libri e i film, è un'opera di ingegno, e come tale è soggetto alla proprietà intellettuale (Intellectual Property) per la quale esistono apposite normative.

In ambito nazionale, con la legge sul diritto d'autore n. 633/41 e relative novellazioni, si garantiscono all'autore di un'opera i seguenti diritti:

- Diritti morali:
  - diritto alla paternità dell'opera, cioè il diritto di rivendicare la propria qualità di autore dell'opera;
  - diritto all'integrità dell'opera, cioè il diritto di opporsi a qualsiasi deformazione o modifica dell'opera che possa danneggiare la reputazione dell'autore;
  - diritto di pubblicazione, cioè il diritto di decidere se pubblicare o meno l'opera.
- Diritti di utilizzazione economica (Diritti Patrimoniali):
  - diritto di riproduzione, cioè il diritto di effettuare la moltiplicazione in copie dell'opera con qualsiasi mezzo;
  - diritto di esecuzione, cioè il diritto di presentare l'opera al pubblico;
  - diritto di diffusione, cioè il diritto di effettuare la diffusione dell'opera a distanza;
  - diritto di distribuzione cioè il diritto di porre in commercio l'opera;
  - diritto di elaborazione cioè il diritto di apportare modifiche all'opera originale.

Tutti questi diritti permettono all'autore di autorizzare o meno l'utilizzo della sua opera e trarne dei benefici economici.

Non ci sono particolari procedimenti per ottenere la proprietà intellettuale su un'opera di ingegno di carattere creativo, infatti essa viene automaticamente attribuita all'autore al momento della realizzazione dell'opera.

La proprietà intellettuale comprende:

- Il marchio
- Il brevetto
- Il copyright

Il marchio (simboli, nomi e loghi) è registrato presso le autorità competenti (in Italia presso l'Ufficio Italiano Brevetti e Marchi) ed è valido per dieci anni o fino a quando il proprietario di esso decide di rinunciare al rinnovo. La registrazione del marchio

consente la tutela giuridica di esso: per poter essere registrato, un marchio deve soddisfare I requisiti di originalità, verità, novità e liceità.

Il brevetto (idea) è un contratto stipulato tra il richiedente e lo Stato, il quale garantisce il diritto in esclusiva per lo sfruttamento economico dell'invenzione in cambio del rilascio al pubblico di essa dopo un certo periodo di tempo (solitamente venti anni per il brevetto industriale). Per ottenere il brevetto, l'invenzione da brevettare deve essere sottoposta a vari esami e soddisfare I requisiti di novità, attività, inventiva, industrialità.

Attualmente non è possibile brevettare un software, ma solamente gli algoritmi che esso usa.

Oltre al marchio e al brevetto, il terzo aspetto della proprietà intellettuale è il copyright o diritto d'autore.

Il termine copyright significa letteralmente “diritto di copia”. Ogni autore può dare licenza a un editore di distribuire una sua opera, in modo da poter guadagnare da essa senza necessità di occuparsi dei costi di pubblicazione, diffusione, commercializzazione di essa.

Verso la fine del ventesimo secolo, l'avvento dei computer e di Internet ha cambiato lo scenario, consentendo a chiunque fosse in possesso di un computer di effettuare copie di contenuti digitali (software, musica, video) e distribuirle a piacimento.

Ciò ha causato un enorme danno alle case produttrici di musica, film e software, che guadagnano dalla vendita di contenuti digitali e non traggono alcun vantaggio dalla loro libera circolazione.

Da allora le case produttrici hanno chiesto leggi più severe e pene più aspre verso coloro che effettuano “copie pirata” di contenuti digitali.

E’ possibile catalogare le principali tipologie di copia di un software nel modo seguente:

- copia necessaria all'uso del programma;
- copia di riserva (backup);
- copia effettuata per lo studio del programma;
- copia per decompilare il programma (per ottenere l’interoperabilità con gli altri programmi).

Solitamente le ultime due operazioni sono consentite da parte della Comunità Europea (direttiva 91/250/CEE) nell’utilizzo dei software proprietari; occorre però fare attenzione poiché, un utilizzo errato potrebbe portare ad una violazione del diritto d’autore, per esempio ricavando l’algoritmo che sta alla base dello stesso e utilizzarlo senza autorizzazione.

3.2 Software proprietario e software libero

Da quanto abbiamo visto, dunque, secondo le leggi che regolano il copyright è illegale per il fruitore di un software copiarlo senza esplicito consenso dei detentori del diritto d'autore.

Attraverso un documento chiamato licenza l'autore di un software stabilisce i diritti dell'utente che lo ha comprato o scaricato.

In base al tipo di licenza utilizzata, possiamo distinguere tre principali tipologie di software: software proprietario, software di pubblico dominio e software open source.

Il Software proprietario si caratterizza per il fatto che l'autore non fornisce al fruitore di esso il codice sorgente. Le licenze dei software proprietari normalmente comportano maggiori restrizioni rispetto a quanto previsto dalla legge sul diritto d'autore. Solitamente l'autore non firma tali licenze ma le accetta cliccando su un'apposita voce di menu al momento dell'installazione o primo utilizzo del programma. Queste licenze vengono comunemente chiamate EULA (End-User License Agreement).

Il software proprietario si distingue ulteriormente in:

- Commerciale: distribuito a pagamento (nei negozi o tramite Internet) per trarne un profitto. Ridistribuirlo è illegale.
- Shareware: è possibile usarlo e copiarlo ma solo entro certi limiti (temporali o di funzionalità) oltre i quali è necessario pagare per l'uso. Possono essere previsti:
    - l'utilizzo entro un certo termine;
    - l'utilizzo di una sola parte del programma;
    - l'utilizzo del programma in forma "disturbata" (il cosiddetto nagware).
- Demo: è una versione castrata del programma, viene distribuita per mostrare le "potenzialità" dello stesso in modo da dare la possibilità di provarlo prima di dover acquistare la licenza d'uso.
- Freeware: può essere utilizzato gratuitamente al fine di promuovere un prodotto più completo (da non confondere con il "free software"). Il freeware rappresenta uno dei tipi di software maggiormente presenti in rete. Può essere copiato ed utilizzato gratuitamente, ma il codice sorgente non può essere utilizzato in assenza del consenso dell'autore. A questa categoria di programmi appartiene il diffuso cardware, che può essere copiato ed utilizzato da chiunque, a condizione che venga inviato all'autore una comunicazione, nonché una somma simbolica a compenso della propria fatica. Anche in questo caso, l'autore non si spoglia dei diritti derivanti dalla paternità dell'opera.

Il software di pubblico dominio è un software il cui autore ha deciso di rinunciare ai diritti d'autore (o sono passati un certo numero di anni dalla sua morte). E' quindi possibile copiare e modificare il programma senza autorizzazione.

Il software *open source* è caratterizzato dal fatto che il suo codice sorgente viene reso disponibile pubblicamente. Si tratta quindi di una sorta di "sistema aperto" che chiunque può migliorare con il proprio contributo. Il software open source fornisce la licenza (che

garantisce maggiori diritti all'utente) in un file chiamato COPYING. Tale licenza di solito non deve essere espressamente accettata dall'utente.

La diffusione del software open source deve molto a tre organizzazioni:

- Free Software Foundation (FSF)
- open source Initiative (OSI)
- Creative Commons

Ciascuna di queste organizzazioni ha una propria filosofia in merito all'open source, che analizziamo nei paragrafi seguenti.

3.2.1 Free Software Foundation

La Free Software Foundation (FSF) è un'organizzazione no-profit fondata nel 1985 da Richard Stallman, considerato il padre della filosofia “Free Software”. Questa dicitura non significa che il software sia gratis, bensì che il fruitore possa usarlo con maggiore libertà rispetto a quella offerta da un software proprietario. In particolare si parla di quattro libertà, e cioè:

1. Libertà di eseguire il programma, per qualsiasi scopo (libertà 0).
2. Libertà di studiare come funziona il programma e di modificarlo in modo da adattarlo alle proprie necessità (libertà 1).
3. Libertà di ridistribuire copie in modo da aiutare il prossimo (libertà 2).
4. Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti in modo da permettere a tutta la comunità di trarne beneficio (libertà 3).

L'accesso al codice sorgente è un prerequisito per le libertà 1 e 3.

Nella filosofia FSF tutto il sofware dovrebbe essere liberamente distribuito (o meglio condiviso) con il codice sorgente e con tutte le libertà appena descritte. Il software proprietario non dovrebbe esistere.

Le licenze che si ispirano alla FSF sono spesso chiamate *copyleft* (in contrapposizione al copyright), e garantiscono libertà che sono l'esatto opposto rispetto a quanto previsto dal diritto d'autore.

La principale licenza software della FSF è la GNU GPL (General Public License) o semplicemente GPL. GPL garantisce le libertà previste dalla FSF ed impone che tutti i prodotti derivati da software libero vengano distribuiti con la medesima licenza, al fine di evitare che un'azienda si appropri di un software libero.

GPLv2 è stata rilasciata nel 1991 mentre nel 2007 è stata approvata la GPLv3 con l'obiettivo di migliorare la precedente versione e adeguarsi alle nuove leggi e regolamenti. GPLv3 contiene inoltre clausole tese a combattere l'uso di restrizioni hardware che limitano le quattro libertà FSF.

Il kernel Linux utilizza la licenza GPLv2.

3.2.2 Open Source Initiative

Il movimento del “free software” prese rapidamente piede in alcuni ambienti, in particolare nel mondo accademico e tra gli appassionati. Le imprese, tuttavia, furono restie ad adottare il “free software”, in quanto lo consideravano erroneamente “gratuito” invece che “libero” e ciò rappresentava una minaccia per il proprio business.

Per ammorbidire alcuni degli imperativi della FSF, nel 1998 Eric S.Raymond, Bruce Perens e Tim O'Reilly diedero vita alla Open Source Initiative (OSI), nell'ambito della quale venne coniato il termine “open source” e i 10 principi ad esso legati:

1. Libera redistribuzione;
2. Disponibilità del codice sorgente;
3. Possibilità di apportare modifiche;
4. Integrità del codice sorgente originale (la licenza può impedire la distribuzione del codice sorgente in forma modificata, a patto che venga consentita la distribuzione dell'originale accompagnato da ”patch”);
5. Nessuna discriminazione contro persone o gruppi;
6. Nessuna discriminazione per campo di applicazione del prodotto;
7. Distribuzione della licenza (i diritti legati a un software devono essere applicati a tutti coloro a cui il software viene distribuito, senza che sia necessaria l'emissione di ulteriori licenze);
8. Specificità ad un prodotto (i diritti legati a un software non devono dipendere dal fatto che il software sia usato o distribuito come parte di un programma più grande);
9. Assenza di vincoli su altro software (la licenza non deve porre restrizioni su altro software distribuito insieme a quello licenziato);
10. Neutralità rispetto a particolari tecnologie o tipi di interfacce.

L'OSI ha certificato come open source diverse licenze (tra cui anche GPL) e non tutte prevedono che i prodotti derivati da software libero vengano distribuiti con la medesima licenza dell'originale.

In linea generale il free software di FSF è anche open source, mentre molte licenze open source non rientrano nella definizione di software libero della FSF.

Nonostante le divergenze esistenti, entrambe le organizzazioni FSF e OSI condividono i medesimi obiettivi e le differenze sono minime. Per questo, alle volte, per indicare il software non proprietario la comunità usa i termini FOSS (Free and Open Source Software) oppure FLOSS (Free, Libre and Open Source Software).

3.2.3 Creative Commons

Creative Commons è un'organizzazione no-profit fondata nel 2001 da Lawrence Lessig con lo scopo di ampliare la gamma di opere creative disponibili alla condivisione e all'utilizzo pubblici in maniera legale. Si basa sul presupposto che la maggior parte della nostra cultura attuale deriva dai precedenti lavori culturali.

Mentre FSF e OSI si concentrano sul software, Creative Commons ha un campo di applicazione più ampio: fornisce sei licenze progettate per scopi differenti e pensate per permettere all'autore di un'opera di donarla al pubblico in modo controllato.

3.3 Licenze open source

Un utente normale non ha generalmente bisogno di conoscere in dettaglio le licenze open source, mentre per le aziende, e in particolare per le software house, è importante conoscerle e capirne le differenze in modo da utilizzarle come parti del proprio modello di business.

Se si modifica un programma è quindi necessario prestare attenzione ai requisiti di redistribuzione, in particolare se si intendono unire due o più programmi o distribuire un programma sotto una licenza differente.

Abitualmente è più conveniente utilizzare una licenza già utilizzata e certificata da un ente (e dai suoi avvocati) piuttosto che scriverne una nuova.

Ecco un panoramica generale sulle licenze comunemente più usate.

3.3.1 *Licenze GNU:*

Ecco le principali licenze GNU:

- ***GNU GPL (v2 e v3)**:* precedentemente trattata, la GPLv2 è usata dal kernel di Linux
- ***GNU LGPL:*** è considerata più permissiva rispetto alla GPL. Viene utilizzata principalmente sulle librerie per permettere ai programmi che le usano di essere distribuiti con licenze differenti (anche a pagamento). Tuttavia, se vengono apportate modifiche al software protetto dalla LGPL, esso deve essere redistribuito con la stessa licenza.
- **GNU FDL:** fornisce una licenza libera per i documenti ed i manuali, permette di fornire le stessi libertà che la GPL fornisce al software ma applicato a materiale testuale: si è liberi di utilizzare, copiare, distribuire che sia per uso commerciale che non commerciale. E' stata pensata principalmente per fornire manuale libero per software libero, ma viene utilizzata anche per manuali sul mondo di Linux e per altri tipi di documenti.

3.3.2 *Licenze BSD e derivate:*

Le licenze BSD (Berkeley Software Distribution) furono create dall'università californiana per lo sviluppo del proprio sistema operativo Unix-Like. Si differenziano dalla licenza GNU GPL perché permettono di ridistribuire le modifiche apportate ad un opera sotto un'altra licenza diversa da quella originale e non obbligano inoltre di rilasciare il codice sorgente al pubblico.

Nel Luglio 1999 venne pubblicata una licenza BSD ulteriormente aperta detta Modified BSD License (MBSDL) che si caratterizza per l'eliminazione della clausola pubblicitaria e

conseguente passaggio da quattro a tre clausole. La licenza MBSDL è approvata sia dalla FSF che da OSI ed è compatibile con GPL.

3.3.3 *Apache license:*

La licenza Apache è stata scritta dalla Apache Software Foundation. Essa consente l'uso e l'adozione di codice licenziato senza richiedere che codice modificato e licenziato da Apache venga reso pubblico.

3.3.4 *Mozilla public license:*

La Mozilla Public License (MPL) è una licenza open source e free software compatibile con GPL usata dai software Mozilla Firefox e Thunderbird; deriva dalla Netscape Public License (NPL) creata quando Netscape pubblicò il codice sorgente del suo web browser.

3.3.5 *Creative Commons:*

La Creative Commons offre varie licenze tra cui scegliere, che differiscono tra di loro per i termini che applicano all'opera da sottoporre a licenza d'uso.

I termini possibili sono:

- Attribution: permette la modifica, la copia e la ridistribuzione dell'opera affinché venga riconosciuta la paternità di essa all'autore originale;
- ShareAlike: permette la creazione di opere derivate solamente se vengono ridistribuite con la stessa licenza;
- No Derivative Works: nega la creazione di opere derivate dall'opera originaria; permette però la copia, distribuzione ed esibizione dell'opera originaria;
- Non-commercial: nega l'utilizzo commerciale delle opere derivate senza autorizzazione dell'autore originale.

La licenza Attribution-ShareAlike 3.0 Unported rispecchia i principi del Free Software, tuttavia Creative Commons sconsiglia di associare le proprie licenze al software poiché potrebbero insorgere delle imprecisioni ed ambiguità.

Creative Commons non vieta all'autore di poter ridefinire i termini della licenza in un secondo momento prendendo accordi con l'autore di un'opera derivata, per esempio permettendo di utilizzare la propria opera per scopi commerciali dove non lo fosse consentito.

3.4 Il modello di business del software open source

Come si può ottenere denaro dallo sviluppo di software libero? Si possono seguire diversi approcci:

- Servizi e supporto: il prodotto software può essere open source e fornito gratuitamente ai clienti, mentre si possono offrire a pagamento servizi come il supporto tecnico e il training degli utenti. Per esempio, un videogioco può essere open source ma richiedere l'iscrizione a un servizio online per poter accedere a una serie di funzionalità;

- Dual licensing: un'azienda può creare due versioni del prodotto software: una completamente open source e una a pagamento che contenga più funzionalità rispetto a quella open source;
- Driver open source: alcuni produttori di hardware hanno interesse nel rilasciare driver open source per incentivare le vendite del loro hardware. Lo sviluppo di driver open source può essere molto remunerativo per le aziende che producono hardware, infatti un driver open source può godere di miglioramenti anche da altre persone non appartenenti alla casa produttrice dell'hardware permettendo così di assicurare un supporto ai driver più lungo;
- Donazioni: le donazioni possono incentivare gli sviluppatori a continuare a migliorare il software, implementando funzionalità e risolvendo bug; inoltre possono anche coprire i costi di gestione delle infrastrutture;
- Ricompense: gli utenti possono offrirsi di pagare una casa produttrice affinché sviluppi un certo software o nuove funzionalità di un software già esistente. Esistono appositi siti[11] nei quali gruppi di utenti si riuniscono e finanziano lo sviluppo di codice per un progetto comune, che individualmente non sarebbero in grado di sviluppare.

Esercizi

Avete mai sentito parlare di programmi FOSS e quali sono? Sono meglio o peggio rispetto alle alternative commerciali? Se si, dite il perchè. Altrimenti, spiegate perchè no. [Max 5-10 righe]

Domande:

1. Free software e software open source sono la stessa cosa?

A. Vero B. Falso

2. Quale dei seguenti non è un prerequisito affinché un software possa essere definito open source?

A. La licenza non deve discriminare contro persone

B. La licenza non deve richiedere che un software sia distribuito come parte di uno specifico prodotto

C. La licenza deve richiedere che le modifiche siano distribuite con la stessa licenza

D. Il programma deve essere fornito con il codice sorgente

E. La licenza deve essere applicata a tutti coloro a cui il programma è redistribuito

3. Quale dei seguenti è uno dei principi della filosofia FSF?

A. Gli sviluppatori devono usare l'ultima versione della GPL

[11] http://www.fossfactory.org

B. Gli utenti dovrebbero avere il diritto di modificare il software libero e distribuirlo con licenza commerciale

C. Gli sviluppatori dovrebbero sviluppare software solamente per sistemi operativi liberi

D. Gli utenti devono avere il diritto di utilizzare il software come preferiscono

4. La legge sul diritto d'autore disciplina, nella maggior parte dei Paesi, le modalità di distribuzione del software.

A. Vero B. Falso

5. La definizione di software libero della FSF e i dieci principi della OSI per il software open source richiedono entrambe che gli utenti abbiano la possibilità di esaminare il codice sorgente

A. Vero B. Falso

6. Quale di queste organizzazioni promuove i principi dell'open source non solamente nel campo dello sviluppo software?

A. FSF B. OSI C. Creative Commons

7. Un software che è distribuito gratuitamente, ma che richiede il pagamento se viene usato oltre certi limiti, è:

A. Freeware B. Open Source

C. Shareware D. Malware

Risposte:

1. B
2. C
3. D
4. A
5. A
6. C
7. C

ISBN: 9788866850090

4 Abilità ICT e lavoro su Linux

4.1 Linea di comando e interfaccia grafica

Come abbiamo detto, il kernel è il nucleo di ogni sistema operativo, ed è trasparente all'utente, nel senso che gli utenti non manipolano direttamente il kernel. Al contrario, gli utenti interagiscono con una serie di altri programmi che sono associati con il particolare sistema operativo scelto.

Tali programmi includono, oltre ai software per la produttività, per la programmazione e per i framework di cui abbiamo già discusso, anche la linea di comando e l'interfaccia grafica.

La linea di comando ha costituito fino a qualche anno fa l'unico metodo di interazione disponibile tra utenti e computer. Una linea di comando è un programma, chiamato *shell*, all'interno del quale è possibile digitare i comandi necessari alla manipolazione di files, all'esecuzione di programmi e così via.

Anche se molti computer oggi non usano più questa modalità di interazione, essa è di fondamentale importanza per l'utilizzo avanzato di Linux. Infatti, abitualmente non viene installato alcun ambiente grafico sui server, sia per un risparmio di risorse hardware, sia per motivi di sicurezza, sia perché le interfacce grafiche variano tra distribuzioni e nel tempo, mentre i comandi restano sempre gli stessi.

L'interfaccia grafica (*Graphical User Interface, GUI*) rappresenta l'evoluzione della shell per quanto riguarda l'esperienza utente. Avendo a disposizione un'interfaccia grafica, per interagire con il computer non è più necessario conoscere comandi complicati, ma basta muovere il puntatore del mouse sullo schermo e cliccare su menu e icone: questo rende l'uso del computer possibile anche a utenti inesperti ed è l'aspetto che più di ogni altro ne ha favorito la diffusione su larga scala.

Windows e OS X hanno le loro interfacce grafiche proprietarie. Linux invece si basa su una GUI di nome X Windows System, abbreviato spesso in X.

X dal canto suo è una GUI abbastanza limitata, motivo per cui viene arricchita con un ambiente desktop aggiuntivo, che varia a seconda della distribuzione in uso. Sebbene le distribuzioni abbiano il proprio ambiente grafico predefinito, è possibile installarne anche altri (attraverso il gestore dei pacchetti) per provarli. Quando su un computer sono installati più ambienti desktop, è possibile scegliere quale di essi utilizzare all'avvio della sessione di lavoro.

La differenza di interfaccia grafica rispetto a Windows e OS X rappresenta il primo scoglio per chi si approccia per la prima volta all'utilizzo di Linux.

Gli utenti alle prime armi preferiscono solitamente lavorare con un'interfaccia grafica piuttosto che tramite linea di comando, ed è per questo che abitualmente i sistemi Linux (tranne alcuni come debian, Arch e Gentoo) installano e avviano in modalità predefinita la GUI. E' possibile effettuare lo switch da ambiente Desktop a shell tramite la combinazione di tasti Ctrl+Alt+F1 o Ctrl+Alt+F2. Per tornare dalla shell alla GUI basta usare la combinazione Alt+F1 o Alt+F7.

4.2 Primi passi su Linux

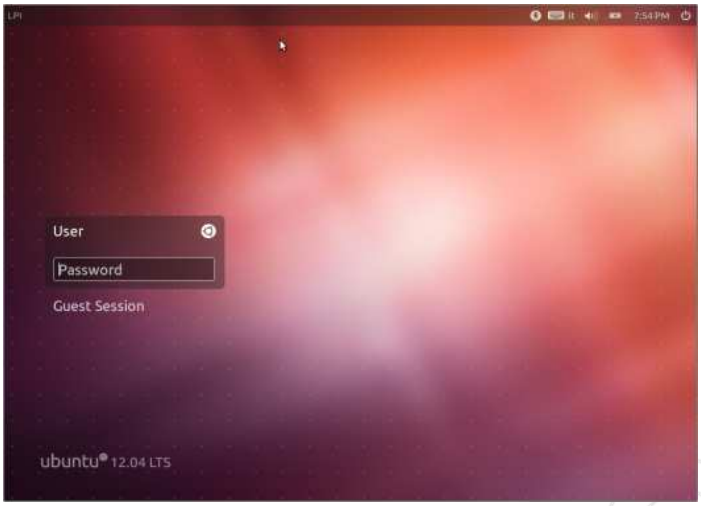
Sia che si utilizzi la shell o l'interfaccia grafica, per poter usare Linux è necessario disporre di un account utente abilitato a operare sul computer. Infatti, i sistemi Linux prevedono l'esistenza di diversi utenti riconosciuti attraverso il loro account (composto da nome utente e password). La creazione degli account utente è un compito normalmente demandato all'amministratore del sistema.

E' importante notare che la password dovrebbe restare segreta, dal momento che chiunque dovesse esserne a conoscenza potrebbe accedere il computer e prendere il controllo dei file, della posta elettronica e così via.

In molte distribuzioni, se sul sistema è definito un solo account utente, è possibile snellire la procedura di accesso al sistema (log in) evitando di inserire sempre nome utente e password. Questa procedura può andare bene se il computer che si sta utilizzando fa parte di una postazione domestica, mentre è assolutamente da evitare, per ragioni di privacy, in ambienti in cui più persone possono avere accesso al computer.

4.2.1 *Accesso tramite ambiente grafico*

Solitamente, quando un sistema Linux è provvisto di ambiente grafico, l'accesso al sistema avviene attraverso un form mostrato a video nel quale è possibile inserire nome utente e password. Notare che la password digitata viene nascosta tramite degli asterischi: questa è una precauzione per impedire a chi è alle vostre spalle di leggere la password dallo schermo stesso.



Dopo aver eseguito l'accesso, il computer avvia la sessione grafica, dalla quale è possibile accedere ad applicativi e programmi tramite interfaccia punta e clicca. La maggior parte degli ambienti grafici per Linux supporta la gestione della sessione, consentendo di riprenderla nello stesso stato in cui era finita la volta precedente. In questo modo non è necessario ricordare quali programmi erano in esecuzione, quali finestre erano aperte e quali file erano in uso.

Una volta terminata la sessione di lavoro è consigliabile uscire dal sistema (log out). Questo è importante anche per consentire al gestore della sessione di salvare la sessione corrente per la volta successiva.

4.2.2 *Accesso tramite linea di comando*

A differenza di quanto succede sui computer per uso personale, i sistemi server spesso supportano solo l'accesso tramite linea di comando attraverso la rete (quindi senza accesso fisico alla tastiera del server). In questi casi non c'è nessuna schermata introduttiva, ma il computer chiede direttamente nome utente e password:

computer login:

Assumiamo che “computer” sia il nome dell'host al quale vogliamo accedere. Qui bisogna inserire il nome utente e premere il tasto INVIO. Il computer successivamente chiederà la password:

Password:

Ora bisogna inserire la password. In questo caso non verrà mostrato niente a video, neanche gli asterischi.

Se nome utente e password sono corretti, il sistema consente l'accesso all'utente e fa partire la shell. A questo punto è possibile inserire i comandi e lanciare programmi. Dopo aver effettuato l'accesso, il sistema posiziona automaticamente l'utente nella sua “cartella home”, dalla quale sarà possibile navigare nei file.

Per uscire dal sistema tramite shell basta usare il comando *logout*.

4.3 Ambienti grafici e browser

4.3.1 *Ambienti grafici*

Quando si esegue l'accesso a un sistema Linux tramite ambiente grafico, il computer mostra un desktop che non è molto diverso da quello che si vedrebbe su un qualunque altro computer moderno. Purtroppo non è possibile essere più specifici, perché su Linux non esiste un ambiente grafico ufficiale. Al contrario, ogni distribuzione ha il suo ambiente grafico predefinito.

I più diffusi ambienti desktop esistenti sono:

- GNU Object Model Environment (GNOME), basato sul GIMP Tool Kit (GTK+), predefinito in Fedora e Debian
- K Desktop Enviroment (KDE), basato sul widget set Qt, ambiente desktop predefinito in Mandriva e SUSE
- Unity: sviluppato dalla Canonical Ltd, predefinito in Ubuntu, punta a essere un ambiente desktop di facile utilizzo
- LXDE e XFCE: ambienti desktop leggeri che ricordano KDE e GNOME, ma sono maggiormente focalizzati sull'uso economico delle risorse.

Anche se due distribuzioni usano lo stesso ambiente grafico, questo non vuol dire che la loro interfaccia utente visualizzata a schermo sia la stessa. Infatti, gli ambienti grafici forniscono moltissime possibilità di personalizzazione del look and feel, basate su temi ed estensioni, e le distribuzioni fanno largo uso di questi strumenti per differenziarsi le une dalle altre.

In ogni caso, generalmente nella parte superiore o inferiore dello schermo c'è una barra di controllo che consente, tramite click su voci di menu, di accedere alle applicazioni o di uscire dal sistema e/o spegnerlo. KDE usa un pannello che ricorda da lontano quello di Windows e un pulsante simile al pulsante “Start” che apre un menu di programmi, mentre il resto della barra mostra icone per le applicazioni in esecuzione e per l'orologio, lo stato della rete e così via.

GNOME non usa un tasto “Start”, ma la sua barra delle applicazioni si trova nella parte alta dello schermo. I programmi più importanti sono accessibile tramite menu che compaiono al di sotto della barra, la cui parte destra è dedicata alle icone che mostrano lo stato del sistema.

L'ambiente grafico tipicamente fornisce un “file manager” che consente l'accesso alle cartelle su disco e la manipolazione dei file e delle sottocartelle. Queste procedure non differiscono molto da quelle disponibili su altri ambienti grafici: è possibile copiare o muovere file trascinandoli da una finestra all'altra oppure aprire un menu di contesto cliccando con il tasto destro su un file.

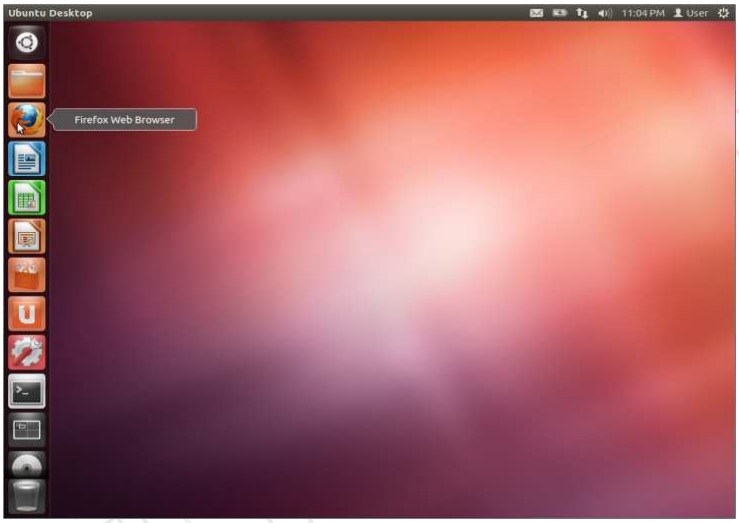
Un'utile funzione messa a disposizione dalla maggior parte degli ambienti grafici di Linux consiste nel “desktop virtuale”. Tramite questa funzionalità è possibile moltiplicare lo

spazio disponibile sullo schermo passando avanti e dietro tra vari desktop simulati, ciascuno dei quali ha una selezione di finestre di determinati programmi.

4.3.2 *Browser*

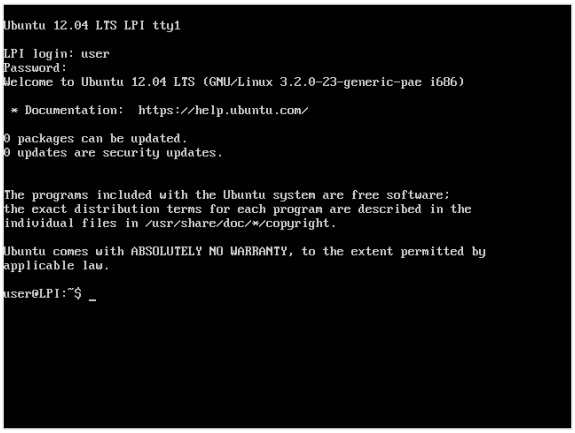
I browser fanno parte dei programmi più importanti dei computer moderni. Fortunatamente, i browser più popolari sono programmi open source, e Firefox e Google Chrome sono disponibili per Linux così come lo sono per Windows e OS X.

Tipicamente il browser è accessibile tramite la barra delle applicazioni: basta cercare nel menu la voce “Internet”.



4.3.3 *Terminale e shell*

Anche dall'interno di un ambiente grafico Linux può servire far partire una finestra di terminale nella quale inserire comandi in una shell. In KDE su Debian, ad esempio, c'è una voce chiamata “Konsole (Terminal)” nello start menu sotto la voce “System”. La finestra di terminale è inoltre accessibile in molte distribuzioni cliccando con il tasto destro sulla scrivania, e poi sulla voce “Open Terminal”.



Il prompt della shell che viene presentato all'utente è solitamente composto dalla concatenazione del nome dell'utente, del simbolo @ (at), del nome della macchina, del simbolo : (due punti) e della directory nella quale ci si trova:Per brevità, all'interno degli esempi presenti in questo libro useremo il simbolo $ (dollaro) come abbreviazione per identificare il prompt dell'utente normale (senza privilegi di amministrazione).

Comandi visti nel capitolo

logout Termina una sessione shell

Domande

1. Se si effettua il login in modalità grafica non è possibile eseguire i comandi in modalità testuale

A. Vero B. Falso

2. Un computer Linux utilizzato come server non necessita generalmente di X

A. Vero B. Falso

3. Quali dei seguenti sono ambienti desktop Linux (scegliere tre risposte)?

A. GTK+ B. GNOME C. KDE

D. Evolution E. XFCE

Risposte

1. B
2. A
3. B, C, E

ISBN: 9788866850090

Parte 2:
Introduzione alla linea di comando

ithum®

5 Linea di comando base

5.1 La Command Line Interface

Un tool indispensabile di un amministratore di sistema Unix (e ovviamente Linux) è sicuramente la Command Line Interface (CLI), l'interfaccia a riga di comando, chiamata anche Text User Interface per differenziarla dalla Graphical User Interface – la GUI, l'interfaccia utente grafica.

Sostanzialmente si tratta di un programma che interpreta ed esegue comandi letti da file o standard input (quindi generalmente inseriti tramite tastiera) e che comunica con l'utente per mezzo di messaggi mostrati su standard output (il monitor usualmente).

Questo modo di amministrare è molto utilizzato soprattutto in virtù della sua rapidità di esecuzione e della sua versatilità.

La CLI nella sua veste grafica viene chiamata "*emulatore di terminale*", questo per differenziarla dall'originale *"terminale*", un apparecchio ormai pressoché in disuso. Nell'uso odierno l'accezione "terminale" sta ad indicare l'emulatore di terminale.

Dipendentemente dall'ambiente grafico utilizzato, l'emulatore di terminale di default ha differenti nomi, ma la sostanza è sempre la stessa. Ad esempio, in un ambiente grafico GNOME si chiama *gnome-terminal*, in KDE invece *konsole*, in XFCE *xfce4-terminal*, e cosi via. Esistono comunque anche altri emulatori di terminale che non sono legati al window manager utilizzato, con caratteristiche analoghe, ma generalmente più scarni di funzionalità accessorie e quindi meno esigenti nella richiesta di risorse di sistema, come ad esempio *xterm*, *xvt*, *aterm*, giusto per citare i più rinomati.

5.2 La Shell

Sia gli emulatori di terminale che le console una volta avviati lanciano l'esecuzione di una "*shell*" che è il vero e proprio programma che interpreta i comandi ivi digitati: una "conchiglia" al cui interno esiste un intero ecosistema con variabili d'ambiente e con un set di comandi interni che permettono di:

- spostarsi nel file system;
- copiare, rinominare, rimuovere file e directory;
- gestire i processi in esecuzione e molto altro, tra cui anche la possibilità di lanciare l'esecuzione di comandi e applicazioni non integrate nella shell stessa, intesa come programma, detti pertanto comandi esterni (si veda il paragrafo 5.4, "Comandi interni e comandi esterni").

Mentre la scelta dell'emulatore di terminale è una questione quasi estetica, quella della shell può invece rappresentare una scelta funzionale, soprattutto quando si parla di

scripting, cioè programmazione della shell. Vi sono molte shell disponibili negli ambienti Unix e citando quelle maggiormente diffuse troviamo:

- *bash*, la Bourne Again Shell, per differenziarla dall'originale shell ("*sh*") di Bourne. È compatibile con gli standard IEEE POSIX e implementa funzionalità più avanzate di derivazione della ksh e csh. Generalmente è la shell di default nelle distribuzioni Linux.
- *tcsh*, la TENEX C Shell, una versione avanzata dell'originale Berkeley Unix C Shell.
- *ksh*, la Korn Shell, nata come estensione (*superset*) dell'originale *sh*, default in sistemi Unix come IBM AIX e SUN Solaris.
- *zsh*, una shell moderna con funzionalità molto avanzate
- *dash*, la Debian Almquist Shell, una shell standard POSIX leggera con poche dipendenze di librerie, più robusta e veloce della bash e pertanto ben adatta per l'esecuzione di *script* (che vedremo più sotto).

Ogni shell ha dei file di controllo e di configurazione che vengono salvati all'interno della home utente. Tali file hanno lo scopo di definire e personalizzare l'ambiente di lavoro di quell'utente specifico. Prendiamo in analisi la shell di default nella maggior parte degli ambienti Linux, la bash. Nella home dell'utente possiamo trovare almeno tre file nascosti – il loro nome inizia con un carattere punto (".") – relativi a questa shell:

- *.bashrc* : contiene comandi e configurazione di variabili d'ambiente che vengono richiamati ogni qual volta viene lanciata una shell bash; praticamente un file di inizializzazione dell'ambiente su cui si andrà a lavorare.
- *.bash_history*: registra un certo numero degli ultimi comandi inseriti da tastiera. Nella bash il default è 500.
- *.bash_logout*: contiene i comandi da eseguire nel momento in cui l'utente chiude la shell. Di solito è vuoto o richiama un comando che pulisce la console, ma può essere personalizzato ad esempio per lanciare un backup, per mandare una e-mail o altro.

Per ogni shell esistono file di controllo analoghi a quelli ora mostrati.

5.3 Caratteristiche avanzate delle Shell moderne

Le attuali shell, rispetto a quelle più datate, hanno una serie di caratteristiche che ne aumentano di molto le potenzialità, influenzando positivamente la produttività dell'utente che le utilizza. In elenco vengono riportate le caratteristiche di spicco di alcune shell moderne, come la bash o la zsh.

Vediamo queste caratteristiche:

1. personalizzazione del promt (*prompt customization*)
2. auto-completamento dei comandi (*command completion* o *tab completion*)
3. storico dei comandi shell (*shell history*)
4. modificatori di parola (word modifiers)
5. modifica della linea di comando (*line editing*)

6. correzione della digitazione (*spelling correction*)
7. alias

Analizziamole in modo un poco piú dettagliato, ponendo a corredo qualche utile esempio.

1. *Prompt customization*

La prima cosa che si vede una volta avviata la shell è il *prompt*, una sequenza di uno o più caratteri che avvisano l'utente che la shell è pronta per accettare comandi. Generalmente termina con il carattere dollaro ($) o percento (%) per indicare che nella shell ci si è autenticati come utente standard oppure il carattere cancelletto (#) che di solito si usa per indicare che i comandi impartiti vengono eseguiti dall'utente amministratore, chiamato “root” negli ambienti Unix.

Prendiamo ancora la shell bash per mostrare degli esempi. Il prompt standard è cosi composto:

username@hostname:path_completo$

La riga seguente mostra un reale esempio:

john@doe:/etc/init.d#

Partendo da sinistra la prima stringa rappresenta lo username, quindi il carattere chiocciola (@) a cui segue l'hostname, poi i due punti (:) seguiti dal percorso assoluto alla directory corrente ed infine il segno cancelletto (#) che indica che si sta operando come amministratore. A questo segue il cursore, visualizzato in modo lampeggiante o fisso.

Il prompt delle shell puó essere personalizzato inserendo anche altri codici di controllo che permettono di visualizzare la data, l'ora, il numero di job in esecuzione dalla shell, il numero sequenziale del comando digitato, lo UID (*user id*), caratteri specifici, testo colorato ed altro. Si puó addirittura includere interi pezzi di codice che possono mostrare nel prompt qualsiasi cosa, ad esempio la memoria RAM occupata o quella disponibile, il codice di errore d'uscita del comando eseguito, una citazione famosa scelta casualmente, le ultime news, etc.

Ma facciamo qualche esempio. Per modificare il prompt occorre andare a modificare la variabile d'ambiente chiamata “*PS1*”. Digitiamo la seguente stringa nella shell e vediamo l'effetto alla riga immediatamente seguente dopo la pressione del tasto invio:

john@doe:~$ PS1='\e[1;42m \e[1;47m \e[1;41m \e[0;0m \u > '

Questa banale impostazione permette di vedere tre spazi colorati verde, bianco e rosso di seguito, a mò di bandiera italiana, seguiti dal nome dell'utente autenticato.

john >

Vediamo ora un esempio senz'altro più utile:

```
john@doe:~$ PS1='\e[1;33m\t \! [\[\j] \e[1;34m\]\u@\H \e[0;0m> '

22:18:17 525 [0] john@doe >
```

22:18:22 526 [0]john@doe > xeyes &

[1] 6989

22:18:24 527 [1]john@doe >

Qui definiamo un carattere grassetto giallo per mostrare l'ora corrente, il numero progressivo del comando nella bash (*shell history*) e tra parentesi quadrate il numero di job in esecuzione attualmente dalla shell; seguono in grassetto blu lo username e l'hostname separati dal carattere chiocciola, per poi terminare col segno maggiore nel colore standard della shell.

2. *Command completion*

Questa funzione avanzata, non presente nelle originali shell di un tempo, permette di completare il nome di un comando, di una directory o di un file scritto parzialmente, solo con i caratteri iniziali. Per attivare la funzione è sufficiente semplicemente premere il tasto TAB. Nel caso esistano piú occorrenze che iniziano con la stessa sequenza dei caratteri digitati, la seconda pressione consecutiva del tasto TAB li elencherà in forma tabulare. Un esempio vale piú di mille parole:

john@doe:~$ ch

a questo punto premiamo il tasto TAB, la prima volta senza apparente effetto, mentre la seconda invece mostra un elenco che puó variare in funzione delle utilities installate nel proprio sistema, tipo:

chage chcon chkdupexe chsh

chardet cheese chmod chvt

charmap cherokee-tweak charset chfn

chown chattr chgrp chrt

dobbiamo essere piú precisi, pertanto digitiamo anche il carattere “m”:

john@doe:~$ chm

premiamo ancora il tasto TAB e vedremo che il comando “chmod” verrà completato dalla shell, essendo l'unico che inizia con i tre caratteri specificati:

john@doe:~$ chmod

Si lascia al lettore la verifica dell'auto-completamento anche con i nomi dei file o le directory.

3. *Shell history*

Questa è la caratteristica senz'altro più utilizzata insieme alla precedente. La shell mantiene una memoria cache con un certo numero di comandi inseriti, generalmente gli ultimi 500 o 1000, dipende dalle impostazioni di sistema, ovviamente modificabile. Nella bash, ad esempio, tutti i comandi digitatati durante la sessione vengono poi salvati al logout nel file *~/.bash_history*, il cui contenuto viene ricaricato al momento del login.

Questo significa che due shell aperte contemporaneamente non condividono i comandi inseriti in ciascuna di esse da quando sono state aperte. Invece degna di lode è la funzionalità della zsh che permette di condividere lo storico dei comandi tra tutte le shell aperte.

I comandi digitati in precedenza possono essere scorsi per mezzo dei tasti freccia sú e freccia giú o per mezzo di *shortcut*. Il comando interno "*history*" permette di vedere lo storico di tutti i comandi digitati con il numero progressivo a lato. Conoscendo quindi il numero progressivo del comando lo si puó richiamare digitando un punto esclamativo (!) seguito da tale numero, mentre un doppio punto esclamativo richiama l'ultimo comando eseguito, come mostrato nel seguente esempio:

john@doe:~$ history

1 cat /etc/fstab

2 df -h

3 more /proc/cpuinfo

...

126 cal

127 grep head /opt/.sh*

127 crontab -l

128 cat /proc/loadavg

john@doe:~$

john@doe:~$!!

cat /proc/loadavg

0.15 0.07 0.01 2/272 8748

john@doe:~$!126

grep head /opt/.sh*

/opt/wifiautoscan.sh: WIF=`iwconfig 2>/dev/null|grep -u 802.11|head -1|awk '{print $1}'`

Nel caso non si sia certi del numero del comando nello storico, per evitare il rischio di lanciare un comando indesiderato, o peggio dannoso (si pensi ad un comando di cancellazione), si puó appendere al richiamo del numero di comando un segnale, come mostrato in questo esempio:

john@doe:~$!126:p

grep head /opt/.sh*

john@doe:~$!!

grep head /opt/.sh*

/opt/wifiautoscan.sh: WIF=`iwconfig 2>/dev/null| grep -u 802.11 |head -1|awk '{print $1}'`

In tal caso il comando numero 126 viene solamente visualizzato e non lanciato; però, intelligentemente, viene posto comunque come ultimo comando nella history e pertanto, se corretto, lo si può eseguire direttamente col doppio punto esclamativo.

Un altro modo per recuperare una riga di comando che si trova nello storico della bash, ma che non sappiamo a che numero progressivo abbia, è tramite l'uso dello shortcut "*CTRL+*r", la ricerca inversa. Proviamo nella nostra shell e vedremo che alla pressione dello shortcut il prompt cambierà come segue:

(reverse-i-search)`':

Ora iniziamo a digitare i primi caratteri, per esempio "c" ed "a" e vediamo cosa avviene:

(reverse-i-search)`ca ': cat /proc/loadavg

Se questo è il comando a cui siamo interessati è sufficiente a questo punto premere invio per eseguire il comando stesso, altrimenti continuiamo a digitare caratteri fino a che avremo trovato il comando che ci interessa o meno. Ora provate voi stessi a premere ripetutamente la combinazione "*CTRL+r*" senza digitare alcun carattere e divertitevi a vedere cosa avviene.

4. *Word modifiers*

Abbiamo già visto i caratteri ":p" che aggiunti al termine di richiamo del comando lo visualizzano a video, ma non lo eseguono. Si chiama modificatore di parola e ve ne sono altri che ovviamente iniziano sempre con i due punti. Vediamone alcuni:

- :h dall'ultimo argomento del comando prende il percorso rimuovendone il nome del file
- :t rimuove la parte iniziale dell'argomento mantenendone la coda (la parte finale)
- :r mantiene la riga di comando eliminandone la coda a partire dal carattere "punto"
- :e rimuove tutto tranne l'ultimo argomento o dal carattere "punto" fino a fine riga

Vediamo qualche esempio per chiarire questo argomento:

john@doe:/tmp$ tar tvfj /usr/src/linux-3.5.3.tar.bz2

{ OUTPUT ... }

john@doe:/tmp$ cd !$:h

cd /usr/src

john@doe:/usr/src$ file !-2$:t

file linux-3.5.3.tar.bz2

linux-3.5.3.tar.bz2: bzip2 compressed data, block size = 900k

Fermiamoci un attimo per analizzare ciò che avviene con questi comandi.

La prima riga non fa altro che mostrare a video il contenuto del file compresso del kernel Linux.

Il secondo comando invece cambia la directory di lavoro prendendo l'ultimo argomento dal comando precedente, richiamato dalla sequenza "*!$*", e rimuovendone il nome del file. Quindi ciò che rimane è il percorso completo dove intende spostarsi.

Con il terzo comando invece si vuole vedere il tipo del file, ma per brevità, invece che riscriverlo completamente si è preferito richiamare il comando scritto 2 righe prima e prelevarne solamente la coda dell'ultimo argomento usato, ossia il nome del file.

Si lascia al lettore il compito (e il divertimento!) di continuare a sperimentare con i modificatori.

5. *Line editing*

Una interessante caratteristica delle shell è quella di permettere di modificare una linea di comando mediante shortcut, il che torna molto utile quando si hanno linee di comando molto lunghe. Di solito per default si utilizzano gli shortcut tipici di *Emacs*[12], l'editor testuale scritto da Richard M. Stallman, il più rinomato dopo lo storico *Vi*[13], ma si possono utilizzare gli shortcut anche di quest'ultimo.

Si riportano qui di seguito alcuni degli shortcut *emacs-style* più utili:

- *ESC+b* sposta il cursore indietro di una parola
- *ESC+f* sposta il cursore avanti di una parola
- *CTRL+b* sposta il cursore di un carattere indietro (a sinistra)
- *CTRL+f* sposta il cursore di un carattere in avanti (a destra)
- *CTRL+a* sposta il cursore ad inizio riga
- *CTRL+e* sposta il cursore a fine riga
- *CTRL+k* cancella tutti i caratteri dalla posizione del cursore fino a fine riga
- *CTRL+u* cancella tutti i caratteri dalla posizione del cursore fino ad inizio riga
- *CTRL+w* cancella la parola indietro (a sinistra del cursore)
- *CTRL+v* permette di inserire un carattere di controllo (ad esempio l'ESC)

Inserendo nella shell il comando "*set -o vi*" gli si comunica che abbiamo intenzione di usare gli shortcut di questo editor per gli spostamenti e modifiche. Per cambiare modalità e tornare all'Emacs è sufficiente digitare "*set -o emacs*".

6. *Spelling correction*

Questa funzionalità è disponibile in particolare nella shell zsh e in versione sperimentale nella tcsh, mentre nella bash è limitata semplicemente alla correzione dei nomi di directory. Per chi scrive velocemente ed è prono ad errori rappresenta senz'altro una notevole comodità. Si tratta di una funzionalità in cui la shell analizza le parole scritte dall'utente dopo la pressione del tasto invio e se verifica che il comando/file/directory

[12] http://www.gnu.org/software/emacs/

[13] http://it.wikipedia.org/wiki/Vi_(software)

non esiste (quantomeno nel PATH) cerca un potenziale sostituto considerando un piccolo insieme di parole simili, con lettere scambiate o con lettere mancanti. Se ne riporta un esempio eseguito in una shell zsh:

john@doe% cd /tmpp

zsh: correct '/tmpp' to '/tmp' [nyae]? Y

john@doe% pdw

zsh: correct 'pdw' to 'pwd' [nyae]? n

zsh: command not found: pdw

john@doe%

john@doe% pdw

zsh: correct 'pdw' to 'pwd' [nyae]? y

/tmp

La zsh chiede di inserire una lettera tra le quattro mostrate in parentesi quadre e quindi premere invio. Le lettere stanno ad intendere, nell'ordine:

- "no", non si accetta la correzione suggerita dalla zsh e si desidera proseguire col testo inserito
- "yes", applica la modifica suggerita dalla shell ed esegue il comando modificato
- "abort", annulla l'operazione e restituisce il cursore pulendo la linea di comando
- "exit", non si accetta il suggerimento fornito e restituisce il cursore per la modifica del testo inserito

7. Alias

Il comando *alias* permette di definire delle parole chiave a cui assegnare un comando personalizzato, un alias appunto. Tale comando puó essere anche la combinazione o una sequenza di piú comandi.

Come per le variabili d'ambiente, ogni alias definito nella shell mantiene il proprio valore solo fino a che la shell non viene chiusa, pertanto in qualsiasi altra shell quell'alias non esiste. Per mantenerne la definizione in ogni shell che viene lanciata occorre scriverli nel file di startup della shell stessa, ad esempio nel file "*~/.bashrc*" nel caso della bash o nel file "*~/.zshrc*" nel caso della zsh. In tal modo al momento dell'apertura della shell, tra le altre cose, verranno anche definiti gli alias. Qualche esempio:

john@doe:~$ alias grep='grep --color -n'

john@doe:~$ grep $USER /etc/passwd

john:x:1000:1000:john,,,:/home/john:/bin/bash

john@doe:~$ alias dse='dpkg -l |grep'

john@doe:~$ dse zeni

ii zenity 2.30.0-1 Display graphical dialog boxes from shell scripts

5.4 Comandi interni e comandi esterni

La Shell interpreta il testo inserito dall'utente seguendo un ben definito *pattern*, uno schema:

- rileva gli spazi (*blank*) per identificare le i comandi e gli argomenti
- interpreta i caratteri speciali (*wildcard*) per sostituirli con i nomi file/directory appropriati
- ricostruisce la riga di comando rimuovendo i caratteri doppi apici (“), interpretando quelli singoli (') ed eventualmente risolvendo i sotto-comandi in modo ricorsivo (si vedano gli esempi relativi le “*sub-shell*” nel prossimo paragrafo)
- sostituisce le variabili
- esegue i comandi, che se non vengono scritti con un percorso assoluto la shell li andrà a cercare nell'elenco delle directory definite nella variabile di ambiente PATH, nell'ordine ivi specificato

I comandi si possono differenziare in interni ed esterni.

I comandi interni sono comandi che la shell stessa, come programma, mette a disposizione dell'utente per eseguire le funzioni più comuni, come maneggiare file e directory, definire variabili, eseguire cicli, verificare condizioni, gestire processi, intercettare segnali, e molto altro.

I comandi esterni sono invece comandi che sono esterni alla shell e che dipendono dall'installazione di altri pacchetti software e che generalmente vengono installati dal package manager nelle directory eseguibili del sistema, come */bin*, */sbin*, */usr/bin*, etc.

La cosa interessante ed importante è che dalla shell si possono lanciare tutti i comandi disponibili sul sistema, siano essi interni o esterni, sia grafici che non, per cui le potenzialità della shell sono teoricamente infinite. Si consideri oltretutto che la shell mette a disposizione un ambiente di programmazione completo e potente¸ immediatamente utilizzabile sulla ogni riga di comando. Per un approfondimento sulla programmazione shell si rimanda il lettore al paragrafo 2.1.7.

Il comando *man* ci mostra il manuale in linea dei comandi. Di solito ogni comando dispone di un proprio manuale. Si provi a vedere ad esempio il manuale dei comandi già visti in precedenza, come ad esempio “*cat”*, “*file”*, “*tar”* o anche lo stesso “*man*”:

```
john@doe:~$ man man
```

anche se si può comunque ottenere una breve informativa sull'uso dei comandi tramite l'onnipresente e storico parametro “*--help*”.

Invece, per ottenere informazioni relative ai comandi interni occorre usare il comando (interno) *help*:

```
john@doe:~$ help kill
```

Mentre per conoscere tutte le potenzialità della propria shell si veda il relativo manuale, ad esempio per la bash si digiti:

```
john@doe:~$ man bash
```

Si lascia il lettore con una piccola curiosità e compito, quello di investigare sulla differenza tra il comando *"kill"* interno e quello esterno!

5.5 Conclusioni

Per concludere, riprendendo l'iniziale considerazione che oggigiorno molti pensino che l'uso della CLI sia obsoleto e lento, possiamo affermare invece che:

- la CLI è un tool indispensabile per gli amministratori di sistema *anche al giorno d'oggi;*
- molto spesso rappresenta *il modo più rapido e flessibile* per impartire comandi ad un sistema;
- in certi casi è addirittura l'*unico modo* per raggiungere, controllare e gestire sistemi remoti;
- la *versatilità, flessibilità e potenza* delle combinazioni dei tool disponibili tramite la shell non possono essere uguagliate, o lo sono molto difficilmente, da strumenti grafici.

Domande (fac-simile all'esame)

1. Cosa permette di fare il comando "alias"?
    a. Creare un comando personalizzato
    b. Creare un utente che condivide la stessa home di un altro
    c. Creare un file con un nome uguale ad un comando
2. Quale dei seguenti è un comando esterno?
    a. cd
    b. ls
    c. trap
    d. select
    e. shift
3. Quale combinazione di caratteri si indica ad una shell che un file di testo è uno script? (Domanda aperta)
4. Cosa rappresenta la variabile speciale "$#"?
    a. Tutti gli argomenti passati ad uno script

b. Tutti gli argomenti passati ad uno script tranne il primo
c. L'ultimo argomento passato ad uno script
d. Il numero di argomenti passati ad uno script

5. Dove occorre scrivere una variabile affinchè il suo valore venga mantenuto in ogni nuova shell bash che viene aperta?
    a. /etc/profile
    b. ~/.bash_history
    c. ~/.config
    d. ~/..bashrc

Risposte alle domande

1 - A
2 - B
4 - D
5 - D

ithum®

6 Usare la linea di comando per ottenere aiuto

6.1 Come ottenere le prime informazioni: apropos e whatis

Utilizzando un computer è spesso necessario verificare il nome, la sintassi o le opzioni di un comando. Per queste situazioni Linux offre vari sistemi di aiuto: i manuali dei comandi - le famigerate "man pages" - e il più recente sistema "info". La logica di "man" e "info" è molto semplice ma prima di approfondirne il funzionamento è bene famigliarizzare con i comandi "apropos" (che ricerca stringhe di testo tra le voci di aiuto) e whatis (che ricerca un nome comando esatto):

```
$ whatis cp
cp (1)  - copy files and directories
cp (1)  - copy files and directories
cp (1p)  - copy files

$ apropos cp
cp (1)  - copy files and directories
cpan (1)  - easily interact with CPAN from the command line
cpio (1)  - copy files to and from archives
[...]
xdm (1x)  - X Display Manager with support for XDMCP, host chooser
xfs_rtcp (8)  - XFS realtime copy command
xfs_rtcp [] (8)  - XFS realtime copy command
```

Spiegando meglio questi due esempi, apropos restituisce sullo standard output la lista delle pagine di aiuto che menzionano la stringa ricercata (in questo caso "cp"), includendo le voci relative al comando "cp" ma anche alle voci per "cpan", "cpio" e "xdm" (la cui corrispondenza è da rintracciarsi nella descrizione del comando), mentre nel caso di whatis, il risultato a video è più scarno ma preciso, mostrando in output esclusivamente l'esatta corrispondenza fra parametro passato al comando e documentazione disponibile.

Passiamo ora al comando man vero e proprio.

6.2 Il comando man

Man è uno strumento assai potente ma è bene chiarirne l'uso: i Man non sono una raccolta di tutorial e non servono per imparare l'uso dei comandi, sono guide veloci e dettagliate che presuppongono la conoscenza dell'argomento e del tool in discussione. I manuali non sono lo strumento indicato per imparare da zero, per questo sono più

adatti i classici tutorial disponibili sul web. Anche la qualità, nelle pagine dei manuali, è molto variabile: alcune sono ottimamente scritte, altre inaccurate o incomplete. I manuali sono normalmente scritti dagli stessi autori del programma e spesso non ripongono nello scrivere documentazione la stessa attenzione profusa nella programmazione.

Veniamo al funzionamento vero e proprio del comando man.

Il modo più semplice di utilizzare questo potente strumento è digitare il comando man seguito dal comando su cui ci si vuole documentare, sul relativo file di configurazione o altra parola chiave significativa:

```
$ man cp

NAME
cp - copy files and directories

SYNOPSIS
cp [OPTION]... [-T] SOURCE DEST

cp [OPTION]... SOURCE... DIRECTORY

cp [OPTION]... -t DIRECTORY SOURCE...
```

Ognuna delle pagine di manuale ricade sotto una delle 9 categorie riepilogate nella tabella 1.

Tabella 1 categorizzazione delle pagine di man

Sezione	Descrizione
1	Programmi eseguibili e comandi di shell
2	Chiamate di sistema operate dal kernel
3	Chiamate di libreria operate dalle librerie dei programmi
4	File di device (spesso localizzati sotto /dev)
5	Formati dei file
6	Giochi
7	Varie (Macro pacchetti, convenzioni e simili)
8	Comandi di amminitrazione di sistema (eseguibili solo dall'utente root)
9	Routine del kernel

L'utilità di questa tabella si evidenzia in caso di argomenti omonimi. ad esempio per la parola chiave passwd:

```
$           man                 passwd
PASSWD(1)   User Commands       PASSWD(1)

NOME
passwd - cambia la password utente

SINOSSI
passwd [opzioni] [LOGIN]

DESCRIZIONE
Il comando passwd permette di cambiare le password per un account utente.

$           man                         5 passwd
PASSWD(5)   Linux Programmer's Manual   PASSWD(5)

NOME

passwd - file delle password

DESCRIZIONE
```

Passwd è un file di testo che contiene un elenco degli account sul sistema, e per ciascuno di questi riporta alcune informazioni utili come user ID, group ID, home directory, ecc.

Nell'esempio precedente, invocando la pagina di manuale con tanto di richiamo alla categoria 5 (il cinque fra "man" e "passwd" del secondo esempio), è stata richiamata la pagina della categoria "formati dei file" (relativa al file /etc/passwd), anzichè quella più comunemente richiesta della categoria 1, ossia "programmi eseguibili e comandi di shell" (relativa all'eseguibile /bin/passwd). Man stabilisce la priorità delle categorie secondo quanto impostato in /etc/man.conf: un utente alle prime armi non avrà probabilmente bisogno nè di modificare questo file, nè di specificare la categoria al momento dell'invocazione di man, poichè tipicamente avrà bisogno delle informazioni che il sistema man saprà restituire in automatico, ma un rapido accenno alla questione era almeno doveroso.

6.3 Struttura delle pagine di man

La struttura delle pagine di man segue un certo schema convenzionale che include di base diverse sezioni, ognuna delle quali ha una specifica funzione. Quelle più comuni sono:

- Nome: ogni pagina di manuale comincia con la dichiarazione del comando, chiamata o file in trattazione, il tutto seguito da una brevissima descrizione.

```
$ man ls
```

LS(1) User Commands LS(1) NAME
ls - list directory contents

- Sinossi: La sinossi presenta brevemente come il comando va usato. Eventuali parametri opzionali saranno racchiusi fra parentesi quadre, mentre le parentesi tonde indicheranno la possibilità di inserire un set di elementi come argomento al comando
- Descrizione: Una breve descrizione su ciò che l'applicazione, file o file di configurazione fa o non fa. La lunghezza di questa sezione può variare da poche righe a diverse pagine, ma solitamente si attesta alle dimensioni di un tipico sommario.
- Opzioni: Questa sezione estende e spiega le opzioni menzionate nella sinossi. Solitamente ogni opzione si presenta paragrafata singolarmente, ed include una spiegazione più o meno dettagliata delle funzionalità.
- Files: Questa sezione elenca i file associati all'argomento in trattazione: ad esempio nel caso di un file eseguibile, verranno qui menzionati i relativi file di configurazione.
- See also: Questa sezione fornisce eventuali informazioni correlate all'argomento in trattazione disponibili altrove. Ad esempio, la pagina man di fdisk (uno strumento per partizionare i dischi), indica come ulteriori riferimenti le pagine per cfdisk, sfdisk, parted ed altri strumenti utili allo stesso scopo.
- Bugs: La sezione dedicata agli eventuali difetti e limitazioni relative all'argomento in trattazione o nel migliore dei casi dichiara che non sono noti difetti allo stato attuale.
- Storico: Alcuni tool possono includere questa sezione per spiegare l'evoluzione storica dell'argomento in trattazione, come le date di inizio del progetto ed eventuale passaggi fondamentali degni di nota.
- Autore: La sezione con i riferimenti ed i contatti dell'autore del programma o progetto in questione.

Oltre questo schema classico, possono essere presenti altre sezioni, a discrezione dell'autore del programma e/o della relativa pagina di manuale.

6.4 Navigare tra le pagine di man

Tipicamente il sistema man fa uso dello strumento less per scorrere le pagine di documentazione, ossia un impaginatore testuale funzionante via terminale. Le sue funzionalità includono lo spostamento avanti e indietro nel testo, la ricerca e lo spostamento a determinate righe del documento. La tabella 2 riepiloga i comandi principali

Tabella 2 - tasti di navigazione del comando less

Tasto	Azione
h o H	Mostra una guida interna all'utilizzo del comando
Pagina giù, Spazio, Ctrl+V, f, Ctrl+F	Avanza di una schermata nel documento
Pagina su, Esc+V, b, Ctrl+B	Torna indietro di una schermata nel documento
Freccia giù, Invio, Ctrl+N, e, Ctrl+E, j, Ctrl+j	Avanza di una linea nel documento
Freccia su, y, Ctrl+Y, Ctrl+P, k, Ctrl+K	Torna indietro di una linea nel documento
*[n]*g, *[n]*<, *[n]*Esc+<	Va alla linea *[n]* nel documento, esempio: 100g va alla linea numero 100. Se omesso il valore *[n]*, il valore di default è 1
*[n]*G, *[n]*>, *[n]*Esc+>	Va alla linea *[n]* nel documento, esempio: 100G va alla linea numero 100. Se omesso il valore *[n]*, il valore di default è l'ultima linea del documento
*[n]*p, *[n]*%	Va al valore percentuale di *[n]* del documento, esempio: 50p va alla metà del documento
/[pattern]	Ricerca in avanti nel documento per la stringa *[pattern]*
?*[pattern]*	Ricerca a ritroso nel documento per la stringa *[pattern]*
n, /	ripete la ricerca precedente
q, Q, :q, :Q, ZZ	Esce dal paginatore less

Il comando less è invocato automaticamente dal comando man ma può essere utilizzato anche per leggere altri file di testo appendendo al comando less il file da aprire. Ovviamente i comandi di navigazione riepilogati in tabella 2.2B saranno altrettanto validi.

6.5 Un alternativa a man: il comando info

La logica alla base del sistema man risale ad almeno un paio di decadi fa e precede alcuni importanti sviluppi maturati nell'ambito della moderna gestione delle informazioni: un esempio su tutti, le pagine di documentazione del sistema man non sono interconnesse fra loro tramite sistema di iperlinking (ossia quello che ci permette di saltare da una pagina web all'altra cliccando su determinati collegamenti). Di conseguenza, pur potendo ottenere informazioni su argomenti correlati grazie alle segnalazioni spesso riportate nelle apposite sezioni "vedi anche" in calce alle pagine di man (vedi par. 2.2.3), per visualizzare dette pagine è necessario uscire dalla pagina ed invocare un nuovo comando. Il sistema info nasce proprio per supplire a queste carenze supportando nativamente l'iperlinking. Ogni pagina di info è definita NODO e tutto il sistema di info pretende di organizzare i nodi disponibili attraverso un sistema di gestione simile a quello che governa il mondo del web. In virtù di ciò, la documentazione di un singolo argomento può risultare suddivisa in più nodi tutti navigabili e debitamente collegati fra loro.

I nodi sono organizzati in livelli in maniera molto simile ai livelli di un libro, diviso in capitoli e paragrafi: la pagina info di un determinato argomento equivale ad un intero capitolo, mentre i vari nodi rappresentano i paragrafi. A parte queste basilari logiche di organizzazione e gestione dei contenuti, i contenuti sono strutturati in maniera simile alle pagine di man e, come nel caso del precedente strumento, anche le pagine info sono maggiormente indicate per riportare alla memoria determinati meccanismi di funzionamento anzichè per apprendere qualcosa da zero.

La vera ed unica differenza tra il sistema man ed il sistema info risiede nella scelta di utilizzare un sistema o l'altro da parte degli autori. Tipicamente la documentazione dei programmi supportati ufficialmente dalla Free Software Foundation (FSF) usa il sistema info, mentre gli sviluppatori non direttamente legati alla FSF sembrano voler rimanere più fedeli al sistema man, forse per ragioni legate ad una maggiore semplicità nella compilazione di documenti tutto sommato a sè stanti. La conseguenza più ovvia è che può capitare di avere documentazione in un sistema e non in un altro, ma da questo punto di vista info risulta più versaltile poichè è in grado di includere automaticamente le pagine di man per un determinato argomento, quando non trova informazioni nel suo sistema interno.

Lo strumento base per leggere documentazione con info è ovviamente info: la pagina info di un determinato argomento si invoca tramite comando info argomento, ad esempio info ls restituirà:

```
$ info ls

File: coreutils.info,  Node: ls invocation,  Next: dir invocation,  Up: Directory listing
```

Anche per questo tool sono disponibili numerose opzioni da usare per navigare tra le informazioni, riepilogate nella tabella 3:

Tabella 3 - tasti di navigazione del comando info

Tasto	Azione
?	Mostra un aiuto interno
N	In una concatenazione dei nodi a gerarchia unica, avanza al nodo successivo. Questa azione potrebbe risultare utile in caso l'autore avesse predisposto una lettura di nodi sequenziale per un determinato argomento
P	In una concatenazione dei nodi a gerarchia unica, torna al nodo precedente. Come nella situazione precedente, da usare per ripercorrere a ritroso la sequenza dei nodi
U	Sale di un livello lungo la gerarchizzazione dei nodi
Frecce	Muovono il cursore sulla pagina, permettendo di selezionare i collegamenti o di scorrere lo schermo
Pagina su, Pagina giù	Scorrono rispettivamente in alto e in basso i contenuti di un singono nodo
Invio	Apre un nodo quando selezionato. I collegamenti sono evidenziati dalla presenza di un asterisco (*) a sinistra di una parola o frase
L	Visualizza l'ultima pagina di info letta, indipendentemente dal livello gerarchico della pagina aperta
T	Porta alla prima pagina dell'argomento in visualizzazione
Q	Esce dal sistema di paginazione del comando info

In caso si volesse usare un interfaccia grafica, sono disponibili i seguenti strumenti:

- Emacs: editor di testi estremamente potente che include internamente un sistema di consultazione delle pagine info.
- Tkinfo: programma completamente a sè stante con interfaccia punta e clicca per la lettura delle pagine info. Per la consultazione si possono usare sia i comandi da

tastiera nativi di info (riepilogati in tabella 2.2C) che gli strumenti visuali della sua interfaccia[14].

6.6 Come ottenere ulteriore documentazione

Benchè i sistemi man e info siano ricchissimi di informazioni, possono essere necessari documenti di tipo differente: how-to, esempi pratici, usi frequenti o guide per iniziare; Per ovviare a questi limiti proveremo ad estendere la nostra trattazione fornendo informazioni su come rintracciare ulteriore documentazione.

Documentazione disponibile sul computer

Molti programmi per Linux vengono forniti con documentazione supplementare oltre alle pagine per man e/o info. In alcuni casi sono addirittura disponibili interi pacchetti aggiuntivi di sola documentazione (come nel caso del pacchetto di documentazione extra "samba-doc" disponibile su debian e derivate, relativo al software samba), ma più tipicamente una prima sorgente di informazioni aggiuntive può essere rappresentata da un file README (readme.txt), in forma di semplice testo riportante informazioni dall'utilità estremamente variabile.

Altre importanti nozioni possono essere apprese dai file di configurazione dei programmi (solitamente localizzati in qualche sottocartella della directory /etc), spesso ricchi di commenti e spiegazioni o scandagliando la directory /usr/share/doc con il comando find filtrando i risultati per parola chiave, come nel seguente esempio in cui si ricerca tutta la documentazione pertinente al termine "zip":

```
$ find /usr/share/doc/ -name "*zip*"

/usr/share/doc/zip-3.0
/usr/share/doc/zip-3.0/ziplimit.txt.bz2
/usr/share/doc/gzip-1.5
/usr/share/doc/gzip-1.5/txt/gzip.doc.bz2
/usr/share/doc/unzip-6.0-r1
/usr/share/doc/bzip2-1.0.6-r3
/usr/share/doc/bzip2-1.0.6-r3/bzip2.txt.bz2
```

Tra gli altri comandi vale anche la pena di citare "locate" che ricerca in un database autonomamente mantenuto tra i comandi disponibili sul sistema, restituendo l'elenco di risultati relativi al termine specificato:

```
$  locate more

/bin/more
/usr/bin/bzmore
/usr/bin/lzmore
```

[14] Sia Emacs che Tkinfo non sono in grado di integrare le pagine man come il tool info.

/usr/bin/xzmore
[...]
/usr/share/man/man1p/more.1p.gz

e "whereis" che restituisce il percorso di elementi relativi al termine ricercato (incluse eventuali pagine man)

$ whereis ls

ls: /bin/ls /usr/bin/ls /usr/X11R6/bin/ls /usr/bin/X11/ls /usr/man/man1/ls.1.bz2 /usr/man/man1p/ls.1p.bz2 /usr/share/man/man1/ls.1.bz2

/usr/share/man/man1p/ls.1p.bz2

Come già accennato sopra, tutta la documentazione supplementare rintracciata, sarà poi leggibile con strumenti gli strumenti di seguito riportati a seconda dell'estensione.

Tabella 4 - ***Gli strumenti utilizzabili per leggere documentazione aggiuntiva***

Estensione del file	Descrizione	Applicazione utilizzabile
da .1 a .9	pagine di documentazione man	man, info, less
.gz, .bz2, .tar, .tgz o .zip	File compresso con gzip, tar o bzip2	gunzip o bunzip2 per decomprimere l'archivio; oppure usare direttamente il comando less che dovrebbe essere in grado di decomprimere il file e leggerlo contestualmente
.txt	Testo semplice	less o qualunque altro editor di testo
.html o .htm	Pagina ipertestuale (HTML)	Un qualunque browser
.odt	Testo OpenDocument	OpenOffice.org, LibreOffice o altri word processor
.pdf	Portable document format (PDF)	Adobe Reader, xpdf
.tif, .png o .jpg	Formato grafico	The GIMP, Eye of GNOME (eog)

6.6.1 Aiuto in linea dai programmi

Pur non essendo obbligatorio la maggior parte dei programmi offrono al loro interno piccoli sistemi di aiuto. Aggiungendo l'opzione --help ai comandi viene, solitamente, stampato a schermo l'elenco delle opzioni o un rapido aiuto:

```
$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                  do not ignore entries starting with .
-A, --almost-all           do not list implied . and ..
[....]
```

L'aiuto offerto è solitamente meno dettagliato di quello offerto nelle pagine man o info, in conpenso questa opzione è solitamente disponibile anche nei comandi privi di manualistica.

6.6.2 Documentazione disponibile online

Anche su internet è possibile trovare una vastissima raccolta di documentazione: oltre ovviamente ai siti di riferimento per i vari progetti, uno dei progetti più famosi è il Linux Documentation Project[15], ricchissimo di tutorial e guide sugli argomenti più disparati. E' da noi molto noto anche un progetto omologo in lingua italiana[16], ma che alla data del presente documento (agosto 2012) non sembra essere più disponibile.

Esercizi

1. Quale comando e` corretto per accedere alla "man page" del comando "ls"?

A. man ls B. manual ls C. book ls D. page ls

2. Sono alla ricerca di informazione per copiare un file, quale di questi comandi può essere utile alla ricerca?

A. man copy B. info copy C. apropos copy D. whatis copy

3. Cosa devo digitare per ricevere aiuto in una qualunque pagina info?

A. ? B. help C. man D.info

4. Quali comandi posso utilizzare per richiedere il manuale del comando info

A. apropos B. info info C. info D. man info

5. Quali comandi sono corretti per visualizzare le opzioni del comando ls

A. ls –help B. whatis ls C. info ls D. apropos ls

[15] http://www.tldp.org

[16] http://www.pluto.it

Risposte:

1. A
2. C
3. A
4. B, D
5. A, C

ithum®

7 Usare le directory ed elencare i file

L'accesso ai file è sicuramente fra le operazioni più frequenti in un computer. Poichè i dati del computer possono essere memorizzati su diversi supporti fisici secondo logiche e filesystem differenti, il compito del kernel e della shell è quello di rendere questo processo umanamente comprensibile ed omogeneo indipendentemente da dove e come siano memorizzati i dati.

7.1 Glossario

- File system - Il file system é la logica con cui il sistema memorizza ed accede ai dati all'interno di uno storage.
- Storage - Come storage si intendono tutti i sistemi ed infrastrutture di memorizzazione di dati non volatili come hard disk, cdrom, nas, ecc.

7.2 Nomi file

I file all'interno del filesystem vengono identificati da un nome e dalla propria posizione all'interno della struttura file detta "full path". Il full path deve essere univoco e non possono esistere omonimi.

- Nome file: Doc2
- Full path: /home/user1/Documenti/Doc2

I nomi file possono essere composti da una qualunque sequenza di caratteri identificati dal sistema con esclusione dello slash '/' e dello 'zero byte' (il carattere della tabella asci con valore binario 0), per gli altri caratteri speciali, a cui la shell attribuisce significati particolari, è sufficiente anteporre il carattere di escape backslash '\' per evitare fraintendimenti da parte della shell stessa.

Tabella 5 - Elenco dei principali caratteri speciali

Carattere	Escape	Motivo
	\	Gli spazi vengono usati per separare i parametri
*	*	Vedi wildcards
?	\?	Vedi wildcards
'	\'	Gli apici singoli vengono usati per scrivere parametri complessi

"	\"	Gli apici doppi vengono usati per scrivere parametri complessi
$	\$	Il prefisso $ indica una variabile

Un altro sistema per scrivere nomi file complessi è includere il nome tra apici singoli o doppi.

Esempi:

```
$ ls Ho\ un\ nome\ \"complesso\"\?
$ ls 'Ho un nome "complesso"?'
$ ls "Ho un nome \"complesso\"?"
```

Il comando ls serve per elencare i file e verrà descritto a breve

Nei nomi file è possibile utilizzare anche caratteri non presenti nella tabella asci standard come lettere accentate, dieresi o caratteri di alfabeti non latini; per linux è indifferente, ma i nomi potrebbero non essere correttamente visualizzati da terminali con codifica differente dalla vostra.

Una confusione comune per i principianti è sicuramente che linux distingue i caratteri maiuscoli dai minuscoli quindi, a differenza di Windows, './Italia' è differente da './italia'.

In linux non è predefinita una lunghezza massima dei nomi file ma varia in base al filesystem in uso, solitamente è piuttosto lunga e a partire da ext2 i nomi possono arrivare fino a 255 caratteri.

Una ulteriore differenza con i sistemi dos/windows è l'assenza delle estensioni ossia del suffisso che indica al sistema la tipologia del file. L'estensione in linux non è necessaria in quanto per controllare il tipo verifica i bite iniziali che compongono il file. Per verificare la tipologia di un file indipendentemente dall'estensione si può utilizzare il comando file [filename].

Esempi:

```
$ file Documenti
Documenti: directory
$ file Documenti/doc3.pdf
Documenti/doc3.pdf: PDF document, version 1.5
```

Le estensioni rimangono comunque comode e sono utilizzate da molte persone e programmi per riconoscere ad occhio il tipo di file, oltre ad essere indispensabili per scambiare file con sistemi windows.

7.3 Directory

Le directory, anche chiamate anche cartelle o folder, sono un concetto fondamentale nei sistemi Linux. Tecnicamente una directory è un file che, invece di contenere dati, indicizza riferimenti ai file in essa contenuti.

Le directory, come i file, vengono identificate da un nome e da una posizione e con i file condivide gli stessi limiti per nome ed unicità.

- Nome directory: Documenti
- Full path: /home/user1/Documenti

Linux ha un approccio gerarchico al filesystem, il livello più alto del filesystem è la directory / detta root, in italiano radice. A differenza dei sistemi windows non esistono lettere che identifichino gli storage, la root è unica indipendentemente dal numero di supporti collegati alla macchina (in linux le periferiche possono essere 'attaccate' a qualunque directory del sistema).

Ogni directory può contenere file e directory, le directory contenute in una directory vengono dette "subdirectory", la directory in cui è contenuta una directory è detta "directory superiore" o "parent directory".

Esempi:

'/home' è una subdirectory di '/' ed è directory superiore di '/home/user1';
'/home/user1' è una subdirectory di '/home';
'/' è l'unica directory priva di parent directory.

7.4 Home directory

Al nostro accesso al computer il sistema ci assegna una posizione all'interno del filesystem detta 'home'. La 'home' viene indicata dall'amministratore al momento della creazione dell'utente e solitamente è posizionata all'interno della directory /home ed è omonima all'utente.

Esempio per l'utente 'user1':

```
/home/user1
```

All'interno della home vengono memorizzate tutte le informazioni personali dell'utente come le configurazioni dei programmi o i file personali.

Un'altro modo per indicare la propria home è il simbolo '~' detto 'tilde'. Questo simbolo, ottenibile nelle tastiere italiane con i tasti 'ALT-GR + ì', viene sostituito automaticamente dalla shell con la full path della propria home directory,

Esempi per un utente 'user1':

```
$ pwd
/
$ cd ~
```

```
$ pwd
/home/user1
$ cd ~/Documenti
$ pwd
/home/user1/documenti
```

Questo alias è fondamentale per script e programmi in quanto, indicando sempre la home dell'utente corrente, rende più facile l'accesso ai relativi file e configurazioni senza il bisogno di doverne specificare il nome.

7.5 Working directory

La posizione corrente all'interno del filesystem al momento dell'accesso è chiamata 'working directory': solitamente corrisponde alla home dir, ma potrà ovviamente essere cambiata a piacimento ed in base alle proprie esigenze. Per visualizzare la propria 'working directory' si usa il comando 'pwd' acronimo di 'print working directory'.

Esempio:

```
$ pwd
/home/user1
```

Per cambiare la 'working directory' si usa il comando 'cd', acronimo di 'change directory'

Sintassi:

```
$ cd [ -L | -D ] [directory]
```

Se non viene specificata alcuna directory di destinazione, il comando riporta l'utente nella home dir. Se viene indicato il simbolo '-' al posto della directory, il comando riconduce l'utente alla directory precedentemente occupata.

Esempio:

```
$ pwd
/home/user1/Immagini
$ cd /home/user1/Documenti
$ pwd
/home/user1/Documenti
$ cd -
$ pwd
/home/user1/Immagini
$ cd
$ pwd
/home/user1
```

7.6 Path assoluta e path relativa

Una volta che sia nota la working directory, saà possibile dare indicazioni al sistema in forma assoluta (ossia riferite alla path partendo dalla root) o relative (ossia riferite alla propria posizione corrente). In altre parole, con una path relativa si esprime un percorso che prosegue la working directory.

Esempio: per andare dalla directory /home/user1/Immagini a /home/user1/Documenti si possono usare molti percorsi diversi:

```
$cd /home/user1/Documenti
$cd ../Documenti
$cd /home/user1/Immagini/../documenti
```

Nel primo caso è stata usata una path assoluta, nel secondo caso è stata usata una path relativa, nel terzo caso è rappresentato come il sistema interpreta un path relativa.

Riconoscere il tipo di path è molto facile: quando inizia con lo slash è assoluta, tutte le altre sono path relative.

Per verificare la correttezza di una path relativa basta provare ad aggiungerla alla propria working directory e leggetela completa.

7.7 File e directory nascoste

Esistono directory in cui normalmente non vogliamo o dobbiamo accedere o che comunque non ci interessa vedere nel nostro normale lavoro come file e directory di configurazioni, archivi e cache dei programmi ecc. per non visualizzare questi oggetti si usa creare file e cartelle nascoste.

Per rendere un file o una cartella nascosta è sufficiente anteporre un punto al nome del file o della cartella.

Esempi:

```
~/.config/
~/.bashrc
```

La directory ~/.config è la directory solitamente utilizzata per i file di configurazione. Il file ~/.bashrc contiene le istruzioni che la shell esegue al login.

Attenzione!

Le directory speciali . e .. sono due directory nascoste perciò, pur essendo sempre utilizzabili, non saranno normalmente visibili.

Attenzione!

Nascondere un file non è un metodo raccomandabile per proteggere dati o informazioni sensibili! L'unico scopo previsto per questa funzionalità è semplificare gli elenchi di file durante le ordinarie operazioni di lavoro: per proteggere al meglio i propri dati da

eventuali accessi indesiderati, sarà ben più produttivo fare riferimento all'implementazione di tecniche di cifratura (argomento affrontabile in questa sede).

7.8 Elencare file

Per visualizzare i file contenuti in una o più posizioni del sistema si usa il comando 'ls'.

Sintassi:

ls [OPZIONI] [FILES]

Il comando ls permettere di visualizzare i file conformi ad un determinato pattern in una o più posizioni all'interno del sistema. Ls è un comando inaspettatamente potente e duttile che vi accompagnerà per tutta la vostra vita con linux, in questo capitolo ci limiteremo agli usi più semplici ma siete invitati a leggervi tutta la man page.

Il comando ls, se privo di altre opzioni, elenca tutti il contenuto visibile della directory corrente e, in assenza di ordini diversi, ordina il risultato per nome file.

```
$ ls
Documenti
Download
file1.txt
Immagini
LPI
```

Nelle moderne shell, ls si occupa automaticamente di colorare alcune tipologie di file per renderle facilmente riconoscibili:

Tabella 6 - Tipi di colori per ls

Tipologia	Colore	Suffisso
File di testo	Nero	Nessuno
Eseguibili	Verde	*
Directory	Blu	/
Link	Ciano	@

Aggiungendo l'opzione -l si ottiene una lista più dettagliata:

```
drwxr-xr-x 2 user1 user1 4096 31 lug 16.52 Documenti
drwxr-xr-x 2 user1 user1 4096  8 giu 12.07 download
drwxr-xr-x 2 user1 user1 4096 29 lug 09.21 Documenti
drwxr-xr-x 2 user1 user1 4096 29 lug 09.21 Immagini
```

```
drwxr-xr-x 3 user1 user1 4096 29 lug 09.21 LPI
-rw-r--r--  1 user1 user1 42960 19 giu 08.29 file1.txt
drwxr-xr-x  4 user1 user1 4096 22 ago 10.21 Games
```

La prima colonna indica i permessi di accesso (verranno affrontati nel capitolo 5.3), la seconda colonna indica la proprietà del file (Sempre affrontato nel capitolo 5.3), quindi dimensione, ultima modifica ed infine il nome.

Tabella 7 - Opzioni comuni per ls

Opzione	Sinificato
-a --all	Visualizza anche i file nascosti
-l --format=long	Mostra maggiori informazioni
-o --no-color	Non mostrare i colori
-p -F	Aggiungi il suffisso per riconoscere la tipologia
-r --reverse	Inverti ordine
-R --recursive	Elenca anche le subdirectory
-S --sort=size	Ordina il risultato per dimensione
-t --sort=time	Ordina il risultato per data
-X --sort=extension	Ordina il risultato per estensione ("tipo file")

E' anche possibile passare al comando ls una o più posizioni da elencare:

```
$ ls  /home ~
/home :
user1
user2
user3
user4

/home/user1 :
Documenti
Download
file1.txt
Immagini
LPI
```

Attivare l'opzione della ricorsività equivale a richiedere l'elenco di tutte le subdirectory contenute nella path indicata:

```
$ ls -R ~

$ ls ~ ~/Documenti ~/Documenti/lavoro ~/Download ~/Immagini .....
```

I due comandi tecnicamente si equivalgono ma l'opzione -R è sicuramente più comoda e non richiede la conoscenza preventiva di tutte le subdirectory presenti nella cartella interessata.

E' possibile selezionare anche soltanto alcuni file o cartelle indicando il pattern di corrispondenza:

```
$ ls -l file1.txt
-rw-r--r--  1 user1 user1   42960 19 giu 08.29 file1.txt
```

7.9 Wildcards e Pathname Expansion

Ogni volta che si esprime una path è possibile inserire alcuni codici che la shell sostituirà in tutte le possibili corrispondenze. Detti codici si definiscono wildcards (ossia caratteri jolly in italiano).

Utilizzando le wildcards, invece di passare un unico parametro, richiediamo al sistema di considerare validi tutti i parametri corrispondenti ad un determinato pattern.

Esempio:

```
$ ls ~/Do*
/home/user1/Documenti:
doc1.txt
doc2
doc3.odf
lavoro

/home/user1/Download:
file.txt
```

In sintesi è come se avessimo inviato il comando con le due opzioni corrispondenti:

```
ls ~/Documenti ~/Download.
```

E' possibile utilizzare le wildcards in tutte le parti della path anzichè solo al termine:

```
$ ls ~/{Documenti,Download}/*.txt
/home/user1/Documenti/doc1.txt
/home/user1/Download/file.txt
```

La risultante di questo comando è

```
ls /home/user1/Documenti/doc1.txt

/home/user1/Download/file.txt
```

Tabella 8 - Pathname Expansion

Wildcards	Corrispondenza
*	Qualunque serie di carattere
?	Un carattere qualunque
[abcde]	Un carattere incluso nella lista
[a-e]	Un carattere incluso nel range (a,b,c,d,e)
[!abcde]	Un qualunque carattere non incluso nella lista
[!a-e]	Un qualunque carattere non incluso nel range (a,b,c,d,e)
{debian,ubuntu}	Esattamente una delle parole incluse nella lista

Utilizzare bene le wildcards richiede un pò di pratica ma, una volta compreso il meccanismo, possono risultare fondamentali in miriadi di occasioni.

Domande

1. Che comando bisogna usare per verificare la current directory?

A. Pwd B. ls C. rmdir D. ps

2. Quali tra questi comandi riporta nella home directory?

A. cd B. cd - C. cd ~ D. cd /home

3. Indicare tra le seguenti opzioni quali sono path assolute

A. /home/user1 B. ~ C. /var/lib/../../tmp

D. ./Documenti/lavoro

4. Indicare tra le seguenti opzioni quali sono path relative

A. / B. - C. ./ D. ../Immagini

5. Indicare quali tra questi comandi sposta l'utente dalla directory /home/user1/Documenti/lavoro a /home/user1/Immagini

A. cd /home/user1/Immagini B. cd ~/immagini

C. cd ../../Immagini D. cd -

6. Indicare i comandi corretti per elencare dettagliatamente il contenuto della propria home directory

A. ls -A ~ B. ls -l C. ls -l ~ D. ls -t

7. Indicare i comandi corretti per elencare il contenuto della directory /home e delle sue subdirectory

A. ls -R ~ B. ls –r C. ls -a /home D. ls -R /home

ISBN: 9788866850090

7. Elencare il contenuto della directory Documenti all'interno della propria home directory

A. ls Documenti B. ls -R Documenti

C. ls ~ Documenti D. ls ~/Documenti

7.9.1 Risposte

1. Il comando per visualizzare la working directory è pwd.
2. Il comando cd senza opzioni riporta nelle home directory e la tilde (~) passata come opzione è un alias della home directory. La directory /home contiene le home directory ma non è una home directory, il carattere - indica di tornare alla directory di provenienza non alla home.
3. La prima e la terza sono evidentemente due path assolute in quanto partono dalla root /, la tilde è un alias della propria home quindi /home/NOMEUTENTE quindi è una path assoluta.
4. La terza e la quarta sono path relative, La prima è la root quindi assoluta la seconda (-) da sola non è non è una path.
5. La prima è la path assoluta della destinazione quindi corretta, la seconda parte dalla home directory mentre non è specificato che si è l'utente user1, la terza è la path relativa corretta mentre la quarta riporta ad una sconosciuta directory d'origine.
6. La prima risposta mostra il contenuto della home directory ma l'opzione -A mostra i file nascosti non i dettagli, la seconda mostra la lista dettagliata della directory corrente non della home, la terza è corretta ed infine la quarta mostra la directory corrente ordinata per data.
7. La prima elenca ricorsivamente la propria home directory non la directory /home, la seconda fa un elenco inverso della directory corrente la terza mostra i file nascosti nella directory /home ed infine la quarta è corretta.
8. La prima risposta sarebbe corretta se noi fossimo nella home directory ma non è indicato, la seconda è identica alla prima ma aggiunge l'opzione per la recursione, la terza elenca il contenuto della propria home directory e della directory Documenti nella current directory, la quarta è corretta.

8 Creare, copiare, spostare e cancellare file

8.1 Creare file e directory

Abbiamo appreso come muoversi (cd) e come elencare (ls) il contenuto del nostro file computer. Il passo successivo è imparare a lavorare con file e directory.

Già al primo login, ogni utente troverà varie directory nella propria 'home'. Queste directory variano in base alla distribuzione e al window manager ma, probabilmente, esisterà una directory 'Scrivania' o 'Desktop', 'Documenti', 'Musica', 'Modelli', 'Scaricati' oltre a file e directory nascosti che imposteranno inizialmente l'ambiente, ovviamente è tutto modificabile a patto di esserne i proprietari.

8.1.1 mkdir

E' uso dividere i propri documenti e possiamo farlo facilmente creando nuove directory. Per creare una directory si usa il comando 'mkdir' contrazione di 'make directory ' in italiano crea directory.

Il comando 'mkdir' può creare, con una sola istruzione, una o più directory incluse le necessarie parent directory.

Sinossi:

mkdir [Opzioni] DIRECTORY

Tabella 9 - Opzioni comuni per touch

Opzione	Sinificato
-m, --mode=MODE	Specifica i diritti di accesso alla directory
-p, --parents	Crea le parent directory se necessario
-v, --verbose	Fornisce informazioni aggiuntive sulle directory create
--help	Elenca le opzioni disponibili per mkdir

Per creare le directory 'casa', 'lavoro' e 'studio' all'interno della directory documenti, sarà sufficiente scrivere scrivere il comando seguito dalle path delle directory da creare:

```
$ pwd
/home/user1
$ mkdir Documenti/casa
```

```
$ mkdir  Documenti/lavoro
$ mkdir /home/user1/Documenti/studio
```

Con il comando 'pwd' abbiamo verificato la nostra posizione all'interno del filesyste, questa pratica è essenziale per utilizzare path relative invece che assolute (vedi capitolo precedente). Con il secondo e terzo comando abbiamo creato le directory 'casa' e 'lavoro'. Con il quarto ed ultimo comando abbiamo creato la directory 'studio' utilizzando la sua path assoluta invece della path relativa. Possiamo verificarne il risultato con il comando 'ls'

```
$ ls Documenti
casa
lavoro
studio
```

Per velocizzare l'operazione è possibile passare tutti i parametri all'interno dello stesso comando:

```
$ pwd
/home/user1
$ mkdir Documenti/casa Documenti/lavoro Documenti/studio
```

oppure

```
$ cd Documenti
$ mkdir casa lavoro studio
```

In entrambi i casi otterremo lo stesso risultato.

Proviamo ora a creare la directory LPI/test all'interno della nostra home:

```
$pwd
/home/user1
$ mkdir LPI/test
mkdir: impossibile creare la directory "LPI/test": File o directory non esistente
```

Mkdi ci risponderà errore in quanto la directory '~/LPI ' non esiste ed il comando, senza ulteriori opzioni, cercherà di creare soltanto la directory 'test'. Abbiamo due possibili soluzioni:

```
$ mkdir LPI
$ mkdir LPI/test
```

Oppure utilizzando l'opzione '-p' il comando 'mkdir' creerà tutte le parent directory necessarie:

```
$ mkdir -p LPI/test
```

In qualunque momento è possibile richiedere un aiuto da parte di mkdir utilizzando l'opzione '--help' oppure aprendo la sua pagina man.

8.1.2 touch

La creazione di file può avvenire in vari modi: editor di testo, videoscritture, ecc. Ma spesso è sufficiente creare dei file vuoti possono da riempire successivamente. Con il comando 'touch' possiamo creare nuovi file o modificare le date di accesso/modifica a file già esistenti.

Sinossi:

touch [OPZIONI]... FILE...

Tabella 10 - Opzioni comuni per touch

Opzione	Sinificato
-a	Modifica soltanto la data di accesso
-c --no-create	Non crea nessun file
-d STRINGA --date=STRINGA	Utilizza la data indicata al posto del tempo corrente
-m	Modifica soltanto la data di modifica
-r --reference=file	Utilizza il contenuto del file al posto del tempo corrente
-t STAMP	Usa [[CC]YY]MMDDhhmm[.ss] al posto del tempo corrente
--help	

Nota: La differenza tra -t e -d sta nel formato di data/ora utilizzato

Usare touch è molto semplice basta aggiungere al comando i nomi e posizione dei file da creare:

```
$ pwd
/home/user1
$ touch Documenti/documentoVuoto
$ ls Documenti
casa
documentoVuoto
lavoro
studio
```

Come per tanti altri comandi anche con touch è possibile passare più di un parametro per volta e creando più di un file con una sola istruzione.

```
$ touch Documenti/lavoro/doc1 Documenti/studio/doc2
```

Attenzione!

Come detto in precedenza, in linux, l'estensione non ha nessun valore e tutti i file creati con touch, anche se con estensioni specifiche, saranno file vuoti quindi aspecifici. Possiamo verificarlo facilmente con il comando file (vedi capitolo precedente per approfondimenti):

```
$ touch Documenti/doc.txt
$ touch Documenti/doc.pdf
$ file Documenti/doc.txt
Documenti/doc.txt: empty
Documenti/doc.pdf: empty
```

Nel caso che touch incontri un file già esistente si limiterà ad aggiornare la sue data di accesso e modifica.

8.2 Copiare e spostare file e directory

8.2.1 cp

Tutti siamo abituati a copiare i file trascinandoli nel nostro file manager. Copiarli tramite console non è molto più difficile e con un minimo di conoscenza si dimostra molto più flessibile e potente.

Per copiare file o directory si usa il comando 'cp' contrazione del termine inglese copy. Il comando cp si occupa di duplicare uno o più file e directory in altra posizione, cambiandone, se necessario, nome e attributi, è possibile fare backup, creare link invece di replicare il file oltre a molto altro.

Sinossi:

```
cp [OZIONI] SORGENTE DESTINAZIONE
```

Tabella 11 - Opzioni comuni per cp

Opzione	Sinificato
-b	Esegue un backup se il file di destinazione esiste
-i, --interactive	Chide conferma prima di sovrascrivere
-p	Mantiene i permessi e gli orari di creazione/accesso/modifica al file
-r, -R, --recursive	Copia tutte le subdirectory ricorsivamente
-s, --symbolic-link	Crea link simbolici al posto di

	copiare i file
-u, --update	Copia soltanto se il file di destinazione non esiste o se è più vecchio di quello di origine
-v, --verbose	Fornisce informazioni dettagliate delle copie eseguite
-x, --one-file-system	Esegue la copia unicamente nello stesso file system

Nella sua forma più semplice 'cp SORGENTE DESTINAZIONE' il comando copierà il file di sorgente nella destinazione:

```
$ cd Dcoumenti
cp doc.txt doc2.txt
$ ls
casa
doc.txt
doc.pdf
doc2.txt
...
```

In questo caso è stato copiato il file doc.txt in doc2.txt

```
$ cp doc.txt casa
$ ls casa
doc.txt
```

Se cp riconosce un directory come destinazione copierà i file indicati all'interno della directory indicata senza alterarne il nome. per alterarne il nome durante la copia basta aggiungere il nome file alla directory:

```
$ cp doc.txt casa/doc3.txt
```

Attenzione!

nel caso cp incontrasse un file esistente come destinazione lo sovrascriverà irrimediabilmente

```
$ cp doc.pdf casa/doc3.txt
```

per evitare questo comportamento bisognerà aggiungere l'opzione -i alla copia

```
$ cp -i doc.pdf casa/doc3.txt
cp: sovrascrivere casa/doc3.txt?
```

cp autonomamente non copierà directory ma soltanto file. Per copiare ricorsivamente file e directory si deve utilizzare l'opzione '-R ':

```
$ cp Documenti Documenti2
cp: directory "Documenti" omessa
$ cp -R Documenti Documenti2
$ ls
Desktop
Documenti
Document2
.....
```

La potenza del comando cp aumenta notevolmente grazie alle wildcards e pathname expansion (vedi capitolo precedente per approfondimenti). Grazie alle wildcards è possibile selezionare molte sorgenti in contemporanea o scremarle in base alla composizione del loro nome:

```
$ mkdit test
$ cp Documenti/**/*.txt test
$ ls test
doc.txt
doc.pdf
doc2.txt
doc3
doc4.pdf
```

Con un solo comando è possibile copiare tutti i file che terminano con '.txt' (con *si indica una qualunque stringa) presenti nella directory documenti o in qualunque subdirectory (con ** si identifica la directory corrente o qualunque subdirectory). Per rivedere il pathname expansion potete rivedere il capitolo 2.2.

8.2.2 mv

Il comando 'mv', contrazione del termine inglese move, pur avendo molte meno opzioni del comando 'cp' ha una sintassi pressoché identica.

Sinossi:

```
mv [OPZIONI] SORGENTE DESTINAZIONE
```

Tabella 12 - Opzioni comuni per mv

Opzione	Sinificato
-b	Esegue un backuo se il file di destinazione esiste
-f, --force	Non chiede prima di sovrascrivere
-i, --interactive	Chiede prima di sovrascrivere
-u, --update	Muove soltanto se il file di destinazione non esiste o è più vecchio

-v, --verbose	Fornisce informazioni su ogni file mosso
--help	Elenca le opzioni disponibili per mv

```
$ mv doc.txt nuovoDoc.txt
$ mv nuovoDoc.txt casa
$ ls casa
nuovoDoc.txt
```

Dando come destinazione un nome file il file d'origine viene rinominato durante lo spostamento altrimenti viene spostato nella nuova path senza alterarne il nome.

Una differenza importante tra cp e mv risiede nel fatto che mv tratta indifferentemente file e directory quindi non esiste l'opzione per la ricorsività e muovendo una directory verrà spostato sempre anche tutto il suo contenuto.

```
$ mv Documenti2 Documenti3
$ ls
Documenti
Documenti3
[.......]
```

Come per cp, mv sovrascrive sovrascrive i file di destinazione e le opzioni --interactive, --force e --update sono essenziali per avere o meno conferme prima di sovrascrivere altri dati potenzialmente importanti.

Anche mv accetta le wildecards per muovere pattern complessi di file

8.3 Eliminare file e directory

Dopo aver tanto creato bisogna imparare anche a cancellar. In linux esistono due comandi principale per questa operazione: rmdir perle directory e rm per file e directory.

8.3.1 rmdir

Il comando 'rmdir' come già accennato si occupa di eliminare unicamente le directory. La rimozione tramite 'rmdir' può avvenire solamente se la directory è vuota, quindi va svuotata di tutti i file prima di cancellarla.

Sinossi:

```
rmdir [OPZIONI] DIRECTORY
```

Tabella 13 - Opzioni comuni per rmdir

Opzione	Sinificato

-p, --parents	Elimina tutte le directory indicate nella path, esempio: 'rmdir a/b/c' equivale a 'rmdir a/b/c a/b a'
-v, --verbose	Fornisce informazioni sulle directory eliminate
--help	Elenca le opzioni disponibili per mkdir

```
$ rmdir Documenti/lavoro
rmdir: rimozione di "Documenti/lavoro" non risuscita la directory non è vuota
$ mv -i Documenti/Lavoro/* Documenti
$ rmdir Documenti/Lavoro
```

Con il primo comando tentiamo di eliminare una directory piena ed il sistema ci restituisce un errore. Con il secondo comando muoviamo tutto il contenuto della directory Documenti/lavoro in Documenti. Con il terzo ed ultimo comando eliminiamo con successo la directory Documenti/Lavoro.

Come per altri comandi è possibile passare più di una path per volta ed inserire wildecard per pattern complessi, rmdir proverà ad eliminare tutte le path fornite restituendo gli errori, se presenti, per ogni opzione fornita.

```
$ rmdir */lavoro test?
rmdir: la directory test non risulta vuota
rmdir: la directory test2 non risulta vuota
```

Il comando 'rmdir' risulta ragionevolmente sicuro in quanto non procede se la directory contiene file o altre directory. Questa sicurezza di 'rmdir' è anche il suo limite: nel caso si voglia eliminare file o directory con contenuti bisogna utilizzare il comando 'rm'

8.3.2 rm

Il comando 'rm' può rimuovere file e ricorsivamente directory. la sintassi è identica a mv ma ovviamente è possibile passargli sia file che directory.

Sinossi:

```
rm [OPZIONI] FILE
```

Tabella 14 - Opzioni comuni per rm

Opzione	Sinificato
-f, --force	Ignora file ed argomenti senza corrispondenza
-i	Chiede conferma prima di ogni eliminazione
-r, -R	Elimina ricorsivamente directory e i file in esse contenuti
-v, --verbose	Fornisce informazioni per ogni file eliminato
--help	Elenca le opzioni disponibili per rm

```
$ rm Documenti/doc1.txt Documenti/lavoro
rm: impossibile rimuovere "Documenti/lavoro": È una directory
$ rm -R Documenti/lavoro
```

Nel primo comando viene chiesto di eliminare un file ed una directory, 'rm' elimina correttamente il file ma da errore sulla directory. Con il secondo comando aggiungiamo l'opzione -R (recursive), rm si occupa di eliminare tutti i file contenuti nella directory Documenti/lavoro e poi di eliminare la directory stessa.

Anche rm accetta path multiple e wildcards per pattern complessi

Attenzione! 'rm' può essere molto pericoloso e può eliminare anche file o directory importanti. Se non siete più che certi aggiungete l'opzione -i ed il sistema vi chiederà conferma prima di procedere.

Esercizi

1. Creare un file vuoto
2. Creare una directory vuota
3. Copiare un files
4. Rinominare un file o directory
5. Cancellare un file o directory
6. Elencare i file in una directory con le opzioni più comuni

Domande

1. Quale è il comando necessario alla creazione di una directory?

A. makedir B. mkd C. mkdir D. mdir

2. Indica i comandi corretti per creare la directory "/home/user1/Documenti" con home dir "/home/user1" e working directory ~

A. mkdir ~/Documenti B. mkd /home/user1/Documenti

C. mdir Documenti D. mkdir Documenti

3. Con quali comandi è possibile eliminare una directory?

A. deldir B. rd C. rm D. rmdir

4. Con quali comandi è possibile eliminare le directory vuote ~/dir/subdir e ~/dir avendo ~ come working directory?

A. rmdir -R dir B. rmdir dir/subdir dir

C. rmdir dir dir/subdir D. rmdir -p dir/subdir

5. Quale comando si puó utilizzare per creare un file vuoto?

A. cfile B. make file C. touch D. mkfile

ISBN: 9788866850090

6. Quali comandi si possono utilizzare per creare i file vuoti "file1" e "file2" nella propria working directory?

A. mkfile file1 file2
B. touch ~/file1 ~/file2
C. mkfile /file1 /file2
D. touch file1 file2

7. Quale comando si può utilizzare per copiare dei file?

A. cp B. mv C. copy D. cf

8. Quali comandi sono corretti per copiare tutti i file contenuti nella directory "dir1" nella directory "dir2"

A. copy dir1 dir2
B. cp dir1/* dir2
C. cp -R dir1 dir2
D. mv dir1 dir2

9. Quale comando server per muovere un file?

A. move B. cp C. touch D. mv

10. Quali comandi sono corretti per spostare tutti i file con estensione .txt presenti nella directory Documenti nella directory Documenti/txt

A. move *.txt Documenti.txt
B. move Documenti/*.txt Documenti/txt
C. mv *.txt Documenti.txt
D. mv Documenti/*.txt Documenti/txt

11. quale comando server per eliminare un file?

A. rm
B. rmfile
C. rmd
D. remove

12. Quale opzione bisogna aggiungere a rm per richiedere una conferma prima di eliminare un file?

A. -c B. v C. -r D. -i

Risposte:

1. C
2. A,D
3. C
4. B,D
5. C
6. D
7. A
8. B
9. D
10. D

11. A
12. D

ithum®

ISBN: 9788866850090

Parte 3:
La Potenza della linea di comando

ithum®

9 Archiviazione file sulla linea di comando

L'archiviazione consente di raccogliere senza comprimerli tanti file in uno unico, in modo da consentire una trasportabilità più comoda e veloce.

9.1 TAR

Archiviare i file (mettere tanti file in un unico file) è un po' come metterli in una cartella, avendo però come risultato un file singolo.

Lo strumento usato su Linux per fare questa operazione è tar, un acronimo di "Tape ARchive", in quanto originariamente era stato pensato per i dispositivi ad accesso sequenziale come le unità di backup a nastro.

Un aspetto importante di tar è che con l'archiviazione non si perdono le informazioni dei file, come data e ora, utente, gruppo, permessi e struttura delle cartelle.

La sintassi per archiviare è la seguente:

tar [OPZIONE...] [ARCHIVIO.TAR] [FILE1] [FILE2] [FILE3]

con cui si riuniscono file1, file2 e file3 nell'unico file archivio.tar.

Le opzioni più comuni sono:

-c "crea"

-v "verbose"

-f "file"

-rf "aggiungi/sostituisci"

Utilizzando l'opzione -rf si può aggiungere uno più file ad un'archivio tar esistente

tar -rf [ARCHIVIO.TAR] [FILE]

La sintassi per archiviare un'intera cartella è la seguente:

tar [OPZIONE...] [ARCHIVIO.TAR] [CARTELLA]

La sintassi per estrarre i file da un'archivio tar è la seguente:

tar [OPZIONE...] [ARCHIVIO.TAR]

le opzioni disponibile sono:

-x "estrai"

-v "verbose"

-f "file"

-tf "legge il contenuto dell'archivio"

ISBN: 9788866850090

l'opzione verbose consente di visualizzare sullo schermo i nomi dei file archiviati.

Per estrarre un solo file da un archivio si può usare la seguente sintassi:

tar -xvf [ARCHIVIO.TAR] [FILE1]

Solitamente, visto che tar non consente la compressione dei dati viene usato in unione a programmi di compressione dati.

Si può effettuare archiviazione e compressione in un solo comando utilizzando la seguente sintassi

tar [OPZIONE...] [ARCHIVIO.TAR.GZ] [FILE1] [FILE2] [FILE3]

le opzioni disponibile sono:

-c "crea"

-z "compressione"

-f "file"

9.2 ZIP

È un utility che consente l'archiviazione di file e directory in modalità compressa

zip [OPZIONE...] [NOMEFILE.ZIP] [FILE1]

Le opzioni che si possono utilizzare sono:

- -9 indica che il file o la cartella da comprimere verranno compressi al massimo. Potevamo scegliere un numero tra 0 e 9, tali interi indicano rispettivamente poca compressione o il massimo della compressione che potremmo ottenere (0 indica di archiviare senza effettuare compressioni). Ovviamente, più il numero sarà alto più tempo servirà per creare l'archivio.
- -t Testa se la compressione avrà successo, utile se vogliamo cancellare i file originali senza ricevere brutte sorprese.
- -m Cancella i file originali dopo aver creato l'archivio.
- -e Cripta l'archivio che vogliamo creare impostando una password che verrà richiesta tramite un prompt che non visualizzerà a schermo i tasti che digiterete. Con l'opzione --password potrete scegliere immediatamente una password scrivendola dietro tale opzione ma questo è insicuro perchè tale chiave verrà passata in chiaro.

Per decomprimere un file .zip si usa il comando unzip, con la seguente sintassi:

unzip [nomefile.zip]

9.3 GZIP

È un programma open-source per la compressione dei dati.

Il suo nome è la contrazione di GNU zip. Fu inizialmente creato da Jean-Loup Gailly e Mark Adler. La versione 0.1 fu rilasciata pubblicamente il 31 ottobre 1992. La versione 1.0 vide invece la luce nel febbraio del 1993.

Normalmente ogni file verrà rimpiazzato da uno con l'estensione .gz, mantenendo le stesse proprietà, date d'accesso e di modifica (l'estensione predefinita è gz per Linux o OpenVMS, z per MS-DOS, OS/2 FAT, Windows NT FAT e Atari).

Gzip usa l'algoritmo di Lempel-Ziv usato in zip e PKZIP. L'ammontare della compressione ottenuta dipende dalla dimensione dell'ingresso e dalla distribuzione delle sotto-stringhe comuni. Tipicamente, testi come codici sorgenti o Inglesi sono ridotti del 60-70%. La compressione è generalmente molto migliore di quella ottenibile da LZW (usato in compress), codifica di Huffman (usata in pack), o codifica di Huffman adattativa (compact).

La sintassi del comando è la seguente:

```
gzip [OPZIONE...] [FILE...]
```

Per decomprimere un file .gzip si usa il comando gunzip, con la seguente sintassi:

```
gunzip [FILE.GZ]
```

9.4 BZIP2

bzip2 è un algoritmo di compressione dati libero da brevetti e open source. Sviluppato da Julian Seward, venne rilasciato pubblicamente nel luglio del 1996. La sua popolarità aumentò in poco tempo in quanto la compressione era elevata e stabile. bzip2 produce con la maggior parte dei casi file compressi molto piccoli rispetto a gzip o ZIP, tuttavia ne "paga" in prestazioni essendo leggermente più lento.

Ciò nonostante, con il costante effetto della legge di Moore che rende il tempo-macchina sempre inferiore e meno importante, i metodi di elevata compressione come bzip2 sono diventati più popolari. Effettivamente, secondo l'autore, bzip2 contiene all'interno dal dieci al quindici percento del miglior algoritmo di compressione attualmente conosciuto

La sintassi del comando è la seguente:

```
bzip2 [OPZIONE...] [FILE...]
```

Per decomprimere un file .bunzip2 si usa il comando bunzip, con la seguente sintassi:

```
bunzip2 [FILE...]
```

Domande:

1. TAR è un'utility che serve per...
- A. comprimere file
- B. archiviare file e directory
- C. verificare i permessi su filesystem

2. Per la creazione di un archivio si usa...
- A. tar xvf archivio.tar file1 file2
- B. tar cvf archivio.tar file1 file2
- C. zip archivio.zip file1 file2

3. Quale differenza c'è tra archiviazione e compressione?
- A. nessuna, sono due modi diversi di fare la stessa cosa.
- B. l'archiviazione ha un'affidabilità maggiore
- C. l'archiviazione permette di mantenere intatte le informazioni dei file

4. È possibile aggiungere file ad un archivio esistente?
- A. no, si deve estrarre i file presenti e ricreare l'archivio con i nuovi file
- B. si, ma non direttamente con tar
- C. si, con l'opzione "r"

5. È possibile archiviare e comprimere con una sola istruzione?
- A. si usando l'opzione "z" del comando tar
- B. no, archiviazione e compressione sono due operazioni distinte
- C. si, ma i file devono essere tutti dello stesso formato

6. È possibile aggiungere file ad un archivio già esistente?
- A. no, bisogna estrarre i file e ricreare l'archivio con i nuovi file da includere
- B. si, con l'opzione "r" dell'utility tar
- C. si, ma usando usando un'utility di terze parti

7. Quali delle seguenti utility si usano per la compressione?
- A. zip, gzip, bzip2
- B. zip, gizip, bzip2
- C. zip, gzip, bzip4

8. Qual'è la sintassi corretta dell'utility ZIP?
- A. zip [opzioni] nomefile.zip file1 file2
- B. zip [opzioni] file1 file2 nomefile.zip
- C. nessuna delle precedenti

9. Quali vantaggi si hanno usando bzip2 invece di zip
- A. nessuno
- B. una maggiore velocità di compressione
- C. un file compresso con dimensioni minori

10. Come si decomprime un file compresso con bzip2?
- A. unzip nomefile
- B. unbzip nomefile
- C. bunzip nomefile

Risposte:

1. B

2. B
3. C
4. C
5. A
6. B
7. A
8. A
9. C
10. C

ithum®

ISBN: 9788866850090

10 Ricerca ed Estrazione di Dati da File

Questo capitolo si occuperà di analizzare, in termini essenziali, gli aspetti e gli strumenti che riguardano il reindirizzamento dello standard input ed output, nonchè la ricerca ed estrazione di dati.

Trattandosi di un aspetto chiave trasversale, si tratteranno dapprima le *espressioni regolari* (più comunemente chiamate in forma breve *regex o regexp*), che sono semplicemente un modo per descrivere uno schema da utilizzare per ricercare dei dati.

Si analizzerà poi come possa essere reindirizzato l'input e l'output di programmi. Infine verrà analizzata l'esposizione di alcuni comandi, che permettono di espletare funzioni di ricerca, estrazione nonchè statistica, tra i quali *find*, *grep e wc*.

10.1 Le espressioni regolari

Questo argomento è trattato in maniera approfondita in manuali specializzati. In questa sede ci si limiterà alle sue analizzare le basi e ancor più specificatamente ci si riferirà alle espressioni regolari definite con gli standard POSIX (POSIX: Portable Operating System Interface per Unix, è il nome che indica una famiglia di standard definiti dall'IEEE. Questi standard derivano da un progetto, iniziato circa nel 1985, finalizzato alla standardizzazione delle API (Application Program Interface) per i software sviluppati per le diverse varianti dei sistemi operativi UNIX).

Come appena introdotto, le espressioni regolari possono essere viste come un modo di esprimere uno schema in un testo, stringa di caratteri, chiamato anche *pattern*. Se si vuole, possiamo trovare qualche similitudine con il principio dei caratteri jolly (wildcards) che a volte sono utilizzati quando elenchiamo il contenuto di una cartella ad esempio.

Le due forme più comunemente usate sono: *la base* e l'*estesa* (Basic Regular Expression BRE, Extended Regular Expression ERE). Le differenze tra queste due forme sono articolate ed anche sottili, ma nei principi sono simili.

Nella sua forma più basilare, può essere rappresentata da una parola o stringa di caratteri (alfabetica [a-zA-Z] o alfanumerica [a-zA-Z0-9]), come *scuola, abc, def123* o anche semplicemente da un carattere *a*. Ad esempio, l'espressione regolare *casa* troverà riscontro in stringhe quali *casa dolce casa* oppure *questa strada conduce a casa mia* ma non troverà riscontro in *questo non è un caso*. La vera forza delle espressioni regolari comunque, risiede nell'utilizzo di caratteri non alfanumerici (metacaratteri), che permettono di attivare più avanzate tecniche di riscontro.

Fra le principali caratteristiche delle espressioni regolari ***base*** includiamo:

- **'^'** Il Control (chiamato anche "caret"). Questo carattere posto all'inizio di un'espressione regolare, ad esempio *^casa*, avrà l'effetto di ricercare l'espressione all'inizio di ogni stringa, quindi troverà riscontro in *casa dolce casa* ma non in *questa strada conduce a casa mia*.
- **'$'** Il dollaro. In opposizione al precedente, questo carattere posto alla fine di un'espressione regolare, ad esempio *casa$*, avrà l'effetto di ricercare l'espressione alla fine di ogni stringa, per cui riscontrerà ancora una volta in *casa dolce casa* ma non in *questa strada conduce a casa mia*.
- **'.'** Il punto. Se avete un po di familiarità con la console, lo troverete simile al '?'. Troverà corrispondenza con qualsiasi carattere ad eccezione del Newline, quindi *b.g* riscontrerà parole come *bag, beg, b_g, btg, bAg* e così via.
- **'*'** L'asterisco. Il suo significato è la ripetizione, un'espressione regolare o anche parte di essa, seguita da questo carattere sarà soddisfatta se incontra zero o più riscontri. Spesso lo incontriamo in congiunzione con il '.' (ma non obbligatoriamente). Per esempio, *M.*Rossi* trova corrispondenza con *Mario Rossi*, *M_Rossi* o *MRossi*.
- **'\'** Il backslash. Come possiamo fare per cercare la corrispondenza di caratteri speciali (metacaratteri) ? Esattamente anteponendo il backslash prima del carattere ricercato. Ad esempio, *www\.google\.com* oppure *c:\\windows\\temp*.
- **Bracket expressions** composte da caratteri racchiusi tra parentesi quadre '[]'. Ogni carattere al loro interno verrà ricercato per il riscontro. Ad esempio *b[aeiou]g* troverà corrispondenza in parole come *bag, beg, big, bog* e *bug*. Se includiamo il metacarattere ^ appena dopo la [avrà l'effetto di cercare ogni carattere che non sia compreso all'interno delle parentesi quadre. Quindi, *b[^aeiou]g* riscontrerà *btg, bAg* ma non *big, beg* o *bug*. I più attenti avranno notato come le maiuscole o minuscole fanno differenza e anche il comportamento del metacarattere ^ sia diverso se portato dentro le parentesi quadre.
- **Range expressions** è una variante della precedente. Invece di elencare ogni carattere si preferisce elencare il carattere di inizio e di fine separati da '-'. Ad esempio, *mp[2-4]* troverà riscontro in nomi di file quali: *qualcosa.mp2, qualcosa.mp3* e *qualcosa.mp4* ma non ovviamente in *qualcosa.mpg*.

La forma ***estesa*** aggiunge alcune caratteristiche, fra cui:

- **'+' e '?'** Operatori di ripetizione addizionali. Sono simili all'asterisco '*', ma con un comportamento differente. Ad esempio il '+' incontra una o più ricorrenze, mentre il '?' zero o al più una ricorrenza, rendendo di fatto opzionale quanto ricercato. Ad esempio, *dar?do* troverà corrispondenza in *dado* e *dardo*, mentre *dar+do* riscontrerà solo in *dardo, darrrrdo* e così via.
- **'|'** Il pipe. Permette di cercare più ricorrenze distinte, ad esempio *casa|caso* riscontrerà tanto in *questa è casa mia* quanto *non è un caso che sia quà*.
- **'()'** Le parentesi tonde. Ci permettono di raggruppare delle espressioni a cui potremo voler applicare un operatore, di ripetizione ad esempio. Un'espressione

ISBN: 9788866850090

regolare come *Set(Value|NotValue)?* corrisponderà a *Set*, *SetValue* o *SetNotValue*, in pratica cercherà necessariamente *Set* ed opzionalmente *Value* o *NotValue*.

Questi sono solo alcuni degli aspetti principali; molto altro sarebbe da aggiungere.

La forma da utilizzare dipende principalmente dal programma e/o comando utilizzato: alcuni ne supportano una, altri entrambe. È bene ricordare che le espressioni regolari contenenti metacaratteri associati alla forma estesa avranno comportamenti diversi a seconda di quale forma si sta utilizzando e comunque la scelta di quale forma a volte è data da una semplice questione di gusto personale.

10.2 Reindirizzamento di input/output e pipeline

Molto spesso ci si trova di fronte a comandi e/o programmi che producono degli output consistenti; a volte si è di fronte anche alla necessità di doverli memorizzare. Il reindirizzamento permette di reindirizzarli in un file. E' possibile anche il percorso inverso, ovvero reindirizzare il contenuto di un file ad un comando o addirittura reindirizzare l'output di un comando verso l'input di un altro comando. Il comando *xargs* (che sarà analizzato successivamente) ancora permette di costruire ed eseguire linee di comandi dallo standard input.

Di default ci sono sempre tre file aperti:

- lo standard input (stdin, la tastiera),
- lo standard output (stdout, lo schermo)
- lo standard error (stderr, i messaggi di errore, normalmente anch'essi indirizzati sullo schermo).

Ad ognuno di questi tre file è associato un numero descrittore di file:

- per lo stdin è lo 0,
- per lo stdout è 1
- per lo stderr il 2.

Come anticipato questi file, come qualsiasi altri file possono essere reindirizzati, di seguito vedremo come.

Il reindirizzamento è rappresentato dai simboli > (reindirizzamento dell'output) e < (reindirizzamento dell'input), processati nell'ordine in cui appaiono nella riga di comando, da sinistra a destra.

Infine, il simbolo del reindirizzamento potrebbe essere preceduto da un numero n descrittore di file, che se omesso, nel caso del reindirizzamento dell'input:

- n< miofile
- 'n' è da intendersi lo 0, reindirizzando quindi "miofile" verso lo standard input. Se invece siamo di fronte al reindirizzamento dell'output:
- n> miofile

- 'n' è da considerarsi 1, reindirizzando quindi lo standard output verso "miofile".

La seguente tabella evidenzia le principali forme di reindirizzamento:

Tabella 15 - Forme di reindirizzamento

Opzione	Significato
> miofile	Crea un nuovo *miofile* con il contenuto dello stdout, qualora il *miofile* già esista, verrà sovrascritto.
>> miofile	Il contenuto dello stdout viene appeso al *miofile* esistente, qualora il *miofile* non esista, verrà creato.
< miofile	Il contenuto di *miofile* sarà utilizzato come standard input.
<<	Permette l'inserimento di righe di testo dallo standard input.
2> miofile	Crea un nuovo *miofile* con il contenuto delle standard error, qualora il *miofile* già esista, viene sovrascritto.
2>> miofile	Il contenuto dello stderr viene appeso al *miofile* esistente, qualora il *miofile* non esista, verrà creato.
&> miofile	Permette di reindirizzare lo stdout e stderr a *miofile*.
n<> miofile	Apre il *miofile* in modalità di lettura e scrittura sul descrittore di file 'n'. Se 'n' è omesso è da intendersi 0, cioè lo stdin.

(Tab. 3.2_1)

Vediamo ora qualche esempio di reindirizzamento, partendo con un semplicissimo comando di *echo* :

```
$ echo "Questo testo è scritto \
> su più righe \
> ma stampato in una sola e reindirizzato su di un file" > fileditest
```

Abbiamo semplicemente chiesto al comando *echo* di stampare un testo su più righe (il backslash ci ha permesso di andare a capo), per poi reindirizzarlo su di un file chiamato *fileditest*. Al lato pratico non vedrete alcun "effetto", dopo aver premuto invio

semplicemente ci si ritroverà sul prompt della console creando un file di nome *fileditest* il cui contenuto può essere facilmente visualizzato con il comando *cat,* che vedremo con maggior dettaglio più avanti:

```
$cat fileditest
```

Analizziamo ora un esempio di reindirizzamento utilizzando il comando *grep* (anche questo comando sarà approfondito più avanti). Questo comando è in grado di ispezionare il contenuto dei file in una cartella, per cercare una parola e/o altro. In questo esempio si cercherà cercherò il nome utente (username):

```
$grep johndoe /etc/*
```

Vi ritrovere un certo numero di righe in output, non troppo dissimili dalle seguenti:

```
grep: /etc/at.deny: Permesso negato
grep: /etc/fuse.conf: Permesso negato
/etc/group:adm:x:4:johndoe
/etc/group:cdrom:x:24:johndoe
/etc/group:sudo:x:27:johndoe
...
```

Avrete notato, che l'output porta con se tutti i riscontri che il *grep* è riuscito a trovare, assieme ai suoi messaggi di errore di "Permesso negato" (in quanto con questo utente, non possiedo i privilegi di root). Proviamo ora lo stesso comando, reindirizzando i messaggi di errore al device null, ove non produrranno alcun effetto e semplicemente si perderanno nel nulla :

```
$ grep johndoe /etc/* 2>/dev/null
```

producendo quanto segue :

```
/etc/group:adm:x:4:johndoe
/etc/group:cdrom:x:24:johndoe
/etc/group:sudo:x:27:johndoe
...
```

Come risultato finale l'output del comando sarà regolarmente stampato sullo schermo, mentre i messaggi di errore verranno reindirizzati verso il "nulla". Se invece avessimo desiderato reindirizzare il tutto su di un file (che chiameremo risultati.txt), output del comando ed errori compresi:

```
$ grep johndoe /etc/* &>  risultati.txt
```

Ecco che il reindirizzamento al file risultati.txt conterrà tutto quanto desiderato. Abbiamo infatti, detto di reindirizzare lo stdout del comando al file e lo stderr allo stdout.

Spostiamo ora la nostra attenzione sulle pipelines, rappresentate dal simbolo "|". Questo strumento ci permette di indirizzare l'output di un comando, verso l'input di un altro comando.

Il comando *dmesg* ad esempio, visualizza una consistente serie di messaggi del kernel di linux riguardanti l'hardware e molti altri task di basso livello, può essere pertanto conveniente reindirizzarlo tramite una pipeline al comando *less* per una più agevole visualizzazione:

```
$ dmesg | less
```

Il comando *less* ci permetterà facilmente di scrollare in avanti ed indietro con i tasti PGUP e PGDOWN, di cercare in avanti parole con "/qualcheparola" oppure indietro con "?qualcheparola", "/" per ripetere la ricerca e "q" per uscire. Ovviamente *man less* ci fornirà una spiegazione completa di questo comando.

Diversamente, potremo voler semplicemente visualizzare le ultime 5 righe, con l'aiuto del comando *tail*:

```
$ dmesg | tail -5
```

Infine, se vogliamo cercare i messaggi riguardanti la nostra scheda di rete, supponendo eth0 ad esempio, potremo scrivere:

```
$ dmesg | grep eth0
```

A questo punto qualcuno di voi potrebbe chiedersi esattamente la differenza tra un reindirizzamento ">" ed una piplines "|". La differenza tra un comando:

```
$ comando1 > tempfile; comando2 < tempfile
```

e

```
$ comando1 | comando2
```

Nel primo esempio la shell interpreterà il simbolo di reindirizzamento prima dell'esecuzione del *comando1*, predisponendo lo stdout per essere indirizzato in *tempfile*, esegue poi il *comando1* il cui output si riverserà nel file.

Quando l'esecuzione di *comando1* è finita (il puntovirgola divide l'esecuzione dei due comandi), la shell procederà indirizzando il *tempfile* allo stdin (dirà quindi che lo stdin non è la tastiera ma bensì il contenuto di *tempfile)*, ed infine eseguirà il *comando2*.

Nel secondo esempio, con la pipeline, la cosa è un po' differente. La shell, prima di tutto, crea una connessione (la traduzione letterale di "pipe" è "tubo") tra lo stdout del *comando1* ed lo stdin del *comando2*, dopodichè esegue i due comandi. Il *comando1* il cui output è indirizzato alla "estremità di scrittura" della pipe ed il *comando2* il cui stdin è connesso alla "estremità di lettura" della pipe, la cosa da notare è che i due processi vengono eseguiti insieme senza la creazione di alcun file.

10.3 Ricerca ed estrazione di dati

Comandi come *grep* e *find*, utilizzano le espressioni regolari e sono molto utili per localizzare file e guardarne il contenuto. Il primo, ricerca all'interno del contenuto del file e ritorna la riga contenente il riscontro dell'espressione regolare, *find* invece svolge il suo

compito ricercando per nome del file, data, dimensione ed in molti altri modi. Il comando *wc* invece, fornisce delle statistiche del file di testo dato in ingresso, come il numero di parole, di righe, di caratteri o altre informazioni.

10.3.1 Il comando *grep*

Questo comando porta con se molte opzioni con cui sarebbe utile familiarizzare, quindi provare a digitare *grep --help* oppure *man grep* è altamente consigliato. Quì di seguito, sono esposte alcune delle più comuni:

Forma breve	Forma estesa	Significato
-G	--basic-regexp	Il pattern (schema di ricerca) è una espressione regolare base (BRE) (n.b. modalità predefinita).
-E	--extended-regexp	Il pattern è una espressione regolare estesa (ERE).
-F	--fixed-strings	Il pattern è un gruppo di stringhe (n.b. non espressioni regolari) separate dal carattere Newline.
-R oppure -r	--recursive	Controlla ricorsivamente anche nelle sottocartelle.
-c	--count	Dà in risposta il numero di corrispondenze trovate per ogni file.
-e	--regexp=PATTERN	Seguita dal pattern indica, l'espressione regolare da utilizzare.
-f	--file=FILE	Indica al comando di prendere il pattern dal file.
-i	--ignore-case	Non terrà conto delle maiuscole o minuscole in fase di ricerca.

Supponiamo di voler controllare eventuali messaggi di sistema relativi alla vostra scheda di rete, normalmente identificata come *eth0*. Il seguente comando ispezionerà il file di log di sistema mostrandovi quanto cercato:

```
#grep eth0 /var/log/syslog
```

oppure, se il computer è dotato di piu schede di rete, ad esempio se dotato anche di scheda wifi:

```
#grep eth[0-9] /var/log/syslog
```

Così facendo, cerchiamo per ogni dispositivo di rete associato. Da notare che nella distribuzione utilizzata per questi esempi il log di sistema è il file */var/log/syslog* ma

potrebbe essere anche */var/log/messages*, dipende dalla distribuzione di linux che state utilizzando.

Se volessimo ispezionare per questo contenuto, tutti i file di una cartella ? La cosa non è poi così complicata, con quanto abbiamo visto:

```
#grep -r eth[0-9] /etc/*
```

Questo comando effettuerà quanto desiderato, cercando ricorsivamente all'interno di ogni sottocartella e così via. Dobbiamo comunque tener presente sempre "cosa" vogliamo ispezionare, la cartella /etc è una cartella di "sistema" ove sono richiesti i privilegi di root per poter leggere alcuni contenuti.

Articolando ancora un attimo, potremo visualizzare se presenti eventuali messaggi di errore del sistema reindirizzando (con il "|" pipeline, che approfondiremo più avanti) al comando *grep*, l'output del comando *dmesg*:

```
$dmesg | grep -i -E "(error|fail)"
```

Nel caso più fortunato che non siano presenti errori, non verrà visualizzato nulla, ciò capita assai di rado, quindi non meravigliatevi di un output.

```
$ps aux | grep -i root
```

Quest'ultimo esempio infine, ci permette di visualizzare tutti i processi eseguiti dall'utente *root.* Potremo volere anche, cercare al posto di *root* il nome di un processo ad esempio *init*.

10.3.2 Il comando *find*

Questo comando, in sintesi, permette di cercare dei file con varie modalità, ad esempio:

- per nome o schema di ricerca (regexp),
- per data di creazione,
- per tipo di permessi oppure se hanno acceduto al file negli ultimi 15 minuti ad es. e così via.

Il comando di ricerca viene eseguito a partire dal percorso da noi indicato (possono essere anche più di uno) ricorsivamente attraverso le sottodirectory. Ricordiamoci che se il percorso di partenza da cui cercare viene dato in forma relativa, il result sarà anch'esso espresso con percorso relativo, diversamente, se in dato in forma assoluta, anche il result avrà un percorso assoluto. La tabella sotto riportata rappresenta alcune delle opzioni più comuni:

Opzione	Significato
-name filename/pattern	Questa opzione affiancata dal nome del file o dallo schema di ricerca (pattern), varie forme di espressioni regolari sono supportate, indica cosa cercare.
-iname	Come sopra, ma indifferente a maiuscole o minuscole.

filename/pattern	
-regextype type	Permetterà al comando find di capire con quale forma di espressione regolare avrà a che fare. Attualmente sono supportati: emacs (predefinita), posix-awk, posix-basic, posix-egrep and posix-extended.
-amin n/+n/-n	E' stato effettuato accesso il file esattamente n minuti fa, o da più di n minuti, o meno di n minuti fa.
-atime n/+n/-n	Come per l'opzione precedente, si contano i giorni (24h).
-mmin n/+n/-n	Come l'opzione precedenti, con la differenza che controlla quando il file è stato modificato.
-mtime n/+n/-n	Come l'opzione precedenti, con la differenza che controlla quando il file è stato modificato.
-size n[cwbkMG]	b sta per 512-byte block, c sta per byte, w per two-byte words, k per kilobyte, M per Megabyte infine G sta per Gigabyte. Ci permette di cercare i file per dimensione, anche qui possiamo usare gli operatori +/- per cercare se piu grandi di o più piccoli di.
-user uname	Specifica il nome dell'utente a cui il file appartiene.
-uid n	Specifica il numero ID dell'utente a cui il file appartiene.
-group gname	Specifica il nome del gruppo a cui il file appartiene.
-gid n	Specifica il numero ID del gruppo a cui il file appartiene.
-perm mode	Ci permette di cercare i file che hanno un determinato tipo di permessi, così come definiti dal comando chmod.
-delete	Avrà l'effetto di cancellare il file trovato
-exec command ;	Comporterà l'esecuzione del comando dato per ogni file trovato

Vediamo ora alcuni esempi:

```
$find ~ -perm 644
```

Questo comando elencherà esattamente, tutti i file non eseguibili, ma leggibili da tutti e scrivibili solo dal proprietario, presenti nella nostra home. Da notare che se avessimo

utilizzato l'opzione *-perm -644*, avremmo trovato quei file che rispondevano almeno ! ai permessi richiesti, ma non solo... anche un file con il bit di esecuzione attivo (es. 755), sarebbe elencato.

```
$find ~ -perm 644 -exec ls -al '{}' \;
```

Lo stesso comando condito con l'opzione -exec, eseguirà il comando ls -al per ogni file trovato, listando così i dettagli del file. Da notare le due parentesi graffe, che verranno sostituite con il nome del file trovato da *find*, protette dagli apici per evitare che vengano interpretate dalla shell. Il carattere di escape "\" (backslash), davanti al puntovirgola ha la stessa funzione di protezione degli apici.

```
$ find ~ -amin -5
```

Quì invece, andremo in cerca di tutti i file nella nostra home che abbiano avuto un accesso negli ultimi 5 minuti.

Complichiamo ancora un po', e con un semplice "tocco" creiamo un file chiamato *"altrofileditest"* :

```
$ touch altrofileditest
```

dopodichè, cerchiamolo per eliminarlo...

```
$ find . -name altrofileditest -type f -print | xargs /bin/rm -f
```

Quì abbiamo cercato il nostro *"altrofileditest"* appena creato, tramite la pipeline "|" lo abbiamo passato al comando *rm* (remove) per il tramite di *xargs*. Il comando *rm* ha bisogno del suo argomento, cioè il file da eliminare, *xargs* dice ad *rm* qual'è questo argomento, passandogli il filename trovato dal comando *find*. Certo, i più esperti di voi mi diranno che in quest'ultimo esempio si poteva utilizzare l'azione *-delete* del comando *find*, vi chiedo di portare pazienza e di guardarlo dal punto di vista didattico.

Con la pratica, il comando *find* in congiunzione con il *grep* risulterà molto utile. Concludiamo ora con questo ultimo esempio:

```
#find /etc -name interfaces | xargs grep -i -E "(eth0|eth1)"
```

Con quest'ultimo comando, si vuole cercare un file di nome *"interfaces"* che, se trovato, verrà passato come argomento al comando *grep* via pipeline per il tramite di *xargs*, al fine di cercare la presenza di eventuali righe contenenti le stringhe *"eth0"* o *"eth1"*, utilizzando le regexp in forma estesa.

10.3.3 Il comando cat

La funzione di questo comando è quella di concatenare uno o piu file e stamparti sullo standard output. Fortunatamente non ha tante opzioni da ricordare, nella tabella a seguire le due più comuni :

Opzione	Significato

-n	Numera tutte le linee stampate.
-b	Come sopra, ma non conteggia le linee vuote (sovrascrive l'opzione -n).

Solo per puro spirito didattico, digitiamo ora:

```
$ man head > manualedihead; man tail > manualeditail
$ cat manualedihead #stampa il file "manualedihead" appena creato, sullo schermo
$ cat manualeditail #come sopra, per il file "manualeditail"
```

mentre

```
$cat manualedihead manualeditail #produce la stampa concatenata di entrambi
```

che possiamo, oltre a stamparla sullo stdout, anche reindirizzarla su di un nuovo file e magari cercare al suo interno le righe che contengono la parola "OPTION":

```
$cat manualedihead manualeditail | tee manualidientrambi | grep OPTION
```

con il seguente output :

```
head [OPTION]... [FILE]...
tail [OPTION]... [FILE]...
```

Vediamo cosa succede in questa riga di comando: nella prima parte, il comando *cat* produce la stampa concatenata dei due file verso lo stdout, la prima pipeline collega questo stdout allo stdin del comando *tee*, il quale ha il semplice ed utile compito di duplicare quanto gli arriva in ingresso, una copia la continua ad inoltrarla allo stdout, l'altra la indirizza sul file "*manualidientrambi*". Infine la seconda pipeline raccoglie quanto arriva dallo stdout del comando *tee* e lo inoltra allo stdin del comando *grep* che processa le righe alla ricerca della stringa "OPTION", con l'output sopra evidenziato.

10.3.4 Il comando *wc*

Nei capitoli precedenti abbiamo imparato come il comando:

```
$ls -al manualidientrambi
```

ci fornisca una serie di utili informazioni, quali:

- permessi sul file,
- proprietario, gruppo,
- data e ora
- dimensione.

Ma se volessimo qualche "informazione dagli interni" di questo file ? Ecco che il comando *wc* si presta allo scopo. Questo comando, corredato di qualche opzione, fornisce una serie di informazioni del file, provate a digitare:

```
$wc manualedientrambi
```

ed otterrete con molta probabilità un output identico nelle informazioni al seguente:

```
158 810 6258 manualedientrambi
```

esattamente, ci sta informando che il file è composto da 158 righe, 810 parole, 6258 byte. Ovviamente si possono passare al comando *wc* anche i due file separati:

```
$wc manualedihead manualeditail

64 290 2298 manualedihead
94 520 3960 manualeditail
158 810 6258 total
```

Come vedete, il conteggio totale non cambia. Nella tab. 3.2_5 a seguire, riporto alcune delle opzioni del comando:

Opzione	**Significato**
-c	Stampa il numero di byte.
-m	Stampa il numero di caratteri.
-w	Stampa il numero di parole, una parola è intesa come una sequenza di uno o più caratteri delimitata da uno spazio.
-l	Stampa il numero di linee.
-L	Stampa la lunghezza della linea più lunga.
In assenza del file in input o se data l'opzione -, legge direttamente dallo standard input (^d per terminare l'inserimento).	

Prima di procedere con il prossimo comando, coraggio siamo in dirittura di arrivo con la conclusione di questo capitolo, permettetemi di giocare con quest'ultimo esempio e digitate il seguente comando, con il testo a seguire, la stringa EOF completerà l'inserimento:

```
$ cat <<EOF | tee unpoditest | wc
> Ciao
> io mi chiamo linux e te?
> Ora ti saluto e ti auguro buono studio.
> EOF
```

ora, salvo l'inserimento di qualche spazio non desiderato, dovreste ottenere questo output:

```
3 16 71
```

mentre digitando:

$cat unpoditest

otterrete la stampa di quanto appena inserito:

Ciao
io mi chiamo linux e te ?
Ora ti saluto e ti auguro buono studio.

Se sono riuscito a trasmettervi qualcosa fino ad ora, quanto digitato sopra non dovrebbe risultarvi così criptico. Vediamo, abbiamo indirizzato al comando *cat* lo standard input, con quella modalità che viene chiamata "here document", indirizzamento che si conclude con l'inserimento della stringa *"EOF"*.

Il comando *cat* stampa nello stdout quanto appena inserito, che tramite la pipeline viene inviato allo stdin del comando *tee* il quale, duplicando quanto ricevuto, da una parte inoltra nuovamente allo stdout e dall'altra crea una copia nel file *"unpoditest"*. Infine, l'ultima pipeline raccoglie quanto prodotto nello stdout dal comando *tee* e lo indirizza allo stdin, dove trova pronto il comando *wc* a processare il testo inserito.

Un ultimo consiglio, questo comando come forse avrete intuito, è efficace con file di testo semplici, ma con file XML, HTML etc... che hanno una loro propria sintassi, fornirebbe dei conteggi non corretti. Per questi tipi di file, bisogna affidarsi ai loro editor.

10.3.5 Il comando *sort*

Questo versatile comando ci offre la possibilità di ordinare le righe di uno o più file di testo o direttamente quanto inserito allo standard input. Vediamo subito, nella tabella 3.2_6, un elenco delle opzioni più comuni per poi passare ad alcuni esempi:

Opzione	Significato
-b	Ignora gli spazi, se presenti all'inizio del campo.
-d	Ordina solo per lettere, numeri e spazi, altro non viene considerato.
-f	Non tiene conto se le lettere sono maiuscole o minuscole.
-k F1[, .. Fn]	Ordina in base al campo F1 e fino a Fn , F di default è 1.
-t char	Il carattere char rappresenta il separatore di campo.
-n	Considera il campo con cui ordinare, come un numero e non una lettera. Gli spazi iniziali saranno ignorati.

Proviamo ora con qualche esempio e prendiamo in considerazione il file /etc/group :

$sort /etc/group

quello che segue è un estratto del suo output, che non dovrebbe essere troppo dissimile dal vostro:

```
...
src:x:40:
ssh:x:112:
ssl-cert:x:107:
staff:x:50:
sudo:x:27:
syslog:x:102:
sys:x:3:
tape:x:26:
tty:x:5:
users:x:100:
utempter:x:134:
utmp:x:43:
uucp:x:10:
...
```

Noterete che è stato ordinato alfabeticamente per riga. Supponiamo ora di volerlo ordinare per numero di gruppo e quindi ordinarlo sul terzo campo:

```
$sort -k 3 -n -t : /etc/group
```

L'opzione "-k 3" fornisce l'istruzione di ordinare sul terzo campo, "-t :" spiega come sono divisi i campi ed infine "-n" gli dice che il campo deve essere considerato come un numero e non una stringa, se siete curiosi provate a vedere il risultato omettendola.

10.3.6 Il comando *cut*

Quest'ultimo comando che tratteremo, ha la funzione di estrarre delle parti, quelle da noi desiderate ovviamente, dalle righe del file passato al comando. La tabella che segue, elenca alcune delle sue opzioni più comuni:

Opzione	Significato
-c LISTA	Seleziona solo la LISTA di caratteri.
-f LISTA	Seleziona solo i campi elencati nella LISTA. Nota bene, stampa anche le righe che non hanno alcun delimitatore di campo, a meno che non sia presente l'opzione "-s"
-d char	Utilizza il carattere char come delimitatore di campo (di default è il TAB, tabulazione).
-s	Non seleziona righe che non contengono delimitatori di campo

ISBN: 9788866850090

sono permesse anche più estrazioni, ad esempio passando l'opzione *"-c 1-3,6-10"* estraggo i primi 3 caratteri ed i caratteri dalla posizione 6 alla 10, ovviamente in questo caso non ha molto senso farlo, se non per l'aspetto didattico.

```
$ cut -c 1-5 /etc/passwd
```

Vediamo ora in concreto qualche esempio, supponiamo di volere visualizzare i primi 5 caratteri del file /etc/passwd :

```
$ cut -d: -f 1,3,4 /etc/passwd
```

In questo caso invece, stiamo estraendo dal file passwd un elenco contenente i nomi utente, il suo numero identificativo ed il numero del gruppo a cui è associato.

Sono cosciente che la trattazione di questo capitolo non può chiamarsi esaustiva, molti altri comandi utili meriterebbero approfondimenti, tra i quali: uniq, tr, nl, join, paste, etc... in questa sede sono stati trattati solo quelli principali. A mio parere, la cosa più importante è che prendiate familiarità con essi per gradi, vi lascio ora con qualche esercizio, buono studio.

Domande:

1. Nel file *`/etc/group`* sono elencati i tutti i gruppi, tra cui 'deamon'. Con il comando *`$grep Daemon /etc/group`* sto estraendo la riga corrispondente a tale gruppo, giusto ?

SI NO

2. Se digito *`$grep -E "[a-z]+\.?\?+"`*, delle seguenti stringhe quali trovano interamente riscontro ? Se presenti, quali parzialmente ?

A) attacco! B) Freddo. C) Caldo? D) che vento.

E) divertimento? F) ok.?

3. Il comando *`$find /etc -name .conf`* elencherà tutti i file di configurazione con estensione ".conf" presenti in */etc* ?

SI NO

4. Nell'esercizio nr.2 è stata utilizzata la seguente *`"[a-z]+\.?\?+"`* regexp. Se avessimo voluto riscontrare la presenza di stringhe che iniziavano con numeri, la seguente regexp *`"[^0-9]+[a-z]+\.?\?+"`* sarebbe una soluzione?

SI NO

5. Il comando *free,* stampa il corrente stato di utilizzo della memoria del computer. Estraete la riga con l'informazione riguardante la memoria di swap.

6. Estraete dal file */etc/group*, gli appartenenti al gruppo *cdrom.*

7. Il file */etc/mtab* contiene una serie di informazione sui filesystem montati dal sistema, analizzatelo ed estraete i primi due campi.

8. Prendendo spunto dall'esercizio precedente, supponete di non conoscere ove sia situato il file *mtab* in */etc*. Cercatelo, scartando l'output di eventuali errori, ed estraete le stesse informazioni, ordinandole e riversandole su di un file di nome *"mtabinfo"*.

9. Analizzate le differenze di questi due comandi :

$grep johndoe /etc/* > vogliotutto 2>&1

$grep johndoe /etc/* 2>&1 > vogliotutto

10. Dal file *protocols* in */etc*, estraete la parte commentata della riga riguardante il protocollo TCP.

Soluzioni:

1. NO ! Ricordiamo che le maiuscole e minuscole fanno la differenza (case sensitive). Il comando corretto è:
$grep daemon /etc/group oppure *$grep -i Daemon /etc/group*
L'opzione *-i* permette di eseguire il comando *grep* in modalità case insensitive.

2. La le scelte E e F riscontrano completamente. La scelta C solo parzialmente in "aldo?", la lettera "C" non viene riscontrata.

3. NO. La versione corretta del comando è *$find /etc -name *.conf*

4. NO. La versione corretta è *"^[0-9]+[a-z]+\.?\?+"*. Notare che il *^ (caret)* è all'esterno della bracket expression.

5. *$free | grep -i swap*

6. *$cat /etc/group | grep -i cdrom | cut -d : -f 4* oppure direttamente *grep -i cdrom /etc/group | cut -d : -f 4*

7. Il comando *$cat /etc/mtab* vi permetterà di analizzarlo, mentre per estrarre quanto richiesto, *$cut -d ' ' -f1,2 /etc/mtab*

8. *$find /etc -name mtab 2>/dev/null | xargs cat | cut -d ' ' -f1,2 | sort > "mtabinfo"*

9. Nel primo caso l'output del comando dato, errori compresi se presenti, viene indirizzato al file *vogliotutto*, nel secondo caso avendo prima dirottato lo stderr verso lo stdout, avremo la stampa degli errori sullo schermo e poi, i riscontri del comando grep sul file. L'ordine dei reindirizzamenti fa la differenza !

10. *$grep TCP /etc/protocols | cut -d '#' -f 2*

ISBN: 9788866850090

11 Inserire comandi negli script (Turning Commands into a Script)

Il sistema operativo Linux permette l'utilizzo del PC da parte di diversi utenti, di conseguenza subito dopo l'accensione occorre autenticarsi (in pratica va effettuato il "login" oppure "logon"). In base alle informazioni che si danno in fase di login, il sistema attribuirà dei diritti d'accesso determinando i privilegi di un determinato account.

L'operazione di creazione degli account è uno dei compiti di un amministratore di sistema: ogni account ha una username ed una password. L'uso di una password (possibilmente composta da almeno 8 tra lettere e numeri) assicura che solo quel determinato utente possa utilizzare quell'account, quindi occorre tenerla segreta e non rivelarla a nessuno. Chiunque conosce username e password di un account può sostituirsi a tale utente per leggere documenti, cancellare file, mandare posta elettronica a nome dell'utente impersonato. Alcune moderne distribuzioni di Linux semplificano il processo di "login" in modo che si possa utilizzare il computer appena acceso, in questo modo non ci sarà autenticazione e ci sarà un'unica sessione. Pur essendo un vantaggio per l'accesso rapido al sistema va evitato sui computer portatili e gli altri dispositivi mobili perché possono essere smarriti.

Il login di solito avviene in un ambiente grafico in cui è già predisposta la propria username. Durante l'inserimento della password verranno inseriti degli asterischi al posto dei caratteri che la compongono per evitare ad altri utenti di poterla leggere. Dopo l'accesso si avvierà l'ambiente desktop grafico vero e proprio (KDE, GNOME, UNITY o altro) che consentirà di utilizzare applicazioni cliccando su menù e icone. La maggior parte degli ambienti grafici per Linux supporta la "gestione delle sessioni" (session management) per ripristinare le sessioni utente in modo più veloce possibile (applicazioni aperte, posizione delle finestre sullo schermo,ecc). All'uscita di ogni sessione di lavoro il computer verrà liberato per essere utilizzato da un altro utente, il gestore delle sessioni (session manager) salverà tutto l'ambiente di lavoro.

A differenza delle workstation, i server supportano quasi esclusivamente un ambiente testuale anziché grafico: in questi casi la schermata login sarà molto semplice (testo bianco su sfondo nero) e al posto degli asterischi durante l'inserimento della password, non sarà visualizzato nulla. Se si inserisce la username e la password correttamente il sistema accetterà il login ed avvierà l'interprete a linea di comando (la shell) per poter inserire i comandi. Subito dopo il login si è posti nella propria "home directory". Anche se si utilizza un ambiente grafico si possono comunque inserire comandi in modo testuale avviando una "console terminale".

11.1 La Shell

La shell è il primo programma che viene eseguito dopo il login, interpreta i comandi digitati direttamente dagli utenti o che possono essere letti da file chiamati script di shell o programmi di shell. Gli script di shell sono interpretati, cioè i comandi vengono letti dallo script riga per riga ed eseguiti. Il sistema LINUX offre una varietà di tipi di shell, ricordiamo le più comuni:

- sh o Bourne Shell: la shell originale ancora utilizzata nei sistemi UNIX, ha poche funzionalità ed è disponibile in ogni sistema LINUX per compatibilità con i programmi UNIX.
- bash o Bourne Again Shell: la shell GNU standard, intuitiva e flessibile consigliata agli utenti principianti pur essendo uno strumento potente per professionisti. In LINUX bash è la shell standard per gli utenti comuni.
- csh o C shell: la sintassi di questa shell a quella del linguaggio di programmazione C spesso utilizzata dai programmatori.
- tcsh o Tenex C shell: un ampliamento della C shell, con miglioramenti nella facilità d'uso e nella velocità.
- ksh o Korn shell: talvolta apprezzata da persone con esperienze UNIX come estensione della Bourne shell decisamente complessa per gli utenti principianti.

Il file /etc/shells offre una panoramica delle shell “conosciute” in un sistema Linux:

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
```

La vostra shell predefinita è impostata nel file /etc/passwd, come in questa linea dell'utente *“utente”*:

```
utente:x:1000:1000:utente,,,:/home/utente:/bin/bash
```

Si può passare da una shell all'altra inserendo da prompt il nome della nuova shell. La shell bash è quella a cui si farà riferimento in questo testo. Quando viene eseguito uno script di shell, bash creerà un nuovo processo bash figlio (child) con ID diverso, a cui affida il compito di eseguire lo script, ricorrendo ad una biforcazione (*fork*); il processo genitore (parent) entra in uno stato di attesa (wait). Mentre la "sottoshell" elabora ogni linea dello script, la shell genitrice aspetta quindi che il processo figlio termini. Quando non ci sono più linee da leggere nello script di shell, la "sottoshell" termina. I comandi di ciascuna linea di uno script vengono letti, interpretati ed eseguiti come se fossero stati digitati direttamente sulla tastiera.

11.2 Inserire comandi da terminale (sleep, echo, date, man)

Prima di cominciare a realizzare degli script di shell può essere utile provare a familiarizzare con la shell stessa, inserendo dei semplici comandi. Si ricorda che i comandi devono essere digitati utilizzando esclusivamente lettere minuscole in quanto la shell è case-sensitive. Avviare una sessione terminale ed iniziare a provare i comandi sleep, echo, date e man ed un primo semplice utilizzo di variabili:

```
$ sleep 10
```

Questo comando genera un pausa circa di 10 secondi. Il simbolo del dollaro ($) rappresenta il prompt comandi in modalità utente (nel caso di root sarebbe #) e non và digitato, viene inserito solo per completezza dell'output visibile sullo schermo del vostro PC, d'ora in poi sarà dato per scontato che non dovete digitarlo.

```
$ echo ciao a tutti
ciao a tutti
```

Scrive su schermo la frase "ciao a tutti".

```
$ p=Tizio
$ echo benvenuto $p
benvenuto Tizio
```

Visualizzerà sullo schermo la frase "benvenuto Tizio".

```
$ p=piano
$ echo ${p}forte
pianoforte
$ f=forte
$ echo $p$f
pianoforte
```

Entrambe le operazioni visualizzeranno sullo schermo la parola “pianoforte”. La prima sequenza di comandi è composta da una parte variabile ed una fissa nella stringa, la seconda sequenza di comandi ha entrambi le parti variabili (la parentesi graffa aperta si ottiene con AltGr+7 la parentesi graffa chiusa con AltGr+0).

```
$ date
```

Questo comando, senza i privilegi di amministratore, consente solo di visualizzare la data e l’ora attuale. Un amministratore (root) invece può utilizzarlo anche per cambiare la data di sistema, infatti il comando:

```
# date 0731093012
```

Imposterà la data attuale al 31 luglio 2012 e l’ora alle 09 e 30. Se volete approfondire ed ottenere ulteriori informazioni sul comandato date, digitate:

```
$ man date
```

11.3 Utilizzo di variabili nella shell (echo, export, env, export, unset, type, hash, which, whereis)

Prima abbiamo visto l’utilizzo di una variabile contente una stringa di testo, ma dobbiamo fare attenzione a distinguere tra il nome della variabile ed il suo contenuto, l’esempio chiarirà meglio:

```
$ miavariabile=Tizio
$ echo miavariabile
miavariabile
```

Visualizzerà sullo schermo la parola “miavariabile”, invece:

```
$.. miavariabile=Tizio
$.. echo $miavariabile
Tizio
```

Visualizzerà sullo schermo la parola contenuta nella variabile “miavaribile” cioè “Tizio”. È importante non inserire spazi dopo il segno = ed è importante inserire il simbolo del $ davanti al nome della variabile se si vuole stampare il suo contenuto, altrimenti verrà stampato il nome della variabile stessa. Anche le variabili (come i nomi dei comandi) sono *case sensitive*, cioè la variabile che si chiama var1 è diversa dalla variabile che si chiama Var1. Dalla versione 2 di Bash è possibile dichiarare le variabili per tipo attraverso il comando declare, ma non affronteremo tale argomento. Se non dichiarato diversamente tutte le variabili sono considerate come stringhe di testo. Ma come possiamo definire, in ambito informatico, il concetto di variabile?

Una variabile identifica uno spazio di memoria destinato a contenere dei dati, che possono essere modificati nel corso dell'esecuzione di un programma. Una variabile è

caratterizzata da un nome alfanumerico inteso solitamente come una sequenza di caratteri e cifre, tale nome inizia sempre con un lettera e mai con un numero. E' buona norma utilizzare nomi significativi per le variabili in modo da auto documentare la funzione che svolgono ed il loro contenuto.

Si distinguono variabili di ambiente e variabili di shell. Le variabili di shell sono visibili solo nella shell nella quale sono definite. Le variabili di ambiente sono passate ai processi figli tramite il comando export:

```
$ miavariabile=Tizio
```

(miavariabile è una variabile di shell)

```
$ export miavariabile
```

(adesso miavariabile è una variabile di ambiente)

Si possono utilizzare più variabili contemporaneamente

```
$ export nome=Pinco cognome=Pallino
```

Digitando il comando export o il comando env senza alcun parametro verrà visualizzato l'elenco di tutte le variabili d'ambiente. Vediamo alcuni nomi standard di variabili di ambiente:

Tabella 16 - Alcune variabili di ambiente

Variabile	Significato
PWD	Nome della directory corrente
USER o USERNAME o LOGNAME	Nome dell'utente corrente (ID utente)
HOME	Home directory dell'utente corrente
PATH	Lista delle directory che contengono programmi eseguibili in modo tale da essere considerati come comandi esterni

Proviamo anche il comando env:

```
$ env nome=Pinco bash
```

(lancia una shell figlia con la variabile nome)

```
$ echo $nome
Pinco
$ exit
```

(ritorna alla shell genitrice)

```
$ echo $nome
```

(non conosce la variabile nome…)

$ _

(…e quindi non visualizza nulla)

In questo caso env ha lanciato una shell figlia con la variabile nome all'uscita dalla shell figlia con il comando exit, la shell genitrice non conosce la variabile nome. Se non occorre più la variabile di shell si può cancellare utilizzando il comando unset.

$ export nome=Pinco

(nome è una variabile di ambiente)

$ export -n nome

(nome adesso è solo una variabile di shell)

$ unset nome

(nome è distrutta per sempre)

Una delle applicazioni delle variabili shell è di controllare la shell stessa. Ad esempio la shell distingue tra comandi interni ed esterni, i comandi esterni corrispondono a programmi eseguibili che vengono cercati in base al valore della variabile ambiente PATH:

$ echo $PATH

/home/utente/bin:/usr/local/bin:/usr/bin:/bin:/usr/games

Notate che PATH è scritto in maiuscolo e non in minuscolo e che i vari percorsi sono separati dal simbolo dei due punti (:), in questo caso ci sono 5 percorsi che verranno presi in considerazione per cercare il comando eseguibile. Se si digita il comando:

$ ls

La shell riconosce che non è un comando interno e quindi lo cerca nelle directory definite indicate nella variabile d'ambiente path. Nel caso del path definito in precedenza il comando ls verrà cercato nelle directory:

/home/utente/bin/ls (non verrà trovato)

/usr/local/bin/ls (non verrà trovato)

/usr/bin/ls (non verrà trovato)

/bin/ls (trovato)

/usr/games (non verrà neanche testata)

Quindi per eseguire il comando ls verrà eseguito /bin/ls. Questa operazione d'identificazione di tutti i file definiti nei percorsi da path, identifica i file in maniera univoca, questo processo è chiamato "hashing" e si può verificare per esempio con il comando hash:

$ hash

```
hits    command
1       /usr/bin/which
3       /bin/date
3       /bin/cat
4       /usr/bin/sudo
1       /bin/sleep
1       /usr/bin/man
2       /usr/bin/env
0       /bin/ls
1       /bin/su
```

Il commando hash indica quali comandi della shell sono stati già indicizzati. Con il commando hash -r si può cancellare completamente la memoria hashing. Se si desidera trovare con precisione dove è memorizzato un comando esterno si può utilizzare il comando which:

```
$ which grep
/bin/grep
$ which ls
/bin/ls
```

Il comando which utilizza tutti i percorsi definiti nella variabile ambiente PATH per verificare se il comando esterno è in una specifica directory. Il comando which non sa nulla al riguardo di comandi interni ed esterni alla shell, infatti se proviamo a scrivere:

```
$ which test
/usr/bin/test
```

Questo non implica che il programma venga eseguito al posto del comando interno test se si vuole essere sicuri (in caso di omonimia) è meglio usare il comando type. Un comando simile al comando which, è il comando whereis che non solo ritorna il nome competo del programma eseguibile ma anche la documentazione con codici sorgenti ed altre informazioni:

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

11.4 Altre funzioni della Shell

Nell'utilizzo della shell è comodo conoscere che sono disponibili anche le seguenti funzioni:

Funzione	Come si ottiene?
Editing in linea dei comandi digitati	Si possono editare linee di comandi come in un semplice editor testuale utilizzando i tasti cursore, i tasti Backspace e Canc , terminando la linea di input con il tasto Invio
Annullare i comandi in esecuzione	Per interrompere un comando in esecuzione utilizzare i tasti Ctrl+c
Storico dei comandi digitati	La shell ricorda i comandi più recenti digitati, per scorrere questo elenco utilizzare i tasti cursore e una volta trovato quello che ci occorre confermare con tasto invio oppure utilizzando i tasti Ctrl+r ed una sequenza di caratteri componenti del comando che si cerca. **N.B**: siccome la history viene memorizzata in un file nascosto ~/.bash_history tutti i comandi digitati sono visibili o editabili aprendo il suddetto file, ad esempio immaginate un cambio password…
Auto completamento nella digitazione dei comandi	La shell permette l'auto completamento dei comandi utilizzando il tasto Tab una volta che ha riconosciuto le iniziali del comando

Esecuzione condizionale di comandi	Può capitare qualche volta di dover eseguire dei comandi dietro condizione, a tal proposito si può sfruttare la caratteristica che ha ogni processo Linux di restituire un valore di ritorno che è 0 in caso di comando riuscito, diverso da 0 in tutti gli altri casi. Ad esempio possiamo utilizzare i caratteri && e || come operatori logici, per dimostrare questo utilizziamo l'opzione –c della shell con la quale passare ad una shell figlia:
	$ bash -c "exit 0" && echo "tutto ok"
	tutto ok
	$ bash -c "exit 33" && echo "tutto ok"
	$_
	Nel primo caso l'and logico è tra due condizioni vere e quindi stampa il messaggio, nel secondo caso non verrà visualizzato nulla perché si fa un and logico tra una condizione falsa ed una vera, vediamo ancora un esempio:
	$ bash –c "exit 0" || echo "tutto ok"
	$_
	Qui invece non succede nulla perché essendo un or logico, basta che la prima condizione sia verificata per procedere, e siccome la prima condizione è già vera non controlla la seconda, sotto invece:
	$ bash –c "exit 33" || echo "tutto ok"
	tutto ok
	Trovando la prima condizione falsa è costretto a verificare anche la seconda che però è solo una echo (quindi sempre vera) e di conseguenza stampa il messaggio "tutto ok".

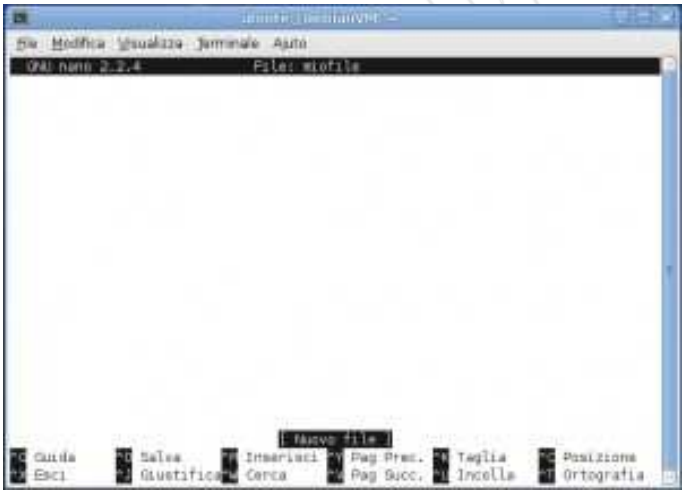
Comandi multipli su una linea	E' possibile inserire più comandi utilizzando semplicemente il punto e virgola (;) ad esempio: $ echo oggi è il ; date oggi è il gio 09 ago 2012, 09.53.46, CEST

11.5 Strumenti per editare script

Per realizzare dei semplici programmi o script, Linux mette a disposizione (tra i tanti strumenti visibili nei menù e nei desktop grafici) anche semplici editor testuali come Nano e Pico, che sono comunque decisamente più spartani dell'equivalente Blocco Note a cui siete abituati in Windows. I vecchi professionisti Linux rimarranno comunque affezionati a Vi. Per avviare GNU Nano da una sessione terminale utilizzare il comando:

$ nano miofile

Nano è il nome dell'editor e miofile è il nome del documento che si vuole modificare.



I menù che sono visualizzati nella parte bassa dello schermo indicano le possibili operazioni:

- Ctrl+x uscita dall'editor
- Ctrl+g richiesta dell'informazione di aiuto
- Ctrl+o scrittura sul file
- Ctrl+r lettura da file
- Ctrl+w cerca testo nel file
- Ctrl+j giustificazione testo
- Ctrl+k taglia testo

- Ctrl+u incolla testo
- Ctrl+y pagina precedente
- Ctrl+v pagina successiva

11.6 Programmare script della Shell (cat, chmod)

Ora che abbiamo visto alcuni comandi possiamo iniziare a realizzare semplici script con la shell ed utilizzarla come linguaggio di programmazione. Capita spesso di compiere operazioni ripetitive e di non ricordare ogni volta correttamente tutti i comandi, quindi un modo efficiente dell'utilizzo del PC è quello di creare programmi una volta per tutte, che lavorino al posto nostro. Il "programma" è contenuto in un file di testo che l'interprete (la nostra shell) processa (effettua il "parsing") ed esegue riga per riga. Il linguaggio di script può differire in base alla shell utilizzata. Prima di iniziare, è d'obbligo definire alcuni parole o termini che si potrebbero incontrare durante il nostro percorso:

Termine	**Significato**
Ciclo o iterazione [*loop*]	Porzione di programma che viene eseguita più volte
Condizione di controllo [c*ommand control*]	Stato di verifica di una condizione per determinare se una parte del programma debba essere eseguita o meno
Flusso logico [l*ogic flow*]	La struttura generale del programma. Determina la sequenza logica dei compiti in modo che il risultato sia corretto e controllato
Input da utente [u*ser input*]	Informazione fornita da una fonte esterna (di solito tastiera) mentre il programma è in esecuzione: può essere conservata e richiamata quando serve, il tipico inserimento dati da parte dell'utente
Salto condizionale [*conditional branch*]	Punto logico del programma in cui una condizione determina cosa avviene dopo e come si svolge l'esecuzione del programma
Commento o Nota o Osservazione all'interno di script bash [remark]	Il carattere # è utilizzato per inserire dei commenti all'interno di script bash $ cat commento #!/bin/bash # Questo testo non verrà visualizzato…

	echo Questo testo sarà visualizzato!!! $ ./commento Questo testo sarà visualizzato!!!

Parametri [parameter o switch]	Quando si affiancano valori da linea di comando al nome di uno script di shell, la shell li tratta come se fossero variabili $0,$1,$2,$3,… consideriamo il seguente esempio:
	$ cat hello
	#!/bin/bash
	echo Ciao $1, sei libero $2?
	$ ./hello Tizio mercoledì
	Ciao Tizio sei libero mercoledì?
	$ ./hello Sempronio domani
	Ciao Sempronio sei libero domani?
	Il $0, contiene il nome dello script, $1,$2,… rappresentano il primo parametro, il secondo parametro, eccetera, il $* contiene tutti i parametri, il $# il numero dei parametri come si può notare dall'esempio sottostante:
	$ cat parametri
	#!/bin/bash
	# Conta ed elenca i parametri
	# Uso: ./paramteri PRIMO SECONDO ….
	echo ci sono $# parametri e sono: $*
	$ ./parametri
	ci sono 0 parametri e sono:
	$ ./parametri tizio
	ci sono 1 parametri e sono: tizio
	$ ./parametri tizio caio sempronio
	ci sono 3 parametri e sono: tizio caio sempronio

Quoting (l'uso di apici e virgolette per racchiudere stringhe)	Per visualizzare una frase si possono usare indistintamente due comandi: echo "frase"

	echo 'frase'
	si usano le virgolette se la frase contiene degli apostrofi; si usa l'apostrofo se la frase contiene le virgolette. Tutto ciò che viene racchiuso tra una coppia di accenti gravi (`) da non confondere con l'apostrofo (') viene interpretato come un comando, in ambito Linux, l'accento grave si ottiene con la combinazione dei tasti AltGr + ' quello sotto il ?
	$ cat somma
	#!/bin/bash
	# Calcola la somma di tre numeri
	#Uso: ./somma NUM1 NUM2 NUM3
	echo "La somma dei tre numeri inseriti e' `expr $1 + $2 + $3`."
	$ ./somma 12 13 56
	La somma dei tre numeri inseriti e' 81.
	lasciare almeno uno spazio tra il comando expr, i parametri ($1,$2,$3) e l'operatore (+), utilizzare il comando expr per permettere la valutazione delle espressioni:
	$ cat versione
	#!/bin/bash
	# Visualizza la versione del kernel in uso
	#Uso: ./versione
	echo "La versione del kernel è la `/bin/uname -a | /usr/bin/cut -d ' ' -f3`."
	$ ./versione
	La versione del kernel è la 2.6.32-5-686.

E' possibile memorizzare quindi degli script di shell memorizzando su un file sequenze di comandi che vogliamo far eseguire. Per lanciare uno script da bash occorre però scrivere il nome del file di script passato come parametro:

$ bash miofilecomandi

In alternativa si può evitare di inserire il comando bash, seguendo il seguente procedimento:

la prima riga dello script deve essere	#!/bin/bash
lo script deve essere reso eseguibile con il comando	chmod +x miofilecomandi
per eseguire uno script digitare	./miofilecomandi

Il carattere #! si chiama “Magic Number” o anche "sha-bang". Il ./ davanti al nome dello script sta ad indicare il percorso attuale, cioè viene eseguito lo script che si trova nella directory nella quale siamo posizionati, se inseriamo lo script in una cartella tra quelle definite nella variabile ambiente PATH il file sarà eseguito indipendentemente in shell genitrici o shell figlie e senza specificare il percorso. Per distinguere immediatamente ad occhio un file di script, può essere comodo aggiungere il suffisso .sh quindi il nostro file comandi diventerà miofilecomandi.sh, in modo simile alle estensioni dei file in ambiente Dos/Windows. E’ possibile consentire la comunicazione tra script diversi (o tra script e comandi) utilizzando i comandi trap (per accettare un segnale) e kill (per inviare un segnale), questi comandi non verranno affrontati in questo testo. Ad ogni modo, non si intende realizzare un corso di programmazione in poche pagine, ma sicuramente il provare degli esempi di programmi/script già fatti, faciliterà la comprensione della programmazione degli script bash, in modo tale che si apprenda per esempi.

11.7 Input e output utente (echo, printf, read)

Per l’output su video di informazioni, variabili e messaggi in generale si è utilizzato finora il comando echo ma è possibile utilizzare anche il comando printf (noto già a chiunque abbia esperienza di programmazione in linguaggio c). Senza dilungarci ulteriormente sappiate che come per il comando echo sono disponibili le sequenze di escape qui di sotto riepilogate in tabella:

Sequenza	**Significato**
\a	Allarme (campanello)
\b	Spazio indietro o *backspace*
\c	Sopprime i nuovalinea in mezzo al testo
\e	Escape
\f	Avanzamento foglio (Form feed)
\n	Nuovalinea (a capo e inizio riga)
\r	Ritorno carrello o CR (*Carriage Return*).
\t	Tabulazione orizzontale
\v	Tabulazione verticale
\\	Barra inversa (Backslash)
NNN	Il carattere a otto bit il cui valore è il valore

	ottale NNN (da zero a tre cifre ottali).
\NNN	Il carattere a otto bit il cui valore è il valore ottale NNN (da una a tre cifre ottali).
\xHH	Il carattere a otto bit il cui valore è il valore esadecimale (una o due cifre esadecimali)

Per quello che riguarda l'input di valori da parte dell'utente, si può utilizzare il comando read, dopo la sintassi, riepilogata qui di seguito, vediamo subito un esempio perché è molto più semplice che la spiegazione teorica:

Sintassi:

read [opzioni] NOME1 NOME2 ... NOMEN

vediamo l'esempio:

```
$ cat somma2
#!/bin/bash
# Calcola la somma di due numeri letti da tastiera con input utente
sommanumeri="0"
echo -n "inserire il primo numero : "
read primo
echo -n "inserire il secondo numero : "
read secondo
sommanumeri=$[$primo + $secondo]
echo "la somma di $primo e $secondo è $sommanumeri"
$ ./somma2
inserire il primo numero : 343
inserire il secondo numero : 122
la somma di 343 e 122 è 465
```

11.8 Strutture di programmazione condizionali ([ ], test, if/elif/else/fi, case/esac)

[ ] o [[]]

Nella bash si utilizzano le parentesi quadre [ ] per racchiudere l'espressione della condizione da verificare. Sono disponibili controlli specifici per variabili numeriche, ad esempio per controllare una specifica variabile 'numero':

```
[$numero -eq 0] vero se uguale a zero (si può usare anche =)
[$numero -ne 0] vero se uguale a zero (si può usare anche !=)
[$numero -gt 0 ] vero se maggiore di zero
[$numero -lt 0 ] vero se minore di zero
[$numero -ge 0 ] vero se maggiore o uguale a zero
[$numero -le 0 ] vero se minore o uguale a zero
```

I termini “incomprensibili” indicati nell’espressione stanno a significare:

- eq = equal = uguale
- gt = greater than = maggiore di
- lt = less than = minore di
- ge = greater or equal = maggiore o uguale di
- le = less or equal = minore o uguale di

Le doppie parentesi quadre si utilizzano nel caso in cui l'espressione da valutare/confrontare contenga stringhe o nomi file contenenti degli spazi.

IF - THEN - ELIF - ELSE – FI

Sintassi:

```
if CONDIZIONE1 then LISTACOMANDI1
elif CONDIZIONE2 then LISTACOMANDI2
elif CONDIZIONE3 then LISTACOMANDI3
….
else LISTACOMANDI_N
fi
```

Si può utilizzare il comando if per eseguire uno o più comandi, a seconda del verificarsi di una condizione:

```
$  cat massimo
#!/bin/bash
# Verifica il massimo tra due numeri inseriti da tastiera
# Uso: ./massimo NUM1 NUM2
if [ $1 -gt $2 ]; then echo "Il numero $1 è più grande di $2"
else echo "Il numero $2 è più grande di $1"
fi
$ ./massimo 122 556
Il numero 556 è più grande di 122
```

In questo caso i parametri inseriti dopo il nome dello script vengono confrontati tra di loro, nello specifico, se (if) il primo parametro è maggiore (-gt) del secondo visualizza la prima echo, altrimenti (else) visualizza la seconda echo. Vediamo un esempio più complesso:

```
$ cat filelink2
#!/bin/bash
# Uso: ./filelink2 NOME1 NOME2 ...
for nomefile
do
if [ -L  "$nomefile" ]; then echo $nomefile: si tratta di un link simbolico
elif [ -d  "$nomefile" ]; then echo $nomefile: si tratta di una directory
```

```
elif [ -f   "$nomefile" ]; then echo $nomefile: si tratta di un file
else echo   $nomefile: non definito


fi
done
$   ./filelink2 prov.txt for.txt ordina.txt Immagini Musica Immagini
prov.txt:   si tratta di un file
for.txt:   si tratta di un file
ordina.txt:   si tratta di un file
Immagini:   si tratta di una directory
Musica:   si tratta di una directory
Immagini:   si tratta di una directory
```

Se il comando if segnala un “successo” (codice d’uscita 0) allora verrà eseguito il ramo then fino al prossimo elif, else oppure fi. Se la condizione non è verificata verranno eseguiti i controlli specificati nell’elif in maniera sequenziale fin quando non sarà verificata una condizione o raggiunto il fi. I comandi dopo l’else saranno eseguiti se nessuno degli if o elif ha avuto successo, gli else e gli elif non sono obbligatori, il fi è obbligatorio. Il comando for, appena incontrato, sarà spiegato tra poco.

TEST

Sintassi:

```
test [opzioni] CONDIZIONE LISTACOMANDI_CONDIZIONE_VERA
```

Un utile strumento per le scelte è il comando test il quale può verificare una grande varietà di condizioni. Esso ritorna in codice di uscita 0 (successo) se la condizione è verificata, altrimenti un valore diverso da 0 se la condizione non è verificata, vedi esempio:

```
$ cat filelink
#!/bin/bash
# Verifica se i file passati sono file, directory o link simbolici
# Uso: ./filelink NOME1 NOME2 ...
for nomefile
do
test -d "$nomefile" && echo   $nomefile: directory
test -f "$nomefile" && echo   $nomefile: file
test -L "$nomefile" && echo   $nomefile: symbolic link
done
$   ./filelink prov.txt for.txt ordina.txt Immagini Musica Immagini
prov.txt: file
for.txt: file
ordina.txt: file
Immagini: directory
```

Musica: directory
Immagini: directory

Questo script passa un elenco di nomi file come parametri e in uscita per ognuno visualizza la directory il nome file o il link simbolico. Il comando test esiste in due modalità: una come programma nella cartella /bin/test, una come comando interno in bash o altre shell. I comandi si comportano diversamente.

CASE – ESAC

Sintassi:

```
case VARIABILE in
VALORE1) LISTACOMANDI1 ;;
VALORE2) LISTACOMANDI2 ;;
……
esac
```

Quando l'utilizzo di if-elif diventa troppo complicato per dover gestire fasce di condizioni, può risultare molto comoda la struttura case-esac. Per separare molteplici condizioni si usa il simbolo "|" per terminare l'elenco dei modelli da confrontare (pattern list) si usa l'operatore ")". Ciascun case e i suoi relativi comandi vengono definiti clausola (clause). Ogni clausola va terminata con ";;". Tutte le istruzioni case terminano con l'istruzione esac. Vediamo un esempio semplice:

```
$ cat testfile
#!/bin/bash
# Verifica se i file passati sono immagini, documenti o programmi
# Uso: ./testfile
for file in `ls . ` ; do # per ogni elemento della   directory corrente
case   $file in # visualizza un messaggio appropriato
*.gif|*.jpg)   echo "$file: file grafico" ;;
*.txt|*.text|*.doc|*.rtf|*.pdf)   echo "$file: documento" ;;
*.cpp|*.c|*.f|*.for|*.sh)   echo "$file: file sorgente" ;;
*) echo "$file: file generico" ;;
esac
done
$ ./testfile
ordina:   file generico
ordina.txt:   documento
```

prov: file generico

prova.sh: file sorgente

prov.txt: documento

11.9 Strutture iterative (for/do/done, while/do/done, until/do/done)

FOR

Sintassi: for NOME [in LISTA] do LISTACOMANDI done

Se [in LISTA] non è presente for esegue COMANDI una volta per ciascun parametro posizionale che è stato impostato. Vediamo quindi come comportarsi quando occorre ripetere (iterare) una stessa operazione (comando), gli esempi sono prima bash e poi inseriti in script:

```
$ for i in 1 2 3
> do
> echo e $i ....
> done
e 1 ....
e 2 ....
e 3 ....
```

La variabile i assume ad ogni iterazione un valore diverso.

```
$ list='4 5 6'
$ for i in $list
>  do
>  echo e $i ....
>  done
e  4 ....
e  5 ....
e  6 ....
```

Se si omette la parola chiave “in…” il ciclo viene ripetuto in base ai parametri della linea di comando. Con il comando for il numero di iterazioni (le volte che si deve ripetere l’operazione) è fissato all’inizio, ma può capitare che non sia chiaro quante volte deve essere ripetuta un’operazione o addirittura che si debba terminare un intero ciclo al verificarsi di una condizione o un evento, vediamo un esempio:

```
$ cat backuptxt
#!/bin/bash
# Copia tutti i   file txt nella cartella backup
# Uso: backuptxt
rm -rf backup; mkdir backup
for f in *.txt
do
cp "$f" backup
done
$ ./backuptxt
```

Non si conoscono a priori quanti sono i file *.txt presenti nella directory, ma verranno comunque copiati tutti nella cartella backup. Si possono utilizzare anche gli operatori && e || in modo da eseguire comandi solo in specifiche circostanze, vedi sotto l'esempio precedente modificato:

```
$ cat backuptxt2
#!/bin/bash
# Copia i file txt   nella cartella backup
# Uso: backuptxt2 PAROLA_CONTENUTA_NEI_TXT
rm -rf backup; mkdir backup
for f in *.txt
do
grep $1 "$f" && cp "$f"   backup
done
$ ./backuptxt2 casa
```

In questo caso la copia dei file viene effettuata solo per i file in cui termina per file .txt che contengono una specifica parola al loro interno ad esempio "casa".

WHILE

Sintassi: while CONDIZIONE do LISTACOMANDI done

CONDIZIONE può essere qualsiasi comando che può uscire con uno stato di successo o fallimento. LISTACOMANDI può essere qualsiasi sequenza di comandi o di programmi o script di shell. Non appena CONDIZIONE è falsa, si esce dal ciclo e viene eseguito il comando successivo all'istruzione done. La CONDIZIONE di while è simile a quella dei costrutti if ma si permette l'esecuzione ripetuta di LISTACOMANDI finchè CONDIZIONE venga verificata con successo (stato di uscita zero).

Vediamo un esempio:

```
$ cat media
#!/bin/bash
# Calcola la media   di una serie di numeri.
# Uso: ./media
punti="0"
media="0"
somma="0"
i="0"
clear
while true; do
echo   "Valore n.$[$i+1]"
echo -n "Inserisci   punteggio percentuale [0-100%] ('x' per finire): "
read punti
if   (("$punti" < "0")) || (("$punti" >   "100")); then
echo "Errore:   il punteggio deve essere compreso tra 0 e 100!"
elif [ "$punti" == "x"   ]; then
echo "Punteggio percentuale medio:   $media%."
break
else
somma=$[$somma + $punti]
i=$[$i + 1]
media=$[$somma / $i]
fi
done
```

In questo caso la CONDIZIONE è sempre verificata (true), infatti il carattere ‘x’ è utilizzato per terminare l’inserimento dei punteggi perché non è noto a priori quanti valori inseriremo. Per uscire dal ciclo while in questo caso si è utilizzato il comando break, che interrompe il ciclo quando all’inserimento da tastiera del carattere ‘x’.

```
$ cat finestre
#!/bin/bash
```

```
# Questo script apre  un certo numero di finestre di terminale specificato come parametro di input.
# Uso: ./finestre  NUMERO
i="0"
while [ $i -lt $1 ]
do
xterm &
i=$[$i+1]
done
$ ./finestre 6
```

La condizione [$i -lt $1] sta a significare “mentre i è minore del numero dato come parametro di input”, nell’esempio verranno aperte 6 finestre terminale.

UNTIL

Sintassi: until CONDIZIONE do LISTACOMANDI done

Lo stesso effetto del programma precedente, si può ottenere modificando lo stesso nel seguente modo:

```
$ cat finestre2
#!/bin/bash
# Questo script apre un certo numero di finestre di terminale.
# Uso: finestre2 NUMERO
i="$1"
until  [ $i -lt 1 ]
do
xterm &
i=$[$i-1]
done
$ ./finestre 6
```

In questo caso l’ “until” si comporta in modo opposto al while, cioè ripete le operazioni del ciclo “fin quando i non è minore di 1”, chiaramente la variabile contatore i parte dal valore massimo delle finestre, per essere decrementata fino ad arrivare ad 1. L’effetto finale sarà lo stesso.

11.10 Collaudo degli script

Per collaudare gli script di shell sono disponibili due interessanti opzioni:

-v, modalità verbose: visualizza ogni comando prima di eseguirlo

$ sh -v ./somma 1 2 3

#!/bin/bash

echo "la somma dei tre numeri e' `expr $1 + $2 + $3`"

la somma dei tre numeri e' 6

-x, modalità execute: visualizza ogni comando eseguito con le relative variabili ad esso associate

$ sh -x ./somma 1 2 3+ expr 1 + 2 + 3

+ echo la somma dei tre numeri e' 6

la somma dei tre numeri e' 6

ESERCIZI

1. Creare 4 variabili, VAR1, VAR2, VAR3, VAR4 e inizializzarle rispettivamente con i valori "prima”,”centoquattordici", "12" e "Buon compleanno"
2. Visualizzare i valori inseriti in tutte le variabili.
3. Rimuovere VAR1 e VAR3.
4. Creare un nuovo file VARIABILI.SH con editor NANO e aggiungere come commenti allo script: DESCRIZIONE, USO, AUTORE e DATA
5. Cambiare i permessi allo script in modo da poterlo eseguire.
6. Lo script dovrà visualizzare il nome di tutte le variabili di shell ed il loro contenuto.
7. Eseguire lo script in modalità normale ed in modalità di correzione.
8. Creare uno script RETTANGOLO.SCRIPT che permetta l’input di due valori corrispondenti alle misure dei lati di un rettangolo e li inserisca in due variabili. Calcolare perimetro ed area del rettangolo che ha le misure inserite. Utilizzare variabili con nomi auto-esplicativi e corredare lo script di commenti.
9. Creare un script ETA.ETA che permetta l’input di un valore corrispondente all'età dell'utente. Se è uguale o maggiore di 18 stampare un messaggio che dice che l’utente è maggiorenne e che può guidare un’automobile. Se l'età dell'utente è inferiore a 18 stampare un messaggio che informi l'utente tra quanti anni potrà guidare l’automobile. Utilizzare variabili con nomi auto-esplicativi e corredare lo script di commenti.
10. Creare uno script PARAMETRI che riceve due parametri $1 e $2 che corrispondono a due numeri, verificare se i due numeri sono uguali o uno

maggiore dell'altro e visualizzare un messaggio per l'utente. Utilizzare variabili con nomi auto-esplicativi e Corredare lo script di commenti.

SOLUZIONI

1.

$ VAR1="prima"

$ VAR2="centoquattordici"

$ VAR3="12"

$ VAR4="Buon compleanno"

2.

$ set

3.

$ unset VAR1

$ unset VAR3

4.

$ nano VARIABILI.SH

Appena caricato l'editor digitare:

#!/bin/bash

Descrizione: Questo script fa questo, questo e qust'altro…

Uso: ./VARIABILI.SH

Autore: Pinco Pallino

Data: 27/08/2012

…..il resto del codice

5.

$ chmod +x VARIABILI.SH

6.

#!/bin/bash

Descrizione: Questo script fa questo, questo e quest'altro…

Uso: ./VARIABILI.SH

```
# Autore: Pinco Pallino

# Data: 27/08/2012

echo Visualizzazione delle variabili e loro contenuto. Premere INVIO per scorrere l'elenco.

set | more
```

7.

Modalità normale:

```
$ ./VARIABILI.SH
```

Modalità correzione:

```
$ sh –x ./VARIABILI.SH
```

Ma in questo esercizio l'output sarà simile…

8.

```
$ nano RETTANGOLO.SCRIPT
```

Appena caricato l'editor digitare:

```
#!/bin/bash

# Descrizione: Calcola perimetro e area di un rettangolo i cui lati sono letti da tastiera con input utente

# Uso: ./RETTANGOLO.SCRIPT

# Autore: Pinco Pallino

# Data: 01/09/2012

# azzero le variabili

perimetro="0"

area="0"

# leggo da tastiera il valore dei lati

echo -n "inserire il primo lato : "

read primolato

echo -n "inserire il secondo lato : "

read secondolato

# calcolo perimetro ed area

perimetro=$[$primolato * 2 + $secondolato * 2]

area=$[$primolato * $secondolato]
```

ISBN: 9788866850090

```
# stampo i risultati
echo "Il perimetro del rettangolo 2*$primolato + 2*$secondolato è: $perimetro"
echo "L'area del rettangolo $primolato x $secondolato è: $area"
```

salvare ed impostare il permesso per l'esecuzione:

```
$ chmod +x RETTANGOLO.SCRIPT
$ ./RETTANGOLO.SCRIPT
```

9.

```
$ nano ETA.ETA
```

Appena caricato l'editor digitare:

```
#!/bin/bash
# Descrizione: Verifica in base all'input utente se è maggiorenne e può guidare una automobile o quanti anni mancano per prendere la patente
# Uso: ./ETA.ETA
# Autore: Pinco Pallino
# Data: 01/09/2012
# azzero la variabile
differenza="0"
# leggo da tastiera l'età dell'utente
echo -n "inserisci la tua età : "
read eta
# verifico se l'età è maggiore o minore di 18 e stampo un messaggio
if (("$eta" >= "18")) ; then
echo "Complimenti puoi guidare l'automobile perchè hai $eta anni!"
else
differenza=$[18 - $eta]
echo "Mi dispiace ti mancano ancora $differenza anni per guidare l'automobile"
fi
```

salvare ed impostare il permesso per l'esecuzione:

```
$ chmod +x ETA.ETA
$ ./ETA.ETA
```

10.

$ nano PARAMETRI

Appena caricato l'editor digitare:

```
#!/bin/bash
# Descrizione: Verifica il massimo tra due numeri inseriti come parametri
# Uso: ./PARAMETRI NUMERO1 NUMERO2
# verifico se per sono stati o meno inseriti i parametri, nota che nel caso ne fossero inseriti più di due sarebbero comunque considerati solo i primi due
if (($# == 0)) ; then
echo "Non hai inserito i parametri!"
echo "Uso: ./PARAMETRI NUMERO1 NUMERO2"
else
echo "Primo numero = $1"
echo "Secondo numero = $2"
if (("$1" > "$2")) ; then
echo "Il numero $1 è più grande di $2."
elif (("$1" < "$2")) ; then
echo "Il numero $1 è più piccolo di $2."
else
echo "I due numeri sono uguali"
fi
fi
```

salvare ed impostare il permesso per l'esecuzione:

```
$ chmod +x PARAMETRI
$ ./PARAMETRI
```

Parte 4:
Il sistema operativo Linux

ithum®

12 Scegliere un Sistema Operativo

Chi legge questo libro, probabilmente, ha un'idea di cosa sia una Sistema Operativo, in inglese Operating System o più semplicemente OS, e quali funzioni svolge all'interno di un computer.

Analogamente all'abbreviazione inglese OS verrà usata la versione italiana SO .

In questa sezione si cercherà di dare una visione sistematica dei compiti del SO, analizzando le funzioni generali di ogni sistema operativo per poi entrare nel dettaglio delle implementazioni dei SO più diffusi, dei quali , al fine di scegliere un SO,verranno messe a confronto le rispettive implementazioni. I sistemi operativi analizzati sono: linux, OS/X e WINDOWS, chiaramente, maggiore attenzione verrà data a Linux. Non mancheranno accenni ad altri SO come BSD.

12.1 Cenni generali sui Sistemi Operativi

Il Sistema Operativo si occupa di molte operazioni che normalmente facciamo quando siamo davanti a un computer: copiare file, avviare/installare programmi, collegare periferiche come stampanti o pen drive, sono solo alcune delle attività che gli competono.

All'interno della macro divisione di un computer in hardware, che sarà oggetto della sezione successiva, e software, il SO fa parte della seconda e funge da collettore tra le due. In pratica tutti gli altri software installati, come editor, player, browser, etc., comunicheranno con l'hardware solo e soltanto attraverso le funzioni messe a disposizione dal SO. In realtà neanche il SO lavora sempre direttamente con l'hardware, esso utilizza, ad es., i driver per la comunicazione con le periferiche e il BIOS per la comunicazione con diverse componenti della scheda madre. Il Sistema Operativo è, dunque, un software indispensabile per un computer.

12.2 Componenti di un Sistemi Operativo: il kernel

Un sistema operativo, per le funzioni che deve svolgere, è un software molto complesso. In realtà è composto da molti programmi, ciascuno dei quali svolge una particolare funzione.

Alcune funzioni sono basilari per il funzionamento del computer sia per la frequenza di utilizzo che per la criticità, un loro problema comprometterebbe l'utilizzo del computer stesso. Tali funzioni costituiscono il *kernel* , in italiano *nucleo,* di un SO. In particolare le funzioni svolte sono:

1. gestione del processore o dei processori: il SO assegna ai programmi in esecuzione i processori disponibili sul computer, tutti i programmi, ciascuno con la propria priorità, devono avere accesso al processore per poter svolgere la propria funzione.
2. gestione della memoria centrale: ad ogni programma in esecuzione il SO assegna una o più aree di memoria centrale, necessarie a contenere sia le istruzioni dei programmi in esecuzione che i valori da elaborare (i dati).
3. comunicazione con l'hardware: il SO fa da tramite tra i programmi e le periferiche come mouse stampanti ecc. In realtà il SO non comunica direttamente con le periferiche ma mediante i driver, programmi che sono legati strettamente alla periferica di cui ne conoscono le specifiche di funzionamento ed utilizzo. Il driver, chiaramente, è strettamente legato al modello della periferica ma anche al SO. I driver assieme ai SO formano il cosiddetto *software di base*, proprio per l'importanza della funzione svolta. Tutti i SO contengono al loro interno i driver delle periferiche più comuni come tastiere, mouse, pen drive ecc, e richiedono il driver, invece, per le altre.
4. interazione tra programmi: tutti i programmi utilizzati possono comunicare scambiandosi dati o richiamando funzioni contenute in altri programmi, compreso il SO stesso..

Ciascun SO ha un proprio kernel che svolge le funzioni prima elencate, ma nonostante le funzioni siano le stesse non è possibile scambiare i kernel di diversi SO in quanto ciascun

kernel svolge il proprio lavoro in modo diverso dagli altri. Per lo stesso motivo ciascun programma può funzionare solo con un tipo di kernel e quindi un programma per linux non può utilizzato anche con altri SO. In definitiva un kernel stabilisce i programmi che possono essere utilizzati e l' hw sul quale può essere utilizzato.

Anche se alcuni programmi, come LibreOffice, li troviamo su diversi SO, questo non significa che è lo stesso programma che gira su più SO. In questi casi chi ha scritto il programma crea diversi file eseguibili per i diversi sistemi operativi utilizzando librerie e programmi aggiuntivi diversi per ogni SO.

Nella terminologia comune si fa riferimento a Linux come un sistema operativo, ma, in realtà, è solo il kernel. Per avere un Sistema Operativo completo sono stati aggiunti dei programmi che vanno sotto il nome di sistema *GNU*, creato da *Richard Stallman*. Per cui il nome corretto, usato in pochissimi casi, è GNU-Linux. Gnu sta per “GNU's Not Unix” (GNU Non è Unix) e si pronuncia g-nu con la g muta senza la 'i'.

Il primo kernel Linux è stato creato da Linus Torvalds nel 1991, quando era ancora uno studente all'università di Helsinki. Da allora il kernel Linux si è evoluto e al suo sviluppo partecipano molti programmatori sparsi per il mondo.

Il nome Linux è derivato dalla contrazione del nome dell'autore (Linus) e del nome del sistema operativo su cui si basa (Unix).

Sia Linux che GNU aderiscono al progetto di software libero e rientrano, in modo diverso, nel tipo di licenza GPL, General Public License, che offre delle garanzie all'utente come libertà di utilizzo, copia, modifica e distribuzione del software utilizzato.

Oggi Linux viene utilizzato su una grande quantità di dispositivi che vanno dai computer, ai cellulari, ai super-computer, ai tablet, ai dispositivi come router e altre apparecchiature di rete.

12.3 Oltre il kernel

Nel paragrafo precedente è stato trattato il nucleo di un sistema operativo, il fulcro che svolge le funzioni fondamentali di un computer. Il SO però svolge anche altre funzioni come:

12.3.1 Interfaccia utente a linea di comando

E' la modalità di interazione tra utente e computer utilizzata con i primi computer, è composta da programmi che accettano comandi esclusivamente da tastiera, rappresentano un'alternativa alla più utilizzata interfaccia grafica. Tutti i programmi utilizzati in un computer possono essere richiesti, oltre che da interfaccia grafica, anche da un'interfaccia a linea di comando, chiaramente la linea di comando richiede una maggiore conoscenza ed esperienza ma esistono dei vantaggi nell'utilizzare la linea di comando come la velocità e la possibilità di utilizzo in caso di indisponibilità dell'interfaccia grafica.

In Linux tali interfacce prendono il nome di *shell,* i comandi disponibili in una shell saranno trattati nella relativa sezione.

12.3.2 Interfaccia utente grafica o GUI

La GUI, Graphical User Interface, è la modalità di interazione tra utente e computer oggi più diffusa in quanto più semplice ed intuitiva. La comunicazione con l'utente oltre che con la tastiera avviene con un uso avanzato del mouse, di icone e menu.

I più diffusi sistemi operativi mettono a disposizione dei loro utenti entrambi i tipi di interfaccia, a differenza di MAC OS e WINDOWS,che presentano un solo tipo di interfaccia grafica, in linux sono disponibili diverse interfacce grafiche.

Mentre in MS-WINDOWS, l'interfaccia è parte integrante del SO stesso, in Linux è un programma separato. Questo ha facilitato lo sviluppo, sotto Linux, di diverse interfacce grafiche come GNOME, KDE, XFCE, quest'ultima particolarmente adatta per computer poco performanti.

In realtà in Linux la GUI è composta da diversi sw, un server che si occupa della visualizzazione del contenuto di una finestra e dell'interazione con l'utente al suo interno (INPUT e OUTPUT), chiamato server X, da un Windows Manager che si occupa della gestione delle finestre, e da un Desktop Manager che altro non è che un insieme di programmi e librerie che servono a gestire barre degli strumenti, sfondi e widget per il desktop. E', comunque, possibile installare più interfacce grafiche e scegliere quella che si vuole utilizzare al momento della fase del login grafico, è possibile anche escludere dall'installazione o dalla fase di avvio del computer il caricamento dell'interfaccia grafica.

Le interfacce grafiche più utilizzate sono:

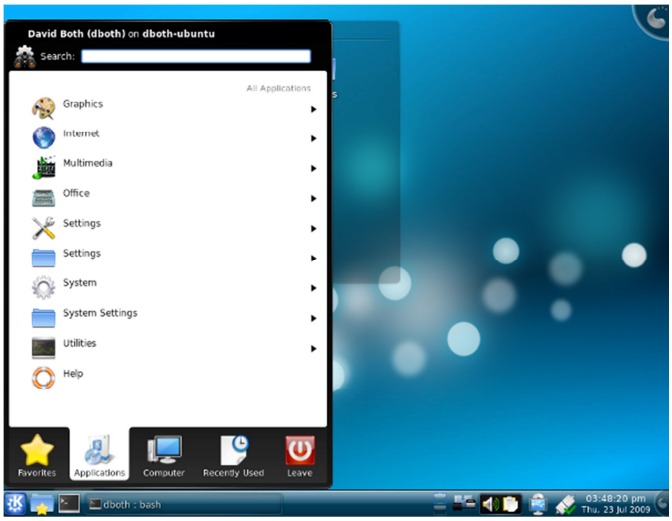
- *KDE*, the K Desktop Environment, è un'interfaccia grafica utilizzata di default su sistemi come SUSE e Mandriva e basata sulla libreria Qt, una libreria grafica molto utilizzata sotto linux. KDE è caratterizzata da una ricca raccolta di software e da un'aspetto grafico avanzato che, per certi aspetti, la rendono simile ad altri sistemi operativi come OS X e WINDOWS.
- *GNOME*: The GNU Network Object Model Environment , utilizzata di default da Debian e RED HAT, è basata su un'altra libreria grafica molto diffusa la GTK. Gnome ha, rispetto a KDE, una grafica più essenziale ma più semplice da utilizzare.
- *LXDE*: The Lightweight X11 Desktop Environment sviluppata per utilizzare poche risorse di sistema e quindi orientata su computer datati.
- *Unity*, di CACONICAL la società che si occupa della distro UBUNTU, come Gnome si pone l'obiettivo di essere facile da utilizzare ed è orientata ad utenti completamente inesperti. Nelle due ultime versioni sono state apportate modifiche per facilitare il passaggio verso sistemi Android.
- *XFCE*: anch'essa basata sulla libreria GTK e tende a consumare poche risorse di sistema.
- Anche se ogni interfaccia grafica può usare programmi diversi è possibile installare programmi caratteristici di KDE, ad esempio, in un ambiente Gnome,

questo comporta, chiaramente, anche l'installazione delle relative librerie su cui è basato.

L'esecuzione dei programmi può avvenire in modi diversi che vanno dall'utilizzo del classico menu suddiviso in categorie come avviene per KDE FOTO al doppio click sui collegamenti a programmi presenti sul desktop. E' possibile lanciare un programma anche da desktop. Un'altra modalità, usata da Gnome e unity, è quella del lancio attraverso i pannelli che sono elenchi di icone posizionati sui bordi delle finestre e che danno accesso ai programmi. Infine tutti gli ambienti grafici danno la possibilità di inserire il nome del programma desiderato in una casella di ricerca e quindi consentono di risalire facilmente al programma.

Ecco un esempio di unity: scrivendo term, volendo aprire un terminale, mi fa vedere l'icona di 3 diversi programmi che posso utilizzare.

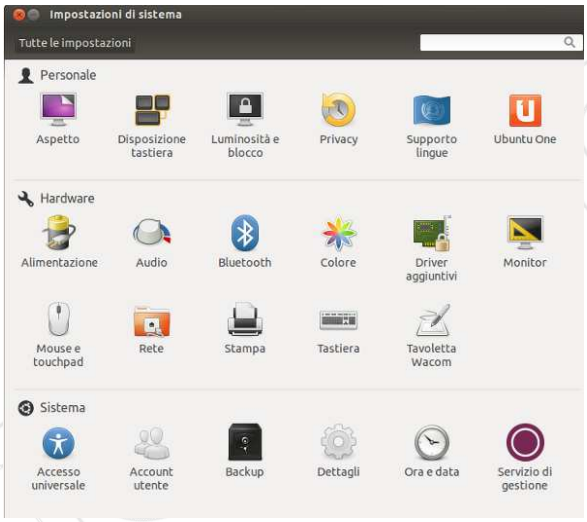
Questa, invece, è un'immagine relativa al menu di KDE:



12.3.3 Utility

I moderni sistemi operativi mettono a disposizione degli utenti una grande varietà di programmi che hanno gli scopi più disparati, vanno dalla gestione del disco a quella dello schermo, alle impostazioni della batteria del computer, ai compilatori fino ai giochi e molti altro ancora. Anche in questo caso, pur essendo simili le funzioni, i programmi cambiano con il SO ma è abbastanza semplice ritrovare tali funzioni nei menu disponibili o con il menu contestuale (quello ottenibile con il tasto destro su un oggetto).

Queste funzioni, solitamente, sono disponibili nella voce Sistema o Impostazioni di sistema., Nel caso di Unity si apre questa finestra:



che consente l'impostazione dello schermo piuttosto che il controllo della lingua che il backup del sistema.

12.3.4 Distribuzioni linux, formati dei pacchetti e repository

Linux, tra le altre cose, si differenzia dagli altri sistemi operativi per la sua modalità di acquisizione. Difficilmente, infatti, abbiamo a disposizione il solo GNU-Linux, cioè il nucleo del sistema operativo con tutte le funzioni messe a disposizione dal software GNU, ma abbiamo a disposizione dei set completi che oltre al sistema operativo di base contiene anche tutta una serie di software a corredo che vanno dai tool di installazione ai programmi di produttività individuale come LibreOffice, a programmi di grafica come GIMP a programmi di masterizzazione. Tutti questi programmi formano una 'distribuzione linux' o 'distro'. Di distro ne esistono molteplici, solo per ricordarne alcune Debian, Red Hat, UBUNTU, openSUSE e Mandriva. Molte di queste sono disponibili anche in modalità live, non necessitano di installazione su hard-disk ma vengono avviate direttamente da cd-rom e mettono a disposizione, nel giro di qualche minuto, una macchina linux con tutti i programmi a corredo.

Le distribuzioni Linux solitamente mettono a disposizione dei repository cioè archivi online di software dal quale prelevare programmi per un'installazione semplificata. Ogni distribuzione mette a disposizione i propri repository, i programmi messi a disposizione possono essere in un formato già compilato sotto forma eseguibile da installare oppure in formato sorgente, per la cui installazione bisogna prima compilare il programma. Distribuzioni come slackware preferiscono quest'ultima soluzione, nella maggior parte dei casi, invece, troviamo dei programmi già compilati che possono essere di due tipi diversi .rpm , usato ad es da RedHat, SUSE, Mandriva e .deb usato da Debian e Ubuntu. Esistono dei tool che consentono di installare pacchetti .deb su distro che non utilizzano tale formato di installazione e viceversa.

12.4 Ciclo di vita delle distribuzioni

Linux

Linux è attualmente alla versione 3.5, rilasciata nel Luglio 2012. dal Luglio 2011 vengono utilizzate due cifre per indicare una versione stabile, mentre alle versioni ancora in fase di sviluppo viene aggiunto un suffisso tipo -rc1 (release candidate 1), rc2 ecc.

Per lungo tempo linux ha utilizzato, invece, un sistema di numerazione a tre cifre sempre separate dal punto, la prima cifra era la versione del kernel incrementata solo in caso di mutamenti radicali, poi c'era la major release incrementata ogni volta che c'erano novità importanti, un numero pari indicava una versione stabile mentre dispari una versione di sviluppo, infine la terza cifra o minor release che veniva incrementata in caso di correzioni o miglioramenti non significativi.

Distribuzioni

La cadenza delle distribuzioni cambia per ciascuna di esse, così come il periodo di supporto che garantisce aggiornamenti e applicazioni di patch. La distro Debian è caratterizzata da un lungo intervallo riguardo al tempo di rilascio, questo per garantire un sistema stabile per un periodo di tempo più o meno lungo, chiaramente non è apprezzato da chi vuole essere aggiornato molto frequentemente. Ubuntu rilascia una versione ogni 6 mesi, più precisamente in aprile e in ottobre di ogni anno. La versione di Aprile del 2012 si chiama 12.04, 12 per l'anno e 04 per il mese. Ogni due anni Ubuntu rilascia una versione LTS (long term Support) che garantisce aggiornamenti per cinque anni, mentre per le altre versioni ne vengono garantiti solo due. Solitamente il rilascio di una distro stabile viene anticipata dal rilascio di una distro versione alfa, soggetta ancora a modifiche e da una successiva versione beta sulla quale vengono fatti ancora interventi per risolvere eventuali problemi prima del definitivo rilascio.

12.5 Differenze Linux, Windows, Mac OS

Delle caratteristiche di Linux se ne è parlato in precedenza, vediamo adesso le caratteristiche di altri sistemi operativi.

12.5.1 Windows.

Windows è il sistema operativo più diffuso al mondo. Il suo successo è dovuto anche al fatto che la quasi totalità dei computer che acquistiamo presenta una copia già installata di Windows. E' possibile anche acquistare un computer senza sistema operativo (quando è indicata la dicitura FreeDOS si intende questo) ma richiede una ricerca approfondita magari su web e non tutti i modelli sono disponibili in questa versione.

Windows è nato con lo scopo di dare un' interfaccia grafica al DOS, sistema operativo a linea di comando che ha fatto la storia nell'era dei personal computer.

Windows si è sviluppato molto negli anni 90 con le versioni WINDOWS 95 e WINDOWS 98, ha continuato poi con WINDOWS 2000 e la versione MILLENNIUM, quest'ultima molto discussa per i suoi problemi. Ha continuato poi con le versioni utilizzate ancora oggi di WINDOWS XP, WINDOWS VISTA, WINDOWS 7 ed è prevista per la fine del 2012 WINDOWS 8. Parallelamente alle versioni menzionate per personal computer si è sviluppata una versione per sistemi server come WINDOWS 2000 Server, 2003 Server e 2008 Server.

Essendo il SO più diffuso ha tutte le caratteristiche di base richieste da un moderno sistema, un'interfaccia semplice, una larga disponibilità di driver per periferiche.

Windows è un software di tipo proprietario, bisogna acquistarlo, è vietata la duplicazione o l'installazione della stessa licenza su più computer ed inoltre non è disponibile il codice sorgente scritto per ottenere il programma.

12.5.2 OS X

Con il termine MAC OS si indica il sistema operativo per computer Macintosh della Apple. Oggi la versione rilasciata è prende il nome di OS X. Il nucleo di tale sistema operativo, basato sul sistema operativo BSD, un altro sistema operativo basato su unix ma rilasciato con una licenza molto più permissiva di quella di linux, è rilasciato in licenza open source, mentre l'interfaccia grafica è di tipo proprietaria. Questo fa si che Mac OS dispone di un'interfaccia a linea di comando molto simile a quella di Linux. Un altro vantaggio derivante dalla struttura di Mac OS è che alcuni programmi a linea di comando, per poter essere utilizzati, hanno bisogno della sola ricompilazione.

Per quanto riguarda i programmi che usano l'interfaccia grafica non è altrettanto semplice usare quelli di lunx o di BSD, l'operazione di porting da un sistema all'altro è molto complessa. Nelle attuali versioni di MAC OS X, la X sta per il numero dieci ed indica la versione del sistema operativo a sua volta suddivisa nelle versioni 'snow leopard', 'lion', ecc. In OS X è presente una versione del server grafico X utilizzato da linux che consente l'esecuzione di programmi senza richiedere alcuna compilazione.

La licenza rilasciata da Apple vieta l'installazione dei sistemi operativi su hardware non marchiato Apple, questo vale sia per i pc che per dispositivi come iPad o iPhone. Per questi ultimi dispositivi è stata creata una particolare versione id OS X chiamata iOS. Questa è una delle principali differenza tra OS X e linux: il primo, infatti, è fatto per funzionare solo su un ristretto numero di dispositivi hardware rilasciati e certificati da Apple, mentre il secondo è fatto per poter essere utilizzato sui dispositivi più disparati.

12.5.3 Windows VS Linux

Di seguito vengono elencate le differenze tra Windows e Linux.

Licenza: Windows è rilasciata sotto licenza proprietaria o commerciale le cui caratteristiche sostanzialmente possono essere concentrate in:

- bisogna acquistare una licenza per ogni dispositivo
- divieto di diffusione e di copia del sw se non per motivo di back-up
- divieto di modifica del codice
- impossibilità di accesso al codice sorgente

per Linux che è rilasciato sotto licenza GPL, le caratteristiche sono opposte, esso e possono essere sintetizzate in:

- possibilità di duplicare e redistribuire il codice
- accesso al codice sorgente scritto per ottenere il programma
- possibilità di modificare il codice sorgente

Il vantaggio principale per l'utilizzatore comune derivante dalla licenza GPL, simile a quella OPEN SOURCE, non è tanto la possibilità di poter modificare e divulgare il sorgente modificato ma quello di avere la garanzia che il codice non contenga “altre funzioni” oltre a quelle necessarie al funzionamento del programma.

Costo: Windows essendo software proprietario è esclusivamente rilasciato in seguito all'acquisto dello stesso, Linux, invece, è disponibile in molte versioni in modo gratuito. C'è da dire a tal proposito, che l'assistenza richiesta da un sistema linux è più costosa rispetto a quella necessaria per un sistema Windows, questo è dovuto probabilmente alla minore divulgazione del sistema Linux. Per avere una idea del reale costo bisogna considerare l'intero costo o TOC (Total Cost of Owner-ship) che tiene conto sia del costo dell'acquisto iniziale che di quello di gestione e manutenzione. Esistono sul web molti confronti a proposito, alcuni a favore di Windows ma molti a favore di Linux.

Compatibilità hardware: i driver rilasciati dai produttori di hardware garantiscono l'utilizzo dell'hardware con un determinato sistema operativo. La maggior parte dei produttori hardware rilascia i driver per windows, anche se si sta diffondendo il rilascio di versioni di driver per OS X e Linux. La gestione dei driver per Linux è demandata alla miriade di sviluppatori della comunità open source che provvede alla scrittura ed al rilascio dei driver per nuovi dispositivi. Questo comporta un ritardo nel rilascio dei driver. Il problema dei driver esiste anche per i sistemi Windows, spesso non si possono utilizzare periferiche come scanner o stampanti di qualche anno prima solo perchè il produttore non produce più la periferica e quindi non rilascia il driver per una nuova versione di windows. Le ultime versioni di Windows, da parte loro, spesso non sono in grado di utilizzare i driver per versioni precedenti dello stesso SO. Diversamente da Windows, Linux tende a mantenere i driver meno recenti consentendo l'utilizzo di una periferica per un periodo più lungo.

Programmi disponibili: spesso è uno dei motivi di scelta di un sistema operativo. Esistono programmi disponibili solo su Windows, come la suite Office prodotta dalla stessa Microsoft, e altri programmi disponibili su entrambi i sistemi. Molti dei programmi scritti per Linux sono stati portati poi in ambiente Windows, tra questi troviamo LibreOffice, VLC e Gimp. Diversi programmi server che hanno avuto ed hanno ancora un grande successo sono stati portati sotto windows, tra questi ricordiamo Apache e Squid. I programmi in comune sono molti, esistono, inoltre, molti elenchi di programmi sostitutivi da usare nel caso in cui alcuni programmi sono disponibili su un sistema e non nell'altro.

Sono disponibili, infine, in entrambi i sistemi degli emulatori di ambiente che consentono di eseguire direttamente un programma Windows in ambiente Linux e viceversa.

Interfaccia utente: l'aspetto grafico delle finestre è un fatto soggettivo, a questo proposito possiamo dire che windows come OS X ha una sola interfaccia grafica mentre con Linux abbiamo la possibilità di scelta tra diversi ambienti grafici e passare da GNOME che ha un aspetto essenziale ma molto stabile, a KDE che ha delle funzioni grafiche e animazioni più avanzate a XCFE che richiede una limitata capacità di elaborazione.

Configurazione: sebbene Windows disponga di molti strumenti di configurazione del sistema, il fatto che Linux è di tipo open source garantisce la disponibilità di un maggior numero di strumenti di configurazione e la possibilità di una impostazione a qualsiasi livello, anche personalizzata.

Sicurezza: la sicurezza di un SO può essere vista sotto due aspetti, quello dell'affidabilità che riguarda la probabilità di andare in blocco durante il normale utilizzo del computer perchè esistono bug di sistema o incompatibilità tra i diversi sw installati o tra sw e hw. Il secondo aspetto riguarda invece la sicurezza derivante dai programmi malevoli come virus, trojan ecc.

Per quanto riguarda il primo aspetto ciascun sostenitore tende a privilegiare il proprio SO, mentre per quanto riguarda il secondo aspetto bisogna dire che la stragrande maggioranza dei programmi malevoli sono scritti per funzionare esclusivamente sotto Windows e non con Linux O OS X.

Sebbene nel confronto fatto in precedenza, ma anche in tutti quelli che possiamo trovare in rete non ci sia una netta differenza tra i diversi SO, Windows continua a mantenere il predominio nel mondo dei SO per pc. Le motivazioni sicuramente sono diverse e tra queste sicuramente il fatto che troviamo Windows già preinstallato sulla quasi totalità dei computer, e il fatto che molte persone conoscono già Windows e sono restie a cambiarlo.

Per quanto riguarda il mondo dei server, la situazione è invertita in quanto la maggioranza dei server web, mail, file server presenti in Internet sono basati su Linux.

13 Comprendere l'hardware del Computer

Linux, come tutti i sistemi operativi, gestisce tutte le risorse hardware di un computer garantendone il corretto utilizzo da parte dei programmi. Linux, dunque, è un gestore delle risorse e delle funzionalità messe a disposizione dell'hardware. Una conoscenza approfondita delle caratteristiche hardware disponibili oltre che del sistema operativo fanno sì che quest'ultimo possa sfruttare al massimo tutte le potenzialità messe a disposizione dell'hardware.

13.1 Il Processore

Il processore, Central Processing Unit, con acronimo CPU, è il motore del computer. Spesso indicato con il termine di microprocessore in conseguenza della sua tecnologia di costruzione caratterizzato da componenti di dimensione micron, ha il compito di eseguire tutte le operazioni necessarie per il funzionamento del computer. Fanno eccezione le istruzioni di visualizzazione che, nei moderni computer, vengono delegate al processore grafico o GPU (Graphics Processing Unit) presente sulla scheda grafica.

Nota bene: in alcuni processori moderni, la GPU è direttamente integrata nel core del processore.

Ogni CPU è in grado di eseguire delle istruzioni scritte in linguaggio binario. Le istruzioni possono essere eseguite in modo diverso a seconda del tipo di processore. Ciò che rende "compatibili" diversi processori è il fatto che mettono a disposizione dei gruppi di istruzioni (set di istruzioni) che garantisce l'interpolabilità tra processori che appartengono alla stessa famiglia, ovvero alla stessa architettura.

Una delle architetture più diffuse è sicuramente x86. Nato con i processori Intel 8086, è stata adottata e e poi mantenuto anche da altre aziende quali AMD, CYRIX e VIA.

L'architettura x86 è fra le più diffuse tra il mercato dei Personal Computer sin dagli anni 80. Il primo processore 8086 è nato nel 1981. I processori nati successivamente contenevo al suo interno il set di istruzioni base tipiche del 8086 (mantenendo così un certo grado di retro-compatibilità), affiancandole, però, con nuove istruzioni. In questo modo si garantisce, insieme all'innovazione, un certa stabilità con il vecchio software.

L'architettura x86-64 (conosciuta anche AMD64, dal nome del produttore che prima la introdusse), è l'evoluzione dell'architettura x86. A differenza di quanto molti credono, l'architettura x86-64 non è un ambiente puro a 64 bit, bensì un'evoluzione dell'architettura x86 a cui sono state aggiunti un set di istruzione a 64 bit, con conseguente adeguamento dei registri.

Il numero di bit di un processore, definisce la dimensione dei dati gestibile in un'unica operazione. Per esempio, un processore a 32 bit, per ogni ciclo di clock, potrà sommare al massimo un numero a 32 bit. Se i numeri da sommare superassero tale soglia,

l'operazione verrà scomposta in più operazioni ed eseguita in più cicli dal processore. Da ciò si deduce come i processori con architettura a 64 bit siano più prestanti dei loro predecessori a 32 bit. Non essendo di facto una nuova architettura, l'x86-64 mantiene la compatibilità con l'architettura x86 ed è, dunque, in grado di eseguire anche codice a 32 bit.

Un'altra architettura degna di nota è quella POWER-PC. Considerata da molti l'unica rivale dell'architettura x86 in quanto a popolarità e diffusione, è stata utilizzata soprattutto dai computer Macintosh fino a al giugno 2008.

Un'altra caratteristica importante dei processori è la velocità o frequenza di clock, ovvero il numero di commutazione tra gli stadi logici "0" e "1" dei circuiti interni che il processore è in grado di eseguire in un secondo (ovvero un Hertz, 1/s) . In un processore a 2 GHz il processore commuta 2 miliardi (Giga) di volte al secondo (Hertz) e sarà più veloce, a parità delle altre caratteristiche, di un processore con frequenza inferiore.

La maggior parte dei processori attuali sono multi-core, dual core o quad-core ma anche a sei e otto core, questi processori sono costituiti dai più nuclei, appunto core, in grado ciascuno di eseguire delle istruzioni. Un sistema multi-core è generalmente più veloce di un sistema a single-core ma non bisogna applicare l'equazione che un dual-core impiega la metà del tempo per fare la stessa operazione su un single-core in quanto non sempre è possibile sfruttare parallelamente i core disponibili, e anche quando è possibile bisogna considerare il tempo necessario al coordinamento tra le operazioni svolte sui singoli nuclei.

Non bisogna confondere i processori multi-core con i sistemi multiprocessori che sono caratterizzati da una scheda madre in grado di ospitare più processori fisici.

13.2 La RAM

La RAM (Random Access Memory o memoria ad accesso casuale) è la memoria di lavoro del processore e contiene tutti i dati e tutti i programmi che devono essere oggetto di elaborazione del processore, in definitiva tutto ciò che si sta utilizzando in un determinato istante si trova nella ram.

La ram è una memoria di tipo volatile, questo significa che in mancanza di alimentazione perde il suo contenuto. A rendere permanenti le informazioni del computer ci penseranno le memorie permanenti come i dischi che vedremo più avanti.

Il processore prende quindi le istruzioni da eseguire e i dati su cui operare dalla memoria ram, che è molto più lenta del processore, per cui il processore si troverebbe spesso in attesa di informazioni che devono arrivare dalla ram. Questo problema è stato superato inserendo nei computer la memoria cache che si può trovare sia nel processore stesso, cache di primo livello, che in un microchip situato al suo fianco, cache di secondo e terzo livello. Il processore non leggerà i dati direttamente dalla ram ma dalla cache a lui più vicina riducendo così i tempi di attesa. Un sistema di gestione della cache farà in modo

che i dati necessari al processore si troveranno, con buona probabilità, nella cache, questo non sarà sempre possibile ma il sistema della cache a più livelli viene utilizzato in tutti i computer moderni in quanto mediamente ne migliora le presazioni.

13.3 La scheda madre

Un'altra componente hardware fondamentale per un computer è la scheda madre, in inglese motherboard, si tratta di un dispositivo elettronico che presenta dei connettori che consentono di collegare componenti come processori, ram , hard-disk, schede di rete, schede video e,direttamente o tramite schede aggiuntive, tutto ciò che è possibile collegare al computer. Sono presenti anche i circuiti e microchip per il controllo e la gestione della comunicazione tra tutti questi dispositivi (chipset).

La dimensione delle schede madri varia a seconda della espandibilità che consentono, esistono schede ridottissime che non consentono di aggiungere dispositivi a schede più grandi che consentono di aggiungere ad es. un elevato numero di dischi, più schede di rete, diverse schede di acquisizione audio video. I computer desktop hanno una scheda madre solitamente rettangolare, le cui dimensioni sono standardizzate e disponibili in un numero limitato di dimensioni dette anche fattori di forma, i più diffusi sono ATX, nelle versioni standard mini e micro, btx e itx nelle versioni mini e nano.

Per i portatili, vista la necessità di sfruttare al massimo lo spazio disponibile, sono disponibili una miriade di fattori di forma che può cambiare anche tra una versione e la successiva di un computer della stessa marca.

Altra caratteristica della scheda madre è il tipo e numero di connettori o socket contenuti. I socket consentono di espandere un computer aggiungendo nuove periferiche, quelli più diffusi sono PCI, PCI-EXPRESS e AGP, quest'ultimi esclusivamente per schede grafiche.

In linux, un elenco parziale dei componenti direttamente presenti su una scheda madre o collegati ad essa da porte come pci è lspci, per eseguirlo bisogna aprire una shell e digitare il comando indicato.

13.4 L'alimentatore

Scopo dell'alimentatore è quello di fornire le corretta energia elettrica a tutti i dispositivi del computer, esso , in pratica, prende in input la corrente alternata a 220 volt e la trasforma in corrente continua il cui voltaggio cambia in base a dispositivi.

La caratteristica principale di un alimentatore è la sua potenza calcolata in watt. In elettronica il calcolo della potenza si calcola come prodotto dei volt (V) per l'intensità della corrente necessaria calcolata in Ampere (i). Un dispositivo che funziona a 12 v e richiede 3A di intensità richiederà 12*3=36watt. Il numero di watt complessivi di un computer si ottiene sommando i watt di tutti i dispositivi collegati. Se la potenza di un

alimentatore è insufficiente o anche al limite della sufficienza, può succedere che il computer va in crache durante un'operazione che richiede maggiore potenza come, ad es., una masterizzazione.

Un computer da tavolo da usare in un ufficio richiederà, quindi, meno potenza rispetto ad un server con molte componenti.

Altra caratteristica degli alimentatori è la quantità e il tipo di adattatori posseduti, può accadere che nei nuovi alimentatori, ad. es. , non sia disponibile un connettore presenti su schede madri datate. In questi casi o si comprano degli adattatori dal nuovo connettore al vecchio o si cerca per qualche tipo di alimentatore che abbia mantenuto il vecchio adattatore.

Elenco immagini connettori con descrizione del tipo:

Figura 1- Connettore per scheda madre, può essere a 20 o a 24 pin o contatti.

Figura 2 - Connettore aggiuntivo per scheda madre a 4 pin.

Figura 3 - Connettore per dischi PATA

Figura 4 - Connettore per floppy-disk

Figura 5 - Connettore per dischi sata

Nella sostituzione di un alimentatore, per evitare il calcolo della potenza necessaria è buona norma sostituirlo con uno che abbia almeno la stessa potenza e i connettori necessari al collegamento di tutti i dispositivi presenti sul proprio pc.

13.5 I dischi: partizionamento e file-system

I dischi rappresentano la cosi detta memoria di massa del computer in quanto contengono tutto il sw, programmi e dati, disponibile sul computer. I dischi possono essere a tecnologia magnetica, oggi la maggior parte degli hard-disk dei computer usa questa tecnologia, ottica, usata per cd, dvd blu-ray compresi. Da tenere in considerazione anche gli hard-disk a stato solido (SSD) che hanno delle prestazioni superiori a quelle dei dischi magnetici ma una capacità, ad oggi, inferiore, circa un decimo.

I connettori per dischi più diffusi sono: PATA, SATA.

Pata, Parallel ATA, è una tecnologia che prevede un'interfaccia a 40 o 80 pin per il trasferimento dati tra hard-disk e scheda madre. Un cavo di collegamento PATA è caratterizzato dalla presenza di un connettore sul lato da collegare alla scheda madre e da due connettori sul lato di collegamento ai dischi, quindi su un cavo pata possono essere collegati due dispositivi PATA. Sulle schede madri, solitamente, erano presenti due connettori PATA quindi era possibile collegare fino a 4 dischi PATA che potevano ad es. essere due hd magnetici e 2 ottici. Spesso si fa riferimento a questi connettori anche con il termine IDE,Integrated Device Electronics , o EIDE, Enhanced IDE.

La tecnologia che prevedeva la trasmissione in parallelo dei bit sul cavo, PATA, si è dimostrata però meno potente della trasmissione seriale, un bit alla volta, SATA che sta per Serial ATA. Nel corso degli anni la tecnologia SATA ha sostituito quella PATA, tanto che oggi è difficile trovare un HD con tecnologia PATA.

Sata esiste in diverse versioni, 1, 2, 3 sempre più performanti ma che mantengono una compatibilità con le versioni precedenti.

Esiste una versione SATA per hard-sisk esterni chiamata eSATA.

Sata è oggi l'interfaccia per dischi maggiormente utilizzata.

Esiste poi un'altra interfaccia che è quella SCSI, utilizzata su sistemi di fascia alta le cui caratteristiche erano superiori a quelle dei PATA, oggi con l'uscita della tecnologia SATA sono meno utilizzati. Diverse distribuzioni linux trattano i dischi, da un punto di vista sw, come se fossero dischi SCSI.

L'interfaccia USB è utilizzata per il collegamento di dischi esterni.

13.5.1 Partizioni e file system

La gestione del contenuto dei dischi è demandata ad un modulo del sistema operativo chiamato file-system. Tutti i sistemi operativi prevedono l'organizzazione di un disco fisico in una o più aree gestite poi come se fossero dischi indipendenti, operazione

chiamata partizionamento, e poi l'assegnazione a ciascuna di un tipo di organizzazione dei dati su essa tramite l'operazione della formattazione.

Il partizionamento di un disco si fa per diversi motivi, ad es. per installare più sistemi operativi, o avere sistema operativo e programmi su una partizione diversa da quella dei dati degli utenti. Linux richiede una partizione a parte per l'area di swap della ram, usata in caso di insufficienza di quest'ultima.

Il sistema di partizionamento più utilizzato, chiamato MBR, prevede tre tipi di partizione:

PRIMARIA: ciascun disco può avere fino a 4 partizioni primarie, tra queste una può essere partizione estesa. Alcuni sistemi operativi necessitano di essere installati su una partizione primaria.

ESTESA: un disco può avere al massimo una partizione estesa, in questo caso il numero massimo di partizioni primarie scende a tre. La partizione estesa non può essere utilizzata direttamente ma è un contenitore poi che dovrà essere suddiviso a sua volte in una o più unità logiche.

UNITA' LOGICHE: le unità logiche vanno dunque inserite all'interno dell'unica partizione estesa possibile su un disco. Il numero è praticamente infinito in quanto nell'ordine dei miliardi di partizioni possibili, ma difficilmente si supera la decina di unità.

Il sistema di partizionamento fin qui descritto chiamato MBR ha un limite che oggi è stato raggiunto che è quello di due TeraByte per ciascun disco. Il nuovo sistema chiamato GPT prevede dei dischi 4 miliardi di volte più capienti.

In Linux sono disponibili dei comandi partizionamento dei dischi a linea di comando come fdisk, cfdisk e sfdisk. Un es. di output del comando " sudo fdisk -l /dev/sda" è riportato di seguito:

```
Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 testine, 63 settori/tracce, 60801 cilindri, totale 976773168 settori
Unità = settori di 1 * 512 = 512 byte
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identificativo disco: 0xfb25335c

Dispositivo Boot     Start       End     Blocks   Id  System
/dev/sda1   *       2048     409599     203776    7  HPFS/NTFS/exFAT
/dev/sda2         409600  469467135  234528768    7  HPFS/NTFS/exFAT
/dev/sda3      469469182  934115489  232323154    f  W95 Esteso (LBA)
/dev/sda4      934123520  968450047   17163264    7  HPFS/NTFS/exFAT
/dev/sda5      469469184  633309183   81920000   83  Linux
/dev/sda6      633311232  645091327    5890048   82  Linux swap / Solaris
/dev/sda7      645106203  934115489  144504643+   7  HPFS/NTFS/exFAT
```

Libparted è una libreria di gestione delle partizioni che può essere utilizzata sia da programmi a linea di comando che da programmi ad interfaccia grafica come Gparted di cui vediamo un esempio nella figura seguente:

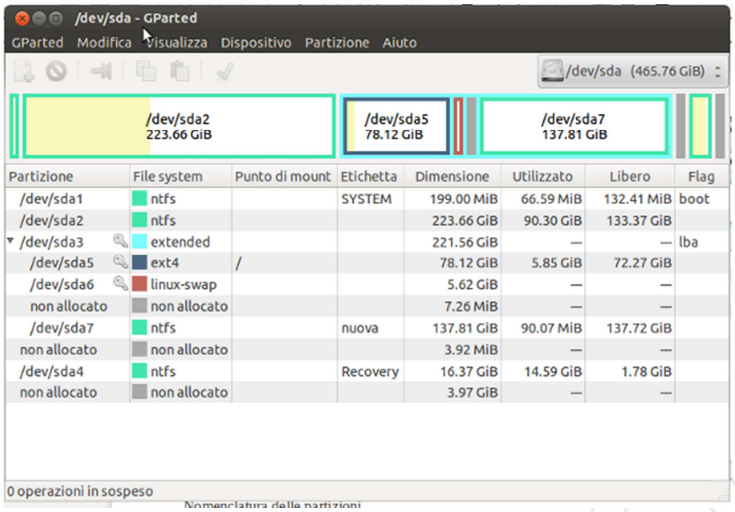


Figura 6 - Gparted

13.5.2 Nomenclatura delle partizioni.

Linux assegna i nomi alle partizioni usando quattro lettere: le prime due indicano il tipo di disco, e possono essere hd per i dischi pata ed sd per dischi sata, usb e SCSI. La terza lettera indica il numero di disco fisico presente sul sistema, a per il primo disco, b per il secondo e così via. Infine, la quarta lettera indica il numero di partizione, da 1 a 4 per le partizioni primarie e dal 5 in poi per quelle logiche.

Riassumendo hdb5 indica la prima partizione logica (5) sul secondo disco (b) che è di tipo pata (hd). Tutti i dischi si trovano all'interno della cartella dev nella cartella principale root ('/'), nel caso di prima il percorso completo sarà /dev/hdb5.

In Windows le partizioni vengono nominate con le lettere dell'alfabeto, a partire dalla lettera c per i tutti i tipi di disco, lasciando le lettere 'a' e 'b' per gli ormai inutilizzati floppy-disk. La lettera dell'alfabeto viene sempre seguita dai due punti (es: c:)

Apple, per i suoi pc usa HFS o, nei sistemi più recenti, HFS+.

I miglioramenti che si sono avuti con l'evoluzione dei diversi file-system possono essere sintetizzati in:

- aumenti della dimensione massima di ciascun file contenuto
- aumento della dimensione massima consentita per ogni partizione
- aumento del numero di caratteri dei nomi assegnabili a file e cartelle
- possibilità di controllo ed eventuale recupero di dati in seguito all'interruzione di energia elettrica (journaled filesystem)

Per i dischi ottici la situazione è più semplice, esiste il vecchio formato ISO-9660 e la sua evoluzione UDF utilizzata anche sui nuovi dischi blu-ray.

Per quanto riguarda il file system, come detto prima, si occupa della organizzazione e gestione dei contenuti di ciascuna partizione. I file system più utilizzati in linux sono ext3, ext4 o ReiserFS, mentre in windows i file-system sono il vecchio fat presente nelle

versioni a 16 ed a 32 bit e il più recente NTFS. Linux riesce a leggere e scrivere su partizioni windows, al contrario, invece, sono disponibili dei programmi sotto windows che consentono l'accesso a file-system linux.

Figura 7 - Esempio di filesystem disponibili in una formattazione con gParted

13.6 I Monitor

La maggior parte degli attuali monitor usano una tecnologia a cristalli liquidi (LCD) o la loro evoluzione che prevede l'uso di led per la retroilluminazione. I monitor a led garantiscono: consumi energetici ridotti, maggiore chiarezza dell'immagine e spessori più contenuti. La storia dei monitor è stata fatta dai monitor a raggi catodici (CRT), nella versione monocolore prima, verde per la precisione, e in quella a colori poi.

Le caratteristiche da tenere in considerazione oltre alla tecnologia sono: risoluzione, fattore di forma, frequenza di aggiornamento e tempo di risposta. La risoluzione indica il numero di pixel, puntini, che colorandosi ciascuno con un solo colore alla volta, determinano l'immagine dello schermo. Un monitor con risoluzione massima 1600x1200 sarà composto da 1,920,000 pixel distribuiti su 1200 righe e 1600 colonne. Più alto è il numero di pixel migliore sarà la qualità dell'immagine visualizzata, sarà più alta anche la memoria richiesta dalla scheda video per visualizzare l'immagine e il lavoro del processore grafico per creare l'immagine da visualizzare.

Il fattore di forma stabilisce le proporzioni tra le due misure del monitor, larghezza e altezza. Si è passati dai monitor con formato 4/3 a quelli più larghi e meno alti con formato 16/9 o 16/10.

La frequenza di aggiornamento indica quante volte al secondo viene aggiornata l'immagine sullo schermo, un monitor a 70Hertz (Hz) aggiornerà il suo contenuto 70 volte al secondo.

Infine il tempo di risposta, nell'ordine di qualche millisecondo, solitamente da 1 a 5, indica il tempo impiegato per il cambio del colore di ciascun pixel. Questa caratteristica è importante per un uso avanzato del monitor come quello che si fa durante i giochi mentre è abbastanza ininfluente per monitor da usare in un ufficio senza particolari elaborazioni grafiche.

I connettori più diffusi per collegare un monitor al computer sono: VGA, DVI e HDMI. I primi di tipo analogico, ormai superati ma ancora disponibili su molti computer e monitor, mentre gli altri due digitali sono caratterizzati dal mantenere una migliore qualità dell'immagine in quanto non hanno bisogna della doppia conversione digitale-analogica prima per avere il segnale sul connettore VGA e analogico-digitale poi per essere visualizzata sugli odierni monitor che sono digitali.

Figura 8 Alcune tipologie di connettori

13.7 L'interfaccia USB

La maggior parte delle periferiche esterne collegabili oggi al computer usa l'interfaccia USB, Universal Seria Bus, disponibile nelle versioni 1.2 con una velocità di trasmissione di 12Mbit/sec, la 2.0 con 480Mbit/sec e 3.0 con i suoi 4800 Mbit /sec.

I connettori usb sono presenti nella versione standard modelli 'A' e 'B', nella versione mini sempre 'A' e 'B' e nella versione micro anche in questo caso 'A' e 'B', quest'ultime molto usata per fotocamere e telefonini.

Figura 9 - Tipologie di connettori basate su standard USB

E' possibile utilizzare la maggior parte dei dispositivi dotati di interfaccia USB semplicemente inserendoli nel connettore, si parla di dispositivi plug & play. Questo è un vantaggio in quanto, in precedenza, prima di utilizzare una nuova periferica come un modem o una stampante, era necessario il riavvio del computer perchè solo in questa fase il sistema operativo rilevava il nuovo hardware e lo associava con il corretto driver.

Domande

1. La licenza GPL di linux garantisce

   A. Libertà di eseguire il programma per qualsiasi scopo.

   B. Libertà di studiare il programma e modificarlo

   C. Libertà di ridistribuire copie del programma

   D. Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti

   E. tutte le precedenti

2. La suddivisione in aree di un hard-disk si chiama

   A. formattazione B. partizionamento C. file-system

3. La formattazione di un disco assegna allo stesso il tipo di file-system

   A. Vero B. Falso

4. Indicare un comando per avere informazioni sul processore utilizzato.

   _____

5. Quali dei seguenti comandi non serve a partizionare un disco:

   A. gparted B. fdisk C. lspci

6. Quale tra le seguenti attività non è di competenza del sistema operativo:

   A. gestione dei contenuti dei dischi

   B. caricamento dei programmi in memoria centrale

   C. conversione dei file multimediali

   D. gestione delle periferiche

   E. nessuna delle precedenti

7. Tutti i sistemi operativi attuali sono compatibili con tutte le periferiche oggi in produzione

   A. Vero B. Falso

8. E' possibile avviare un sistema linux escludendo completamente l'interfaccia grafica

   A. Vero B. Falso

9. Cosa si intende per "distribuzione" Linux?

A. Una collezione completa e installabile di software di sistema e utilities

B. L'attivazione di un file system distribuito

C. Il solo kernel linux

10. Tutto il software che gira su Linux è gratis?

A. Si, è un requisito della GNU

B. No, basta che i sorgenti siano disponibili

C. No, può anche essere a pagamento e/o con sola licenza d'uso

11. La Shell rappresenta:

A. Un programma linux interprete dei comandi dell'utente

B. Un'utility di linux che consente di impostare le variabili di ambiente

C. Una delle interfacce grafiche di linux

12. Il sistema operativo GNU-linux è adatto solo per personal computer e server e non per altre piattaforme.

A. Vero B. Falso

14 Dove archiviare i dati

Leggere, memorizzare e organizzare dati, in una parola filesystem. Contenuto all'interno di tutti i sistemi operativi, i filesystem, offrono agli sviluppatori la possibilità di integrare i propri software all'interno del sistema operativo e strutturare le informazioni in modo da garantire sicurezza, coerenza e omogeneità dei dati. Quasi tutti i sistemi attuali, e Linux non fa eccezione, offrono linee guide su come organizzare il proprio filesystem, definendo una gerarchia File System Hierarchy.

14.1 Un po' di storia.

Prima dell'avvento di Linux, esistevano diversi standard proprietari che definivano il l'organizzazione del filesystem. *Unix V7 Filesystem*, nato nel lontano 1979, è considerato il primo tra questi standard, a cui fanno seguito *SunOS Filesytem* (1988) e ,l'*HP-UX Filesystem*.
Nell'agosto del 1993, a pochi anni dalla nascita di Linux, era forte l'esigenza di delineare un nuovo standard che definisse la struttura gerarchica del filesystem, in modo da rendere più omogenea la collocazione dei file del sistema e degli applicativi fra le diverse distribuzioni. Su questa nuova scia nasce FSSTND (*Filesystem Standard*), rilasciato per la **prima** volta nel febbraio 1994.

Più tardi, nel 1996, grazie ad un gruppo di sviluppatori BSD, è stato possibile fare un ulteriore passo in avanti. Si concretizzava, infatti, la possibilità di rendere lo standard applicabile a tutti i sistemi Unix-like e non solo alle distribuzioni Linux. Tale cambiamento, accolto molto positivamente dagli addetti ai lavori, segnava una profonda linea di demarcazione con il passato, tanto da rendere indispensabile ribattezzare lo standard con il nome Filesystem Hierarchy Standard. Attualmente lo standard, alla versione 2.3, è mantenuto dal FHS Group ed è in continua evoluzione (la versione 3.0 è attualmente in sviluppo).

14.2 Gerarchia del File System

Nel mondo Linux, più in generale Unix,non vi sono lettere assegnate ai vari dispositivi. Tutti i dispositivi fanno riferimento alla directory root (simbolo “/”, ovvero directory radice). Il layout della directory root è definito dallo standard Filesytem Hierarchy Standard (FHS), attualmente adottato dalle principali distribuzioni Linux e da altri sistemi Unix-like.

Quello che segue è l'elenco directory (o dei link simbolici) strettamente necessario e richiesto nella directory root (“/”) per rispettare lo standard:

/bin
/boot
/dev
/etc
/lib
/media
/mnt
/opt
/sbin
/srv
/tmp
/usr
/var

Vanno altresì aggiunte le seguenti cartelle “opzionali” (secondo lo standard), che di fatto sono però presenti in qualsiasi distribuzione Linux

root
home

Segue una breve descrizione sui contenuti delle suddette cartelle:

/boot – file di avvio La directory contiene tutti i file statici necessari all'avvio del sistema. Fanno parte di questi file le immagini del Kernel e i file di configurazione dei vari bootloader (es. GRUB o LILO). In ~~alcuni~~ taluni casi le immagini del Kernel possono risiedere anche nella directory root /.

/bin/ - comandi essenziali di sistema La directory contiene i binari degli applicativi necessari al corretto avvio e utilizzo del sistema. Molti di questi sono utilizzati in fase di avvio del sistema, per esempio mount e mknod; altri sono invece essenziali per avere un livello minimo di gestione del sistema, sia da parte degli amministratori di sistema che degli utenti (per esempio ls, date e mkdir).

/dev/ - file dei dispositivi In /dev e nelle sue sottodirectory sono collocati i device file (anche detti special file). Questi file privi di contenuto, hanno la funzione di fornire una semplice interfaccia verso i driver del kernel che gestiscono i vari dispositivi. È possibile distinguere tra due differenti tipologie di device file, che corrispondono a due diversi dispositivi: i character special file (per i dispositivi a carattere) e i block special file (per i dispositivi a blocchi). I primi sono periferiche che processano, in ingresso o in uscita, un singolo carattere per volta. Appartengono a questa categoria dispositivi quali tastiere, mouse e modem seriali. Appartengono, invece, ai dispositivi a blocchi quelli che processano i dati in forma di blocchi di dati (tipicamente 512 byte o più), tra cui hard disk, lettori DVD, lettori di Flash Card e più in generale tutti quei dispositivi dove la lettura e la scrittura di informazioni non è sequenziale. È possibile osservare a quale tipo di periferiche appartengono i diversi file di dispositivo eseguendo il comando “ls -l /dev”.

```
crw-------  1 root root 5,  1 set  2 15:53 console
crw-rw----  1 root video 29, 0 set  2 15:53 fb0
```

```
brw-rw----  1  root  disk  8,  0  set  2  15:53  sda
brw-rw----  1  root  disk  8,  1  set  2  15:53  sda1
brw-rw----  1  root  disk  8,  2  set  2  15:53  sda2
brw-rw----  1  root  disk  8,  3  set  2  15:53  sda3
brw-rw----  1  root  disk  8,  4  set  2  15:53  sda4
brw-rw----  1  root  disk  8,  5  set  2  15:53  sda5
brw-rw---- 1 root disk 8, 6 set 2 15:53 sda6
```

Osservando la prima colonna di lettere è possibile riconoscere i diversi dispositivi. I device contraddistinti con una "c" sono dispositivi a caratteri, mentre quelli contraddistinti dalla lettera "b" sono dispositivi a blocchi. Inoltre, osservando l'output è possibile notare che nella colonna indicante la dimensione del file sono presenti due numeri. Il primo è il major number ed indica il tipo di driver utilizzato dal kernel per gestire la periferica, il secondo è minor number e la sua funzione è quella di distinguere fra i vari dispositivi che uno stesso driver può controllare oppure contraddistinguere tra diverse partizioni in caso di dispositivi di memorizzazione. Nell'esempio, il device /dev/sda1 ha come major number 8 (che corrispondente agli hard disk SATA/SCSI), mentre il numero 1 indica la partizione del disco.

Oltre alle periferiche fisiche, all'interno di /dev sono elencati ~~anche~~ gli pseudo-dispositivi (pseudo-devices). Si tratta di file di dispositivo a cui non corrisponde una periferica fisica, bensì funzioni e algoritmi inclusi nel kernel che si comportano alla stregua di dispositivi reali. Il più noto è /dev/null, meglio conosciuto come null device (dispositivo nullo) accessibile a qualsiasi utente. Questo pseudo-dispositivo funziona come un vero e proprio buco nero, accettando come input qualsiasi tipo di informazione e non restituendo alcun dato (più precisamente restituisce esclusivamente un EOF, end-of-file). Il device /dev/null viene utilizzato comunemente per ignorare gli output dei programmi non desiderati.

Per esempio, il comando:

```
$ programma > /dev/null
```

farà sì che lo standard output di programma sia reindirizzato in /dev/null: di conseguenza non verrà visualizzata nessuna informazione nel terminale.

Altri due pseudo-device molto importanti sono /dev/random e /dev/urandom. Entrambi producono in output dati casuali, ma con una sostanziale differenza: /dev/random genera dati casuali più accurati rispetto a /dev/urandom, ma ciò va a discapito della velocità~~.~~ La differenza fra i due device è dovuta al sistema utilizzato per la generazione dei dati casuali. /dev/random utilizza esclusivamente il "rumore" (noise) generato dalle periferiche del sistema. Un esempio di rumore è costituito dal tempo che intercorre tra la pressione di due tasti sulla tastiera. La lentezza di questa pseudo-periferica è data proprio dalla possibilità che tali dati non siano disponibili al momento della richiesta. /dev/urandom, invece, non attinge i dati casuali esclusivamente dal rumore di sistema, ma li genera tempestivamente in caso di necessità (da qui la minore precisione). Il device /dev/zero, infine, restituisce un numero infinito di null bytes. Può essere

utilizzato, per esempio, per sovrascrivere le informazione contenute in un file.

Attenzione: l'output di /dev/null differisce da /dev/zero, come si può intuire eseguendo i seguenti comandi:

```
$ cat /dev/null
$ cat /dev/zero
```

Entrambi non producono nessun output visibile a schermo. Tuttavia, /dev/ restituisce un EOF, mettendo fine immediatamente al programm cat, mentre con il secondo comandocat rimarrà in esecuzione (se non interrotto dall'utente) in quanto lo psedo-dispositivo /dev/zero continuerà a fornire all'infinito dei null byte.

/etc/ file di configurazione – Questa directory contiene i file di configurazione statici del sistema, nonché i file di script di avvio dei servizi necessari al corretto funzionamento del sistema. Per essere conforme allo standard FHS, la directory non può contenere file binari. Attraverso i file di configurazione è possibile modificare il comportamento di alcuni programmi. Molti di questi file hanno un contenuto sensibile e, pertanto, possono essere modificati solo dagli amministratori di sistema (agli utenti è concesso solo accedere in lettura).

Nota bene: /etc ospita, per lo più, file di configurazione globali, ovvero identici per tutti gli utenti. Alcune applicazioni, però, hanno bisogno di un diverso file di configurazione per ogni utente. In questo caso, ~~tali~~ questi file vengono creati nelle rispettive home degli utenti.
Visto la centralità di alcuni file presenti nella cartella /etc, faremo un breve elenco includendo solo i più importanti:

Sono di seguito brevemente descritti alcuni tra i più importanti file contenuti nella cartella /etc:

/etc/fstab Ogni sistema contiene un certo numero di partizioni che devono o meno essere montate all'avvio o in momenti successivi ad esso. /etc/fstab contiene le informazioni e le opzioni dei filesystem, ovvero come il percorso al file di dispositivo (es. /dev/sda1), il loro punto di mount, i permessi per l'accesso e così via. /etc/fstab è un file statico, creato al momento dell'installazione della distribuzione e modificabile esclusivamente dall'amministratore di sistema.

Nota bene: per effettuare il mount di un filesystem, quest'ultimo non deve essere necessariamente presente nel file /etc/fstab: l'amministratore di sistema ha sempre, comunque, la possibilità di montare un filesystem, con l'ausilio del comando mount.

Nota bene: nel caso di sistemi con più periferiche di archiviazione è possibile che il kernel,al momento dell'aggiunta di un nuovo dispositivo, attribuisca una diversa nomenclatura alle periferiche già esistenti (es. /dev/sda1 può diventare /dev/sdb1), provocando forti disagi all'avvio del sistema. Per questo molte distribuzioni linux, al momento dell'installazione, generano un fstab identificando le varie periferiche

attraverso l'UUID (*Universally Unique IDentifier,* ID unico di identificazione) e non con il nome del device file.

L'esempio sottostante illustra due modi equivalenti per identificare in fstab la stessa periferica:

```
#Mount con UUID
UUID=facd5af0-b7fd-4239-8f3a-7b0a4314898b /mnt/backup ext4 noatime 0 0

#Mount specificando il percorso del device file
/dev/sda3 /mnt/backup ext4 noatime 0 0
```

/etc/group contiene l'elenco dei gruppi di utenti presenti nel sistema. Ogni gruppo è identificato da una singola riga contente l'ID univoco del gruppo e l'elenco degli utenti che ne fanno parte.

/etc/hosts contiene informazioni per la risoluzione degli host name in esso contenuti. Questo file è utile nella configurazione di piccole reti, in quanto rende superflua l'adozione di un server per la risoluzione dei domini (DNS).

/etc/inittab permette di configurare e personalizzare il comportamento del processo di init e la configurazione dei terminali da avviare. Dalla versione 6.10 di Ubuntu, il file inittab non è più presente ed è stato sostituito dal nuovo sistema di avvio upstart.

/etc/motd contiene il messaggio del giorno (motd è l'acronimo di **m**essage **o**f **t**he **d**ay) che viene visualizzato in seguito al login dell'utente. Questo file è utile agli amministratori di sistema che vogliono comunicare notizie importanti agli utenti all'inizio della propria sessione.

/etc/mtab contiene le informazioni riguardanti i filesystem in uso nel sistema ed è generato e aggiornato automaticamente, a differenza di /etc/fstab che, come precedentemente detto, è un file statico. A differenza di fstab, la cui funzione è quella di elencare le opzione di mount dei vari filesystem, il file mtab contiene esclusivamente i filesystem montati. Il contenuto di /etc/mtab è molto simile a quello che si può ottenere con il comando

```
$ mount
```

È da notare come /etc/mtab sia l'unico file a non mantenere la natura statica tipica dei file presenti in /etc/, scelta dettata da ragioni storiche e di retro-compatibilità.

/etc/passwd contiene l'elenco degli utenti, le rispettive password e altre informazioni importanti come il percorso della propria home directory e la shell assegnata all'utente. Per una maggiore riservatezza, nelle moderne distribuzioni le password vengono memorizzate nel file **etc/shadow,** non accessibile agli utenti non autorizzati. Questo tipo di configurazione impedisce l'accesso ai dati personali, limitando il rischio di attacchi “brute-force” sulle password degli utenti del sistema.

Oltre a questi file, la directory /etc contiene alcuni importanti subdirectory che, a loro volta, contengono altri file di configurazione. È da sottolineare come la struttura delle

subdirectory (nonché il nome di alcuni file) possa differire a seconda della distribuzione usata.

/etc/X11/ contiene i file di configurazione di X Windows System o di X.Org Server (se installati), i due gestori grafici più diffusi nei sistemi Unix-like.

/etc/init.d/ contiene gli script di avvio dei vari servizi del sistema. A seconda dalla distribuzione usata, gli script possono trovarsi in altre cartelle. Per esempio, nella distribuzione Red Hat e derivate, gli script di avvio si trovano nella cartella /etc/rc.d/.

/home - directory home degli utenti Questa directory contiene le home directory di tutti gli utenti (ad esclusione dell'utente root). La configurazione ideale prevede che, fatta eccezione per gli amministratori di sistema, ogni utente possa accedere esclusivamente alla propria home directory, garantendo così la riservatezza di tutti gli utenti del sistema.

/lib – Librerie condivise essenziali e moduli del kernel La directory /lib contiene le librerie necessarie all'avvio del sistema e/o ai programmi essenziali presenti in **/bin e /sbin**. Le librerie sono contraddistinte dall'estensione .so, l'equivalente nei sistemi Windows™ delle Dynamically Linked Library (DLL). Nelle directory **/lib/modules/<versione-kernel>** sono presenti i moduli del kernel organizzati in subdirectory, e rispecchiando la struttura dei codici sorgenti del kernel stesso.

Nota bene: oltre alla cartella /lib, in alcuni sistemi è possibile trovare una seconda cartella /lib<architettura> (es. /lib32). L'esistenza di questa directory è legata alla necessità da parte di alcuni applicativi, di accedere a librerie di architetture differenti. L'esempio più comune si ha con distribuzioni compilate per i sistemi a x86_64 (anche chiamate AMD64) dove si ha la necessità di avviare applicativi compilati per sistemi x86.

/media – Punto di mount per le periferiche removibili La directory **/media** contiene le subdirectory usate per il mount delle periferiche removibili, come flash card, CDROM e Pen Drive USB. Alcune distribuzioni non fanno uso di /media e utilizzano esclusivamente la cartella /mnt.

/mnt – Punto di mount per filesystem temporanei Le subdirectory presenti in questa cartella sono utilizzate dall'amministratore di sistema per il mount temporaneo di un filesystem quando necessario.

/opt – Add-on Software Questa directory è riservata all'installazione di applicazioni add-on oppure a pacchetti software che non fanno parte della distribuzione in uso. Perchè sia conforme allo standard FHS, ogni pacchetto installato in /opt deve avere una propria directory con il nome del software o del produttore (es. /opt/<pacchetto> o /opt/<produttore>). Uno dei motivi dell'esistenza di questa directory è quella di evitare che librerie e binari di software di terze parti vadano in conflitto con gli applicativi e le librerie della distribuzione in uso.

/root – Home directory per l'utente root /root è la home directory dell'utente root e non è accessibile dagli utenti normali. Come accennato precedentemente, questa cartella non risiede /home/, ma è collocata nel filesystem principale. Tale scelta non è

casuale: comunemente la directory /home si trova in un filesystem diverso rispetto alla root di sistema. Se, per qualsiasi motivo, fallisse il mount di tale directory, l'amministratore di sistema non avrebbe accesso ai dati e alle impostazioni contenuti nella propria home directory, causando non pochi disagi.

/sbin – Binari di sistema In questa directory sono contenuti i programmi necessari all'avvio (es. init, getty) e quelli utili in caso di danneggiamento del sistema (es. fsck).

/srv – Cartella dati dei servizi offerti dal sistema In questa cartella vengono collocati i dati dei vari servizi installati sul sistema. Non vi è un vero e proprio metodo su come organizzare la struttura delle sottodirectory. Solitamente si cerca di suddividere i dati in sottodirectory assegnando nomi che coincidono con quelli dei protocolli, per esempio /srv/ftp e /srv/www. Molte distribuzioni non fanno uso di questa cartella, delegando tali dati in /var o altri percorsi.

/tmp – Cartella dei file temporanei Nella cartella /tmp, accessibile da tutti gli utenti, vengono collocati i file temporanei prodotti dagli applicativi. Molte distribuzioni scelgono l'eliminazione automatica ad ogni avvio del contenuto di questa directory: cancellano il contenuto di questa directory ad ogni avvio del sistema: di conseguenza i file temporanei prodotti dagli applicativi non devono essere rilevanti per le successive esecuzione.

/usr – User System Resources Questa è una delle cartelle più importanti dei sistemi Linux, in quanto al suo interno risiedono tutte le librerie e i binari non necessari all'avvio del sistema. Tra le più importanti sottocartelle troviamo:

/usr/bin contienela maggior parte dei comandi non necessari all'avvio del sistema.

/usr/include ospita i file header inclusi nei programmi C.

/usr/lib contiene le librerie degli applicativi, ad esclusione di quelle necessarie all'avvio dei binari presenti in /bin e /sbin. Come per la cartella /lib, è possibile trovare cartelle del tipo /usr/lib<architettura> in caso della coesistenza di binari appartenenti a diverse architetture.

/usr/local Questa cartella replica la struttura della cartella /usr, creando un nuovo livello di gerarchia. Viene utilizzata soprattutto per l'installazione di pacchetti che non fanno parte della distribuzione, oppure per i pacchetti compilati e installati direttamente dai sorgenti. La sua funzione è utile per evitare la sovrascrittura di binari e/o il conflitto di librerie del sistema.

/usr/sbin Questa cartella contiene gli applicativi per gli amministratori di sistema, che non sono necessari né all'avvio né in caso di danneggiamento del sistema.

/usr/share Qui risiedono i dati condivisi degli applicativi che non dipendono dall'architettura del sistema. In passato, quando lo spazio era limitato, condividere tali dati fra più server e tra piattaforme diverse era ~~considerato~~ essenziale. Questa peculiarità è successivamente svanita grazie alla sempre maggiore disponibilità di spazio offerta dai dispositivi moderni. Tra i dati che possiamo trovare in questa directory vi sono:

/usr/share/man che contiene le manual pages (pagine di manuale), ovvero i file utilizzati dal programma man;

/usr/share/doc dove risiede la documentazione del sistema e dei singoli applicativi, nonché template, how-to e file di configurazione di esempio ;

/usr/share/info contene le info pages dell'applicativo info;

/usr/src Contiene i sorgenti del Kernel o di altri applicativi. Nella maggior parte delle distribuzione questa cartella è vuota in quanto i soli sorgenti del Kernel occupano più di 700 MB.

/usr/games Contiene i file binari dei giochi

/var –Dati variabili La directory /var contiene i dati che sono soggetti a modifiche durante il normale utilizzo del sistema. Sono qui collocati i file spool di stampa (/var/spool/cups), i log sistema /var/log, i file temporanei che non possono essere cancellati tra diversi riavvi del sistema (/var/tmp). Il programma man, per esempio, usa la directory /var/cache/man per depositare i file man decompressi e formattati. In /var/mail, invece, è possibile trovare le mail degli utenti del sistema non ancora consultate. In /var/run, infine, si trovano le informazione relative ai programmi in esecuzione del sistema.

Nota bene: la suddivisione dei dati in continuo cambiamento in una directory diversa da /usr permette agli amministratori di sistema di poter effettuare il mount di /usr anche in sola lettura.

Nota bene: come per /usr/local, anche /var/local riproduce un nuovo livello di gerarchia della cartella /var, permettendo l'utilizzo delle directory a programmi non facenti parte appartenenti alla distribuzione corrente o installati da utenti diversi dall'amministratore di sistema.

/proc – Pseudo-filesystem dei processi e del kernel Questo pseudo-filesystem (che non occupa spazio su disco) offre una vera e propria interfaccia per le strutture del kernel. Originariamente la funzione principale era quella di esportare tutte le informazioni relative allo stato dei processi in user space. Con il tempo si sono aggiunte diverse funzioni come, ad esempio, informazioni sull'hardware, sulle periferiche IDE e SCSI e sull'utilizzo della rete.

La maggior parte dei file contenuti in questa directory sono di sola lettura. Fanno eccezioni alcuni file dove è possibile modificare le variabili del kernel senza interrompere l'esecuzione del sistema. Sempre tramite /proc è possibile ricavare informazioni riguardanti l'hardware e il software in uso. Ecco un elenco, non integrale, dei più importanti file:

/proc/cpuinfo contiene informazioni sulla CPU del sistema quali frequenza, modello, dimensione della cache.

/proc/filesystems elenca tutti i tipi di filesystem supportati dal sistema, indicando quelli in uso.

/proc/interrupts elenca il numero degli interrupt generati nel sistema, suddivisi per CPU e per driver che li ha generati.

/proc/kcore rappresenta la memoria fisica del sistema e, infatti, la sua dimensione è uguale a quella della memoria fisica installata sul macchina. Questo file è molto utile per effettuare debug di sistema e di applicativi grazie a programmi quali gdb.

/proc/meminfo elenca tutte le informazioni riguardanti la memoria di sistema. Il comando free fa riferimento a questo file per ricavare le informazioni necessarie sulla memoria.

/proc/swaps elenca tutte le periferiche di swap presenti nel sistema.

/proc/mounts elenca tutti i filesystem montati nel sistema.

/proc/version fornisce informazioni dettagliate sulla versione del kernel Linux utilizzata, sull'utente che l'ha compilata, sulla versione del compilatore usata e la data di compilazione.

/proc/loadavg fornisce informazioni sull'utilizzo della CPU per gli ultimi 1, 5 e 15 minuti.

/proc/uptime mostra sia il tempo trascorso dall'avvio della macchina che il periodo di inattività della stessa (idle). 4.3.1 Gerarchia del File System

Nota: il comando uptime fa riferimento a /proc/uptime e /proc/loadavf per ricavare le informazioni necessarie.

Nonostante /proc fornisce molte informazioni, la consultazione dei singoli file non è né pratica né immediata. Per la consultazione dei processi, per esempio, sono utili due tool forniti a corredo di qualsiasi distribuzione: top e ps.

/sys – Informazioni driver e sulle periferiche Uno dei limiti di /proc è quello di avere una scarsa organizzazione delle informazioni. /sys, di contro, presenta le informazioni così come sono viste dal Kernel Linux, fornendo informazioni sulle periferiche, sui driver e sulle loro interconnessioni. Per fornire tali informazioni, la directory /sys è strutturata in modo estremamente ramificato e con molti link ridondanti fra le diverse subdirectory.

14.3 Log di sistema

Come accennato nel paragrafo precedente, ogni sistema Linux mantiene una lunga varietà di log, di solito localizzati in /var/log. La gestione dei log di sistema viene gestita da due demoni principali: syslogd e klogd. Come dice il nome stesso, klogd si occupa di intercettare i messaggi provenienti dal kernel. In alcune distribuzione, però, questa funzione è delegata direttamente a syslogd, rendendo inutile klogd. Alcuni dei file generati rivestono una grande importanza per l'amministratore di sistema, in quanto offrono a quest'ultimo una finestra sulle attività del kernel e degli applicativi. Fra i più importanti file di log possiamo accennare:

/var/log/dmesg Contiene le informazioni del kernel fornite durante il boot ed è visualizzabile da qualsiasi utente di sistema. Per visualizzarne il contenuto è possibile digitare da console il comando dmesg.

/var/log/messages Questo è, di facto, il file di log più importante del sistema e contiene a suo interno i messaggi di boot (esclusi quelli del kernel), degli applicativi di sistema e tutti i messaggi visualizzati dal comando dmesg. Solo l'amministratore può visualizzarne il contenuto.

/var/log/secure Questo file contiene informazioni riguardante la sicurezza del sistema. In esso vengono registrati tutti le informazioni riguardanti il login degli utenti e messaggi di avviso e errori riguardante la connettività di rete. L'accesso a questo file è riservato all'amministratore di sistema.

/var/log/daemon.log Questo file contiene tutte le informazioni generate dai demoni di sistema.

Nota bene: in alcune distribuzioni alcune di questi file possono non comparire. Questo dipende dalla configurazione di syslogd. Per esempio, in alcune distribuzioni il file daemon.log è accorpato al file /var/log/messages.

ISBN: 9788866850090

15 Il tuo computer in rete

Quando un computer viene connesso ad Internet le sue funzionalità si estendono in maniera esponenziale, indipendentemente dal fatto che questa connessione avvenga attraverso la LAN aziendale, il router ADSL di casa o una rete WiFi. Nonostante la molteplicità di metodi per collegarvisi, Internet svolge le sue funzioni sempre allo stesso modo. In questo capitolo si analizza come un sistema Linux si confronta con le problematiche base della connessione ad una rete e come affrontare eventuali problemi.

GLOSSARIO INTRODUTTIVO

- DNS - il *Domain Name System* è una rete globale di server che compone il database di corrispondenze biunivoche tra *hostname* e indirizzi IP. La maggior parte dei computer ha come parametro di configurazione l'informazione riguardante un server DNS, per permettere la risoluzione degli *hostname*.
- DHCP - Il *Dynamic Host Configuration Protocol* permette di ottenere automaticamente da un apposito server (il *server DHCP*) i parametri di configurazione necessari ad un computer per partecipare ad una rete. DHCP è il fulcro sul quale si basa la facilità di configurazione delle reti attuali. Il suo utilizzo viene limitato da considerazioni di sicurezza.
- ETHERNET - E' lo standard hardware che viene utilizzato attualmente nella maggioranza delle reti cablate a breve distanza. Ha diverse specifiche a seconda delle versioni: la sua velocità di trasferimento dati varia dai 10 megabit al secondo (Mbps) per arrivare, nelle declinazioni più recenti, ad un gigabit al secondo (Gbps).
- HOSTNAME - una serie di caratteri alfanumerici (eventualmente separati da ".") che vengono assegnati ad un computer per facilitarne il riconoscimento agli esseri umani. Un hostname é un nome semplice se non vengono usati punti, ad es. *serverposta,* o un nome a dominio, ad es. *pluto.uniroma3.edu,* in questi casi si compone di una porzione host e una porzione di rete. Nell'esempio di prima*, pluto* è il nome della macchina mentre *uniroma3.edu* è il nome della rete cui questa macchina appartiene.
- INTERNET - Usato generalmente con la *i* minuscola, *internet* rappresenta una rete di reti collegate tra loro attraverso apparati detti *router.* Con la *I* maiuscola, il termine *Internet* rappresenta la ben nota rete globale. *Internet* è la più grande *internet* mai creata.
- INDIRIZZO IP - Un indirizzo IP è un numero che viene assegnato ad un computer con l'obiettivo di venire riconosciuto all'interno di una rete. Il suo significato è in tutto simile a quello di un numero telefonico o un indirizzo postale. Nella versione IPV4 é formato da 32 bit che vengono raggruppati in blocchi da 8 per poterne facilitare la conversione in decimale e la comprensione agli esseri umani.

Gli indirizzi IP permettono ai computer di comunicare tra loro all'interno di una rete e vengono divisi in due porzioni: una porzione *host* e una di rete. La porzione di rete identifica la rete alla quale il computer sta partecipando, mentre la porzione host identifica il computer stesso. La distinzione tra le due porzioni viene operata tramite l'utilizzo di una *maschera di sottorete.* Oggi si sta passando alla nuova versione IPV6 a 128 bit.

- MASCHERA DI SOTTORETE - La maschera di sottorete (*subnet mask*), applicata ad un indirizzo IP, permette di definire quali bit fanno parte della porzione *network* dell'indirizzo stesso e quali della porzione *host;* si compone di 32 bit che vengono raggruppati in blocchi da 8 similarmente a quanto accade per gli indirizzi IP. Per ogni bit della maschera di sottorete avente valore "1", il corrispondente bit dell'indirizzo IP è parte della porzione di rete; per ogni bit della maschera di sottorete avente valore "0", il corrispondente bit dell'indirizzo IP è parte della porzione *host.*
- ROUTER - Il *router* è un apparato che connette due o più reti tra loro; al loro interno spesso (soprattutto sui modelli destinati ad uso domestico) sono implementati servizi base come il DNS ed il DHCP. Qualora si vogliano far comunicare due computer non appartenenti alla stessa rete, il router rilancia i dati verso altri router per raggiungere la destinazione.
- SSH - il *Secure SHell* è un protocollo applicativo che permette la connessione sicura a computer remoti
- TCP/IP - Il *Transmission Control Protocol/Internet Protocol* è un set di standard che sottintende la maggior parte delle moderne reti di comunicazione.
- WIFI - Questo termine viene utilizzato generalmente per identificare le reti senza fili, delineate dallo standard *IEEE 802.11*. Sono state sviluppate molte varianti per questo standard che ne estendono le capacità.

15.1 Fondamenti di networking

15.1.1 Introduzione e protocolli

Internet si basa su una famiglia di protocolli chiamata TCP/IP. Un protocollo non è altro che una serie di operazioni concordate che i computer eseguono per poter comunicare. L'applicazione di protocolli, dunque, spazia su tutte le tipologie di comunicazione, dai segnali elettrici alla richiesta di una pagina web. Generalizzando, è possibile distinguere tra tre tipi differenti di protocolli:

- Protocolli di accesso al mezzo fisico: governano la trasmissione dei dati a livello di schede di rete e cavi. Includono, ad esempio, Ethernet per le LAN o IEEE 802.11 per le reti wireless
- Protocolli di comunicazione: governano le comunicazioni tra elementi di reti differenti. Se si vuole accedere dall'Italia ad un server situato in Australia, i

protocolli di comunicazione IP e TCP permettono che i dati scambiati arrivino a destinazione.

- Protocolli di applicazione: assicurano che il destinatario dei dati in Australia possa effettivamente utilizzarli. Il protocollo di applicazione del Web, ad esempio, HTTP, permette di recuperare un'immagine; il server in Australia interpreta la richiesta e restituisce l'immagine desiderata mentre il web browser in Italia interpreta effettivamente che è stata inoltrata un'immagine invece di un messaggio di errore.

Questo approccio multi-livello (usato anche dal "protocollo o pila ISO/OSI", composto da sette livelli) ha il considerevole vantaggio di permettere la comunicazione solo tra livelli contigui. Il protocollo applicativo HTTP non ha necessità di conoscere che i dati viaggiano tra Australia e Italia perché delega questa funzione al protocollo di comunicazione. Quest'ultimo lascia il compito della trasmissione effettiva dei dati al protocollo di accesso al mezzo. La suddivisione del processo di comunicazione in livelli ben definiti, inoltre, permette un approccio sistematico ed efficiente alla risoluzione dei problemi

I protocolli di applicazione TCP/IP -oltre HTTP, questi includono SMTP (invio email), SSH (per login remoti), DNS (per la risoluzione di *hostname*), SIP (VoIP) e tanti altri- sono principalmente basati su due protocolli di comunicazione: UDP o TCP. Mentre UDP permette un servizio *senza connessione e non affidabile* (non c'è garanzia, infatti, che i dati trasmessi arrivino a destinazione nello stesso ordine nel quale sono stati inviati, o che arrivino del tutto), TCP al contrario offre un *servizio con connessione ed affidabile* (i dati inviati arrivano tutti nello stesso ordine con il quale sono stati inviati) a discapito però della velocità.

E' l'applicazione a stabilire quale protocollo di comunicazione sia il più appropriato. Sul Web, ad esempio, si vuole garanzia di ricezione dei dati trasmessi (siano essi del testo, immagini, o del software) dunque TCP è la scelta corretta. Per lo streaming video o le chiamate vocali, al contrario, si predilige la continuità del servizio a scapito di eventuali perdite di dati favorendo quindi la scelta del protocollo UDP. UDP inoltre è preferibile per servizi come DNS che, non richiedendo un elevato scambio di dati, permettono risposte in tempi brevi.

Il terzo protocollo di comunicazione per TCP/IP, ICMP, viene usato per controllo e risoluzione di problemi di rete. Normalmente non viene usato da applicazioni al livello utente. Ne esamineremo un'implementazione in seguito.

VERIFICA:

- Quali sono le principali differenze tra UDP e TCP?
- Perchè nei processi di comunicazione si preferisce un approccio multilivello?

15.1.2 Indirizzamento e routing

Per permettere accesso diretto ad Internet -si inoltra una richiesta ad un server specifico, la cui risposta deve tornare al computer che ha generato la richiesta- ogni computer usa un identificativo univoco, l'indirizzo IP. Nella versione più popolare al momento, IPv4, questo si compone di una sequenza di quattro "ottetti", ovvero gruppi di 8 cifre binarie in grado di comporre numeri decimali da 0 a 255, ad esempio 192.168.178.10.

Non è possibile scegliere il proprio indirizzo IP in maniera arbitraria; questo infatti viene assegnato in modo dinamico, quando cambia ad ogni nuova connessione alla rete come nel caso dell'accensione del computer o del router di collegamento a Internet, o in modo statico se viene impostato e rimane sempre lo stesso per tutte le connessioni. In un'azienda, l'assegnazione dell'IP viene gestita dall'amministratore di rete o di sistema, mentre sarà l'ISP a farsene carico per quanto riguarda la connessione a Internet.

Approfondimento: In ambiente aziendale, probabilmente, è preferibile avere indirizzi assegnati in modo statico in modo da rintracciare facilmente i server aziendali mentre in ambito domestico l'ISP tenderà ad assegnare un indirizzo IP solo per un tempo limitato. Questo permette agli ISP da un lato di usare un numero di indirizzi IP inferiore al numero dei propri utenti (non tutti gli utenti, infatti, sono online nello stesso istante), dall'altro di scoraggiare gli utenti a fornire a loro volta servizi bastati su risorse internet, visto che il cambio costante di indirizzo IP potrebbe risultare in un disservizio.

Gli indirizzi IP non sono assegnati casualmente ma secondo una logica, favorendo lo scambio di dati tra reti differenti, altrimenti detto *routing*. Si ritornerà su questo concetto in seguito.

I computer generalmente hanno più di un indirizzo IP. L'indirizzo di loopback 127.0.0.1 è disponibile su ogni computer e si riferisce al computer stesso; non è accessibile dall'esterno. Un servizio che utilizza l'indirizzo 127.0.0.1 è accessibile solo da client attivi sul computer che sta erogando il servizio stesso.

Approfondimento: L'utilità di questo indirizzo va ben oltre quanto intuitivamente si potrebbe pensare; nello sviluppo di applicazioni con funzionalità di rete, l'indirizzo di loopback permette una verifica di funzionalità preliminare in ambiente controllato, senza la necessità di lanciare in rete -e quindi esporre- le applicazioni stesse.

Per computer che hanno a disposizione più di un interfaccia di rete -ad esempio Ethernet e WiFi- ad ognuna di queste corrisponde un indirizzo IP (quantomeno per ognuna delle interfacce collegate ad una rete). Inoltre, nulla vieta di assegnare ulteriori indirizzi IP alle interfacce di rete per scopi di test o configurazioni specifiche.

La sfida, dunque, è configurare due computer (computer A e computer B) aventi Internet a separarli in modo tale che siano in grado di comunicare conoscendo l'uno l'indirizzo IP dell'altro. Lo strumento grazie al quale tutto questo è possibile è il routing. Il principio di funzionamento è orientativamente questo:

- il computer A che vuole iniziare la comunicazione scopre l'indirizzo IP del computer B magari attraverso il DNS. Ad esempio, 10.11.12.13;
- contestualmente, il computer A scopre, come lo vedremo più avanti, che tale indirizzo non fa parte della propria rete locale e non è quindi in grado di inviare i dati direttamente al destinatario;
- la configurazione di rete del computer A, però, contiene un campo particolare detto default route, che rappresenta l'indirizzo di riferimento da utilizzare qualora l'IP del destinatario non facesse parte della rete locale del mittente. Questo indirizzo corrisponde ad un altro computer sulla rete, detto default gateway;
- il default gateway (che, nel caso di connessioni domestiche, molto probabilmente, è il router ADSL) non è in grado a sua volta di inviare i dati direttamente al destinatario; ciononostante conosce come gestire questo tipo di comunicazione, inviando i dati ad altri apparati di rete (*router*) capaci, dopo una serie di passaggi, di consegnare i dati al destinatario;
- i dati vengono quindi passati di apparato in apparato finchè non arrivano ad un router che riconosce avere l'indirizzo IP 10.11.12.13 all'interno della propria rete locale;
- questo router sa come inoltrare i dati al reale destinatario e procede. Ogni risposta proveniente dal computer B avrà un percorso simile.

Il percorso che i dati compiono per raggiungere il computer di destinazione viene calcolato *durante la trasmissione stessa*. Il mittente non specifica le destinazioni intermedie alle quali inviare i dati, ma confida nel meccanismo per il quale ogni computer partecipante alla comunicazione sarà in grado di gestire i dati nella maniera corretta. A maggior ragione, ogni computer dovrà avere una conoscenza locale della rete e non una estesa a tutto Internet (il che, come è facile intuire, sarebbe impossibile).

Approfondimento: una delle proprietà di IP come protocollo di comunicazione è che il percorso potrebbe variare da pacchetto a pacchetto, rendendo quindi la comunicazione in grado di reagire dinamicamente ad eventuali congestioni o interruzioni di rete.

Approfondimento: questo principio è essenzialmente quello della posta classicamente intesa. Non si ha bisogno di conoscere gli indirizzi di tutti gli uffici postali attraverso i quali la lettera inviata verrà smistata, ma solo quello del destinatario.

Per riassumere, un computer Linux avrà bisogno di tre informazioni:

- *il proprio indirizzo IP;*
- *il gruppo di indirizzi che è in grado di raggiungere direttamente;*
- *l'indirizzo IP del proprio default gateway.*

Il gruppo di indirizzi che il computer è in grado di raggiungere direttamente, ovvero i computer partecipanti alla stessa rete locale alla quale il computer è collegato, vengono ricavati a partire dalla NETMASK o maschera di sottorete.

Approfondimento: Supponiamo, ad esempio, che ad un computer sia assegnato l'indirizzo IP *192.168.178.111* all'interno della rete locale *192.168.1.0/24*. Il */24* rappresenta la maschera di sottorete, ovvero che i primi 24 bit dell'indirizzo (i primi 3 ottetti, *192.168.1*) individuano la rete locale. Questo significa che tutti gli indirizzi IP da *192.168.1.1* a *192.168.1.254* sono raggiungibili direttamente. (Attenzione! *192.168.1.0 e 192.168.1.255 non si possono assegnare ai computer in quanto il primo* identifica la rete stessa e il secondo indica l'indirizzo di broadcast). Per raggiungere reti esterne a quella locale, come già detto, si dovrà passare attraverso il *default gateway*, al quale andrà assegnato un indirizzo appartenente al range precedentemente menzionato.

Generalmente questi parametri fondamentali -*indirizzo IP, maschera di sottorete e default gateway*- sono configurabili manualmente (dipende dalla distribuzione sulla quale si sta lavorando), ma è assai probabile che la propria rete fornisca un server DHCP che automaticamente inoltri tali parametri al computer in uso. Per far partecipare un computer Linux ad una rete locale, quindi, dovrebbe essere sufficiente collegare il cavo Ethernet all'adattatore di rete dedicato o selezionare la rete WiFi desiderata e, nell'eventualità che questa sia protetta, inserire la password corrispondente.

VERIFICA:

- Da quanti bit è composto un indirizzo IPv4?
- Cosa rappresenta l'indirizzo 127.0.0.1?
- Quale componente di rete permette la comunicazione con un computer esterno alla propria rete locale?
- Cosa permette di individuare la porzione dell'indirizzo IPv4 relativa alla rete locale?

15.1.3 Nomi e DNS

Come abbiamo visto gli indirizzi IP sono importanti ma anche poco intuitivi da usare. Sono una sequenza di numeri e in quanto tali facili da dimenticare o confondere. Un indirizzo alfanumerico è molto più facile da gestire per un essere umano.

Un modo per dare dei nomi agli indirizzi IP (e viceversa) è il DNS (*Domain Name System*). DNS è un database distribuito che correla in maniera biunivoca gli indirizzi IP a degli hostname; è possibile accedere a questo servizio tramite degli appositi server, chiamati appunto server DNS, localizzati negli ISP o all'interno delle reti aziendali.

L'indirizzo del server DNS va aggiunto ai tre parametri fondamentali precedentemente menzionati affinchè un computer Linux partecipi ad una rete.

Così come per gli indirizzi IP e i router, nessun server DNS ha conoscenza completa delle associazioni tra hostname e indirizzi IP; i server, infatti, sono organizzati in maniera

gerarchica. Poniamo ad esempio di dover trovare l'indirizzo IP corrispondente all'hostname *pippo.pluto.com*:

- i server DNS *root-level* sono incaricati di risolvere solo la parte finale dell'hostname -nel nostro caso *.com*- e indirizzare la richiesta di risoluzione verso i server DNS che contengono le informazioni relative al dominio risolto;
- questi server DNS leggono la richiesta e, potendo risolvere gli hostname *.pluto.com* indirizzano a loro volta la richiesta ad altri server.
- questi ultimi server DNS hanno nel loro database la corrispondenza biunivoca tra l'hostname *pippo.pluto.com* ed un indirizzo IP e la forniscono al richiedente, completando il processo di risoluzione.

Le informazioni ricavate da questo processo, ovviamente, non vengono scartate immediatamente dopo l'uso ma vengono mantenute dal computer per eventuali successive richieste; ciò non toglie che dopo un certo quantitativo di tempo, queste debbano essere rinnovate.

VERIFICA:

- Quali sono i primi server DNS pubblici ad essere interpellati per una richiesta di risoluzione di un hostname?
- Perchè è stato introdotto il meccanismo del DNS?

15.1.4 IPv6

IP è un protocollo di comunicazione che è stato introdotto circa 30 anni fa e da allora alcune assunzioni sulle quali è stato basato si sono dimostrate piuttosto ingenue. Per esempio, IPv4 (la versione più usata al momento) permette 2^32 ovvero all'incirca 4 miliardi di indirizzi. A causa di alcune limitazioni del protocollo e problemi riguardanti la loro distribuzione, sono rimasti pochi indirizzi disponibili e questo, in un'era dove gli apparati dotati di un'interfaccia di rete stanno aumentando in maniera esponenziale, rappresenta un problema.

Approfondimento: Sono stati sviluppati metodi per alleviare questo problema; per le reti domestiche, a seconda della loro dimensione, vengono utilizzati set di indirizzi IP cosiddetti *privati*, ovvero indirizzi non abilitati a comunicare al di fuori delle reti locali. Per permettere la comunicazione, è necessaria la presenza di un router che implementi il protocollo NAT (*Network Addess Translation*), che traduce gli indirizzi privati in uno o più indirizzi utilizzabili su internet; questo garantisce a tutti i componenti di rete ai quali è assegnato un indirizzo privato di comunicare con elementi esterni alla rete locale.

IPv6, il successore designato di IPv4, è disponibile dal finire degli anni 90; risolve molte delle limitazioni di IPv4 ma gli ISP ne hanno implementato l'utilizzo molto lentamente. Linux lavora molto bene con IPv6 e, visto che è possibile far lavorare IPv4 e IPv6 in parallelo, nulla osta allo strutturare la propria rete -anche domestica, visto che la maggioranza dei router ADSL oramai lo supportano- basandosi su IPv6.

Le proprietà più importanti di IPv6 sono:

- NUMERO DI INDIRIZZI ESTESO - l'indirizzo IPv6 è composto di *128 bit* con l'obiettivo di coprire l'inevitabile espansione delle apparecchiature in grado di collegarsi ad Internet in un prossimo futuro. Gli indirizzi IPv6 vengono rappresentati in 8 gruppi di 4 cifre esadecimali separati da ***:***

  fe80:0000:0000:0000:0a00:27ff:fe0d:844f

Gli zero iniziali possono essere rimossi dai blocchi di 4 cifre

  fe80:0:0:0:0a00:27ff:fe0d:844f

Inoltre, al massimo per una sola sequenza di zero consecutiva, questa può essere sostituita da ***::***

  fe80::a00:27ff:fe0d:844f

L'equivalente per IPv6 dell'indirizzo di *loopback* è ::1;

- ASSEGNAZIONE INDIRIZZI - con IPv4, l'ISP provvede ad assegnare un singolo indirizzo IP o al massimo un set più. Nel caso in cui si abbia la necessità di collegare più computer di quanti sono gli indirizzi assegnati dall'ISP bisogna ricorrere a metodi alternativi come il NAT, descritto in precedenza. Con IPv6, al contrario, viene assegnata direttamente un'intera rete, tramite il prefisso di sottorete (*subnet prefix*), identificata generalmente solo da 48 o 56 bit dei 128 disponibili in un indirizzo IPv6: in questo modo è possibile assegnare fino 2^64 indirizzi per singolo utente.
- SEMPLICITA' DI CONFIGURAZIONE - Se con IPv4 un computer ha bisogno di un indirizzo IP locale -assegnato eventualmente da un server DHCP- con IPv6 un computer può assegnarsi in maniera autonoma un indirizzo coerente che gli permetta di comunicare con altri computer nelle immediate vicinanze. Con IPv6, inoltre, un computer è in grado di localizzare, senza l'uso del DHCP, i router in grado di inoltrare pacchetti al di fuori della rete locale. Questo risolve alcuni problemi inerenti all'uso del DHCP; IPv6, di conseguenza, *non usa una default ruote*.
- ALTRI MIGLIORAMENTI - il formato dei pacchetti IP è stato cambiato per permettere un routing più efficiente. IPv6 supporta nativamente l'utilizzo di metodi di protezione crittografica dei dati (*IPSec*) e la mobilità (*Mobile IP*, permette la mobilità dei terminali tra reti differenti mantenendo le connessioni precedentemente stabilite)
- COMPATIBILITA' - L'introduzione di IPv6 inferisce solo sul livello IP dello stack TCP/IP o sul ivello direte della pila ISO/OSI; protocolli di livello superiore come TCP o UDP rimangono inalterati. E' possibile, come accennato in precedenza, implementare parallelamente IPv4 e IPv6

VERIFICA:

- Per quale motivo è stato sviluppato NAT?

- L'indirizzamento IPv6 prevede una *default route*? Motivare la risposta.
- Qual è l'indirizzo di loopback per IPv6?

15.2 Linux come un client di rete

15.2.1 Requisiti

Abbiamo dunque delineato le componenti necessarie affinché un computer Linux possa far parte di una rete (IPv4):

- indirizzo IP
- maschera di sottorete
- indirizzo del default gateway
- indirizzi dei server DNS

Nel migliore dei casi il computer otterrà automaticamente queste informazioni tramite DHCP all'accensione, quando viene connesso il cavo di rete o quando ci si connette ad una rete wireless. Qualora questo non avvenisse, saremo costretti a configurare il computer manualmente. I dettagli variano a seconda della distribuzione:

- Su *Debian GNU/Linux e i suoi derivati*, le configurazioni di rete sono memorizzate in un file chiamato /etc/network/interfaces. Un esempio commentato è memorizzato in /usr/share/doc/ifupdown/examples/network-interfaces.gz
- Su distribuzioni basate su *SUSE*, i parametri di rete vengono configurati principalmente tramite YaST. Alternativamente, i file di configurazione sono in /etc/sysconfig/network
- Sulle distribuzioni *RedHat e i suoi derivati* i file di configurazione sono nella cartella /etc/sysconfig/network-scripts

Indipendentemente dalla distribuzione utilizzata, linux assegna ad ogni scheda di rete un nome formato dal prefisso *eth* seguito da un progressivo, ad es eth0 per la prima scheda di rete.

15.2.2 La linea di comando e le configurazioni di rete

ifconfig - è il comando principale per le configurazioni di rete: è in grado di associare un indirizzo IP e una maschera di sottorete ad un'interfaccia. Per configurare i parametri base tramite terminale il comando ifconfig può essere usato come segue:

```
# ifconfig eth0 10.0.2.15 netmask 255.255.255.0
```

In questo modo assegnamo l'indirizzo *10.0.2.15* all'interfaccia di rete *eth0* (Ethernet). Specificando la maschera di rete 255.255.255.0 siamo in grado di ricavare che l'indirizzo di rete di è *10.0.2.0/24*.

route - è il comando che gestisce la *tabella di routing* del nostro computer e con il quale è possibile configurare il *default gateway.* Un esempio di configurazione è il seguente:

```
# route add -net default gw 10.0.2.1
```

Attenzione: questa configurazione sarà valida solo fino al prossimo riavvio del computer.

/etc/resolv.conf - è il file dove sono solitamente memorizzati:

- gli indirizzi dei server DNS (al massimo tre - *nameserver*),
- i domini che verranno utilizzati nelle ricerche quando l'utente li ometterà dall'hostname da risolvere (*search*)
- il dominio al quale appartiene il computer (*domain)*

 # Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
 # DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN

domain lpi.edu
search prova.com prova.org
nameserver 10.0.2.1
nameserver 10.0.2.100

(In questo esempio *prova.com* e *prova.org* verranno aggiunti a tutti i nomi che verranno specificati senza essere preceduti dal punto. Per esempio *www* risulterà in una richesta di risoluzione www.prova.com e www.prova.org)

Approfondimento: Quando la configurazione viene effettuata automaticamente, ad esempio quando ci si connette ad una rete wireless, il file */etc/resolv.conf* viene sovrascritto. Controllare sulle specifiche della propria distribuzione dove memorizzare permenentemente le informazioni sui server DNS

dhclient e dhcpd - sono i comandi relativi rispettivamente al client DHCP ed al server DHCP presenti sul computer. Per avviare una configurazione DHCP come client (che spesso avviene in automatico al collegamento alla rete) è sufficiente lanciare il comando *dhclient* da terminale.

VERIFICA:

- Scrivere i comandi necessari per configurare da terminale l'interfaccia Ethernet 0 del computer con l'indirizzo *192.168.1.2/24*. Il default gateway è *192.168.1.1*
- Quale campo del file */etc/resolv.conf* informa sugli indirizzi IP dei server DNS?

15.2.3 Troubleshooting

Se le procedure base per collegarsi ad Internet (collegare il cavo di rete Ethernet o accedere ad una rete wireless) non funzionano o emergono altre tipologie di problemi come, ad esempio, lunghi ritardi durante la navigazione web o connessioni interrotte è buona pratica consultare il proprio amministratore di rete.

E' possibile identificare ed analizzare, però, grazie all'uso di alcuni strumenti implementati nelle distribuzioni Linux, i problemi di base evitando di scomodare l'amministratore di rete.

ifconfig

Abbiamo già introdotto il comando *ifconfig* in relazione alle configurazioni di rete, ma questo può essere usato anche per interrogare il sistema riguardo la configurazione dell'interfaccia di rete

```
$ /sbin/ifconfig eth0

eth0 Link encap:Ethernet HWaddr 08:00:27:0d:84:4f
indirizzo inet:10.0.2.15 Bcast:10.0.2.255 Maschera:255.255.255.0
indirizzo inet6: fe80::a00:27ff:fe0d:844f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:24918 errors:0 dropped:0 overruns:0 frame:0
TX packets:21465 errors:0 dropped:0 overruns:0 carrier:0
collisioni:0 txqueuelen:1000
Byte RX:13256467 (13.2 MB) Byte TX:4633982 (4.6 MB)
```

- la prima riga dell'output contiene l'indirizzo MAC dell'interfaccia interrogata (*HWaddr*). E'un indirizzo unico composto da 6 byte che identifica l'hardware e viene assegnato dal produttore. (in questo caso, un'interfaccia Ethernet)
- la seconda riga dell'output contiene l'indirizzo IPV4 (inet) assegnato all'interfaccia, seguito dall'indirizzo di broadcast e dalla maschera di sottorete
- la terza riga dell'output specificano l'indirizzo IPV6 (inet6).

UP all'inizio della quarta riga indica che l'interfaccia è effettivamente accesa.

ping

E' possibile usare il comando `ping` per verificare la connettività tra computer a livello IP. `ping` utilizza il protocollo di controllo ICMP per verificare l'effettiva possibilità di raggiungere un altro computer e il tempo impiegato per raggiungerlo. Se la verifica ha esito positivo questo significa che a) il computer dal quale è stato lanciato il comando `ping` è in grado di inviare dati al destinatario scelto; b) tale destinatario è in grado a sua volta di inviare dati al computer dal quale stiamo lavorando. Nel caso più semplice si invoca il comando `ping` seguito dall'indirizzo IP del computer con il quale vogliamo comunicare:

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=63 time=1.99 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=63 time=1.71 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=63 time=2.02 ms
...<interruzione tramite control+c>....
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.711/1.910/2.020/0.140 ms
```

Questo esempio dimostra un caso in cui il comando *ping* dà esito positivo. I tre pacchetti inviati sono tornati indietro, in sequenza e con tempi ragionevoli per una rete locale (circa 2 ms).

Se non si è in grado di ottenere una connessione ad un altro computer, il problema, teoricamente, può risiedere ovunque tra la sorgente e la destinazione. Per un approccio sistematico è bene seguire le seguenti procedure:

- usare ping per verificare se l'interfaccia di loopback 127.0.0.1 (IPv4) è raggiungibile. In caso contrario, il problema potrebbe essere il proprio computer
- usare ping per verificare se l'indirizzo che l'interfaccia del proprio computer sta usando per connettersi ad Internet è raggiungibile. Questo è reperibile tramite il comando:

$ /sbin/ifconfig eth0

(*eth0* si riferisce alla prima interfaccia Ethernet; questo è un parametro da modificare nel caso in cui si vogliano testare altre interfacce)

- usare ping per verificare se il *default gateway* è raggiungibile o meno. Nel caso non si sia a conoscenza dell'indirizzo del *default gateway* è possibile ottenerlo usando il comando *route*

Nel caso in cui un ping verso il *default gateway* restituisca messaggi come

Destination Host Unreachable

ci potrebbe essere in problema di configurazione della rete locale. Va verificato, comunque, che l'entry relativa al default gateway sul proprio computer corrisponda effettivamente al hostname/indirizzo IP assegnato al default gateway dall'amministratore di rete.

- Nel caso in cui si riesca a raggiungere correttamente il proprio *default gateway*, il problema potrebbe risiedere al di fuori della LAN presso la quale siamo collegati o in qualche altro livello dello stack TCP/IP o della pila ISO/OSI. In questo caso è richiesto l'intervento dell'amministratore di rete.

ping -f con il parametro -f , ping viene forzato ad inviare pacchetti il più velocemente possibile invece che alla frequenza di default di uno al secondo. Ogni pacchetto inviato verrà rappresentato da un punto (".") mentre ogni pacchetto ricevuto verrà rappresentato da un trattino ("-"). Una connessione poco efficiente avrà lunghe sequenze di punti come output per questo comando.

Approfondimento se non si è utenti *root* i pacchetti vengono inviati con un intervallo di 0.2 secondi l'uno dall'altro. Questo metodo inonda la rete di pacchetti ICMP, degradandone le prestazioni, di conseguenza l'utilizzo è riservato agli amministratori di rete.

Per verificare le connessioni IPv6 è necessario usare il comando `ping6` al posto di ping

$ ping6 ::1

I parametri a disposizione per il comando *ping* sono molteplici e permettono una moltitudine di strumenti per verificare la connettività della propria rete. Si rimanda al man relativo per ulteriori approfondimenti.

route

Abbiamo già incontrato il comando *route* quando abbiamo affrontato le configurazioni di rete e, nello specifico, quando dovevamo configurare il *default gateway*. Lanciandolo, però, da terminale senza alcun parametro ci permette di osservare la tabella di routing del nostro computer.

```
$ /sbin/route
Tabella di routing IP del kernel
Destination Gateway Genmask Flags Metric Ref Use Iface
default 10.0.2.2 0.0.0.0 UG 0 0 0 eth0
10.0.2.0 * 255.255.255.0 U 1 0 0 eth0
link-local * 255.255.0.0 U 1000 0 0 eth0
```

L'entry relativa alla destinazione *default* rappresenta il nostro default gateway (nel caso rappresentato *10.0.2.2*). L'output del comando route specifica, inoltre, per ciascuna entry, l'interfaccia di uscita (eth0, lo, etc.)

dig

il comando *dig* viene usato per testare le risoluzioni DNS. Al netto di eventuali parametri, il comando esegue la ricerca del IP corrispondente al nome fornito come input

```
$ dig www.lpi.org

; <<>> DiG 9.8.1-P1 <<>> www.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38547
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.lpi.org. IN A

;; ANSWER SECTION:
www.lpi.org. 3600 IN A 69.90.69.231

;; Query time: 183 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Aug 14 16:28:25 2012
;; MSG SIZE rcvd: 45
```

L'output specifica nella sezione denominata *QUESTION SECTION* il nome da risolvere mentre nella *ANSWER SECTION* vengono specificati i passaggi operati per la risoluzione (nel caso rappresentato www.lpi.org corrisponde ad un computer che ha come indirizzo IP 69.90.69.231).

L'ultima sezione dell'output specifica il tempo (183 ms) e il server (127.0.0.1) impiegato per elaborare la richiesta.

In caso di eventuali errori nella risoluzione degli hostname, verificare se tutto è configurato correttamente sul file ***/etc/resolv.conf*** o sul server DNS.

E' possibile inviare richieste di risoluzione di nomi a server DNS specifici in questo modo:

```
$ dig www.lpi.org @127.0.0.1
```

Per eseguire l'operazione inversa, ovvero abbinare hostname ad indirizzi IP a partire da questi ultimi, è sufficiente utilizzare il parametro ***-x***:

```
$ dig +short -x 31.24.175.68
s0a.linupfront.de
```

(in questo esempio, il parametro `+short` permette di ottenere un output meno prolisso). Le funzionalità di *dig* non si fermano a questi esempi ma la descrizione estesa esula dagli obiettivi di questo testo, pertanto si rimanda al manuale.

Vi sono ulteriori comandi in grado di testare le funzionalità del servizio DNS quali ***`host`*** e nslookup. Mentre il primo permette una verifica base della risoluzione di un hostname limitandosi a fornire il corrispondente indirizzo IP, l'uso del secondo viene sconsigliato; è nei piani degli sviluppatori rimuoverlo dall'uso comune in favore degli altri due sebbene al momento sia comunque disponibile.

netstat

il comando `netstat` è utile per una molteplicità di operazioni riguardanti le funzionalità di rete del nostro computer. Eseguire da terminale

```
$ netstat
```

fornisce la lista di tutte le connessioni di rete attive. Queste non includono solo le connessioni TCP, ma anche connessioni locali attraverso i socket Unix.

Approfondimento i protocolli TCP e UDP permettono al computer di offrire e utilizzare differenti servizi contemporaneamente ognuno dei quali viene identificato da una *porta;* per una lista delle porte, consultare il file */etc/services.*

Parametri di `netstat`:

- -tl ottenere la lista di tutte le porte sulle quali servizi TCP sono in attesa di connessioni entranti. È uno strumento importante nel troubleshooting di una connessione con servizi di rete. Se un servizio configurato sulla rete non appare nell'output del comando, c'è un probabile errore nella sua configurazione (controllare gli indirizzi e le porte assegnate) o il servizio stesso non è attivo.

```
$ netstat -tl
Connessioni internet attive (solo server)
Proto Recv-Q Send-Q Indirizzo locale Indirizzo esterno Stato
tcp 0 0 localhost:domain *:* LISTEN
```

```
tcp 0 0 localhost:ipp *:* LISTEN
tcp 0 0 *:ssh *:* LISTEN
```

Nell'esempio riportato osserviamo che il computer in esame fornisce un servizio DNS (identificato da *localhost:domain)*, un server CUPS per le stampanti (`localhost:ipp`) . Inoltre per tutti gli indirizzi (identificati da "*") viene fornito un servizio `ssh.`

- -u permette a *netstat* di fornire i servizi basati su UDP, mentre il parametro "-p" permette di fornire il nome e il PID del processo che supporta il servizio. Quest'ultima funzione è disponibile solo se il comando viene lanciato come utente *root*.
- -n fornirà un output dove al posto dei nomi verranno visualizzati gli indirizzi IP e i numeri di porta. Questo potrebbe fornire un ulteriore aiuto a chi ha conoscenza dei numeri di porta utilizzati nella rete.
- -s permetterà al comando di fornire statistiche riguardanti le prestazioni della rete, ad esempio

```
$ netstat -s
Ip:
30685 pacchetti totali ricevuti
2 con indirizzi non validi
0 inoltrati
0 pacchetti entranti scartati
30683 pacchetti entranti consegnati
29760 richieste inviate
2 dropped because of missing route
Icmp:
291 messaggi ICMP ricevuti
17 messaggi ICMP in input falliti.
Istogramma input ICMP:
risposte di echo: 274
651 messaggi ICMP inviati
0 messaggi ICMP falliti
Istogramma output ICMP:
richiesta di echo: 651
IcmpMsg:
InType0: 274
OutType8: 651
Tcp:
3643 connessioni attive aperte
0 connessioni passive aperte
1 tentativi di connessione falliti
[...]
Udp:
2845 pacchetti ricevuti
```

0 pacchetti ricevuti su porta sconosciuta.
0 errori sui pacchetti ricevuti
3035 pacchetti inviati
UdpLite:
TcpExt:
220 TCP sockets finished time wait in fast timer
64 delayed acks sent
2555 packets directly queued to recvmsg prequeue.
[...]

Altri parametri:

Parametro	Funzione
-r	simile al comando route
`-i`	simile al comando ipconfig. Fornisce informazioni sulle interfacce di rete
-m	Informazioni sulle funzionalità NAT implementate sul computer
-p	Lista dei programmi che stanno accedendo alle funzionalità di rete

VERIFICA:

- Un computer in uso è correttamente connesso fisicamente alla infrastruttura di rete, tuttavia non riesce a navigare. Quali sono i comandi da lanciare via terminale per rispettivamente verficare: a) la configurazione di rete sull'interfaccia usata, b) la connettivita' con il default gateway, c) la tabella di routing del computer.
- Qual è l'indirizzo IP di *www.google.com*?
- Quale comando devo lanciare per verificare se il servizio DNS è attivo sul computer in esame e il relativo ID di processo?

Comandi trattati nel capitolo

ifconfig - configurazione delle interfacce di rete

route - gestione della tabella di routing del sistema

dig - risoluzione delle query DNS

netstat - informazioni riguardanti le connessioni di rete

ping/ping6 - verifica della connettività di rete di base (IPv4/IPv6)

dhclient - configurazione tramite protocollo DHCP

dhcpd - avvio del servizio server DHCP

ithum®

Parte 5:
Sicurezza e Permessi dei File

ISBN: 9788866850090

16 Sicurezza base ed identificazione dei tipi di utenti

Nei sistemi multiutenza e quindi anche in Linux occorre autenticarsi per accedere. Questo avviene di solito con l'inserimento di un nome utente (username) e di una password. Ciò consente al sistema di riconoscere l'utente e di assegnargli i diritti di accesso corretti.

Spesso in un sistema Linux ci si può trovare davanti a uno schermo nero (console) con un cursore che lampeggia dopo una scritta login: il che sta a significare che il sistema si aspetta un nome utente riconosciuto e una password che non verrà visualizzata durante la digitazione.

Se invece il login è di tipo grafico, l'aspetto può essere più completo a al login si potrà scegliere a quale tipo di sessione si vorrà accedere.

Una volta ottenuto l'accesso verranno concessi i diritti che l'amministratore 'root' (o superuser) avrà assegnato. Ogni utente dispone di una 'home', dove all'interno di essa potrà gestire i propri file e directory e organizzarli secondo una struttura ad albero.

La memorizzazione della combinazione utente e password è conservata all'interno dei file /etc/passwd e /etc/shadow (in forma crittografata), mentre i gruppi vengono gestiti dal file /etc/group.

L'accesso al sistema è consentito dal meccanismo di comparazione tra la password digitata e la corrispondenza del relativo contenuto nel file /etc/passwd prima e /etc/shadow poi.

In quest'ultimo la password naturalmente non viene conservata in chiaro, ma criptata con algoritmi Crypt o MD5, in esso inoltre vengono memorizzate informazioni riguardo alla scadenza o validità dell'utenza. Questo file è leggibile solo da root per ovvie ragioni di sicurezza.

Nel file /etc/passwd, oltre allo username, sono memorizzati sia la descrizione dell'utente che la shell assegnata (di solito /bin/bash), quindi all'accesso potremo usare immediatamente i comandi interni della shell 'bash', oltre a tutti quelli riconosciuti nel PATH di ambiente, ovvero i files binari contenuti in vari percorsi tra i quali /bin, /usr/bin, /home/utente/bin, /usr/local/bin.

I sistemi Linux permettono l'accesso al sistema contemporaneamente, alcuni comandi che ci fanno comprendere con quale utente stiamo lavorando e chi altri lo sta facendo sono:

- **w –** mostra chi è loggato e cosa sta facendo
- **who** - mostra chi è loggato
- **id –** stampa identificativo utente e gruppo di appartenenza

Se diamo un'occhiata alla riga del file /etc/passwd, dopo lo username troveremo i numeri ad esso associati dell'identificativo (ID) e gruppo (GID), e altri campi descrittiivi,come in questo esempio:

utente:x:1000:1000:Nome
Cognome,12,3456,12345,Altro:/home/utente:/bin/bash

⟨user name⟩: ⟨password⟩: ⟨UID⟩: ⟨GID⟩: ⟨GECOS⟩: ⟨home directory⟩: ⟨shell⟩

Utente:x:ID:GID:Nome completo, Stanza, Numero telefonico di lavoro, Numero telefonico di casa, Atro

Nel file /etc/shadow invece vengono conservate la password criptata e le informazioni su validità e scadenza dell'account:

utente:$6$7sldslkk4kl5klklk4l54l5apfdf:15456:0:99999:7:::

L'utente root è dichiarato invece nella prima riga di passwd e shadow e ha ID e GID 0:0.

Con l'utente 'root' si intende l'amministratore del sistema, esso può rimuovere e aggiungere utenti normali, assegnare priviliegi speciali e intervenire su ogni parte del filesystem o eseguire comandi con accesso privilegiato alle periferiche. In Linux questo è possibile in modo molto semplice poiché le periferiche sono sempre puntati a livello hardware da un file contenuto quasi sempre nella directory /dev

Un altro elemento importante di questo semplice e robusto meccanismo è l'appartenenza ad un gruppo piuttosto che un altro. La nostra presenza nei vari gruppi è visualizzata dall'output del comando "groups". Vedremo infatti che gli utenti normali avranno solo un gruppo "users" o uno con lo stesso nome utente.

I gruppi amministrativi sono assegnati per gradi e permettono l'accesso a periferiche senza usare necessariamente usare l'utenza di root. I privilegi di superutente infatti potrebbero compromettere anche involontariamente l'integrità dell'intero sistema.

Nelle recenti distribuzioni, si fa ricorso a un programma particolare che permette l'assegnazione di priviligi elevati ad alcuni utenti per fare in modo di usare il meno possibile l'utente root e quindi di mantenere riservata la password, ove esista, dell'amministratore.

Questo è possibile con l'utilizzo di 'sudo' che attraverso il file /etc/sudoers garantisce l'esecuzione di uno o più comandi privilegiati con l'inserimento della password dell'utente in questione.

Permette ai membri del gruppo sudo di eseguire tutti i comandi

%sudo ALL=(ALL:ALL) ALL

Quando siamo loggati con il nostro utente possiamo, conoscendo la password, passare a un altro utente con il comando su.

Nel caso di voler ereditare anche l'ambiente di quest'ultimo potremo aggiungere il trattino '-' come negli esempi di seguito:

```
$ id
uid=1000(utente) gid=1000(utente) gruppi=1000(utente)
su – utente1 (viene richiesta la password di 'utente1)
$id
uid=1003(utente1) gid=1003(utente1) gruppi=1003(utente1)
```

Stabilito quindi con quale utenza siamo loggati attraverso il comando 'id' useremo il comando 'w' per vedere chi altri è presente, come in questo esempio:

```
$ w
19:50:38 up 8:31, 2 users, load average: 0,32, 0,17, 0,09
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
utente tty2 19:50 15.00s 0.35s 0.34s -bash
utente1 tty1 19:50 5.00s 0.36s 0.34s -bash
```

Il comando 'who' ci riporta invece un'informazione più semplice:

```
$ who
utente tty2 2012-09-02 19:50
utente1 tty1 2012-09-02 19:50
```

A titolo di cronaca, esiste anche un comando 'whoami' (chi sono io) che da come risultato il solo nome utente, cosa che sembrerà banale, ma a volte è necessaria.

Esistono poi diversi utenti di sistema, ognuno con un ruolo specifico e che quasi mai ha una password valida. Sono ovviamente utenti che garantiscono, per il ruolo che compete loro, di mantenere in memoria i processi relativi e far accedere i programmi e gli utenti alle periferiche. Li troviamo elencati nel file /etc/passwd, un elenco parziale è:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
….
```

Dopo il nome utente troviamo una 'x' che non è altro il campo password. Nei vecchi sistemi *nix qui veniva inserita la password criptata e quindi se una volta letta poteva venire ricavata quella reale con meccanismi di attacco sull'hashing o basati su dizionari di parole.

I meccanismi attuali invece usano la forma shadow password che come riportato prima è leggibile solo da root. In realtà l'autenticazione è complessa, regolata spesso da moduli o plugin che usano PAM (Pluggable Authentication Module)

Il file /etc/group contiene per ogni riga il nome del gruppo e gli utenti ad esso appartenente separati da virgola, come in questo stralcio:

```
adm:x:4:utente
cdrom:x:24:utente
sudo:x:27:utente
dip:x:30:utente
plugdev:x:46:utente
lpadmin:x:109:utente
utente:x:1000:
sambashare:x:124:utente
users:x:500:utente,utente1,utente2
```

L'amministratore potrà aggiungere nuovi gruppi a aggungere nuovi utenti a un gruppo esistente attraverso l'uso dei comandi:

```
groupadd ufficio
```

Crea un nuovo gruppo 'ufficio'

```
adduser utente ufficio
```

Aggiunge 'utente' al gruppo 'ufficio'.

Esercizi

- usare i comandi w, who, id per 'identificazione del proprio utente e degli utenti connessi
- usare il comando groups per controllare i gruppi di appartenza
- provare a leggere il file /etc/password
- provare a leggere il file /etc/group
- provare a leggere il file /etc/shadow

Domande

1. Con quale comando si capisce qual'è 'utente che si sta usando?
2. Quale è il comando per vedere chi è connesso al sistema?
3. In quale file è memorizzata la password criptata?
4. Quali sono i numeri identificativi di utente e gruppo?
5. Dove sono memorizzate le informazioni sull'utente?
6. In quale file è specificato il gruppo di appartenza?

7. Come si chiama l'ammistratore con id=0?
8. Come si ottiene l'elenco dei gruppi di appartenenza?
9. In quale file è dichiarato l'utente che può usare 'sudo'?

ithum®

17 Creare utenti e gruppi

Il concetto di utenti e gruppi è di fondamentale importanza per la gestione di sistemi operativi in grado di gestire l'utilizzo delle risorse di una medesima macchina da parte di più soggetti; tali sistemi operativi vengono definiti multiutente e Linux è uno di questi.

17.1 Gruppi e Utenti

Un utente può rappresentare una persona in carne ed ossa che ha accesso al sistema, oppure può essere utilizzato da specifiche applicazioni o servizi quando questi vanno in esecuzione. I gruppi invece sono una collezione di utenti raggruppati in base ad una qualche logica: gli impiegati di un'azienda, gli studenti di un corso, etc.

E' importante sapere che ad ogni utente è associato un identificativo numerico unico nel sistema chiamato *user id* (spesso riferito con l'acronimo *UID*). E' il sistema che decide il valore dell'identificativo, l'unica costante è rappresentata da quello dell'utente root che ha sempre valore pari a zero. Solitamente gli UID partono dal numero 500, altre volte da 1000, in ogni caso la convenzione prevede che gli UID di valore più basso siano riservati agli utenti associati a programmi e servizi. Per vedere qual'è il valore del proprio UID, una volta loggati basta eseguire dal terminale:

```
$ echo $UID
```

Anche i gruppi possiedono un identificativo numerico univoco chiamato *group id* e riferito con l'acronimo *GID*. Alcune distribuzioni utilizzano il concetto di Gruppo Privato, spesso riferito con l'acronimo *UPG, User Private Group*: in pratica ogni qualvolta viene creato un nuovo utente il sistema crea anche un nuovo gruppo con lo stesso nome e vi aggiunge unicamente quell'utente. Questo accorgimento spesso semplifica le operazioni di gestione dei permessi per file e cartelle.

Un utente può appartenere a più di un gruppo (per tornare agli esempi precedenti, si immagini uno studente che frequenta due corsi) ma uno solo è il cosiddetto *Gruppo Principale* (o *Main Group*). Il gruppo principale è quello che determina i permessi alla creazione di file e cartelle. Per vedere a quali gruppi appartiene il proprio utente, una volta loggati basta lanciare dalla shell il comando:

```
$ groups
```

Se il nostro utente appartiene a più gruppi questi verranno elencati tutti. Il gruppo principale è il primo della lista. Esiste poi un comando che mostra tutte le informazioni relative ad utenti e gruppi fin qui trattate:

```
$ id [nomeutente]
```

Il comando *id* stampa a terminale il l'UID dell'utente specificato dal parametro *nomeutente*, il gruppo principale e relativo GID, la lista dei gruppi e GID cui appartiene

l'utente. Se non si passa alcun argomento al comando verranno visualizzate le informazioni relative al proprio utente.

17.2 Shadow password

Quando il sistema operativo offre un ambiente condiviso da più utenti la sicurezza e la riservatezza dei dati sensibili risultano fondamentali. Tradizionalmente nei sitemi Linux tutte le informazioni relative ai singoli utenti vengono memorizzate in un file di testo in formato ASCII (quindi consultabile anche dalla shell). Ogni riga del file rappresenta un utente ed ogni riga ha questo formato:

Nome:Password:UserID:GruppoPrincipale:NomeCompleto:HomeDirectory:Shell

Ovviamente il campo Password non contiene la password in chiaro bensì il suo hash, ovvero una forma cifrata a partire dalla quale non è possibile risalire alla password originale. Al posto dell'hash delle password, nei sistemi più recenti possiamo trovare la lettera *x* oppure il carattere *!*, il che sta ad indicare che sono in uso le cosiddette shadow password, una tecnica volta ad aumentare la sicurezza dei dati degli utenti attraverso diversi meccanismi e strategie, in particolare:

- Gli hash delle password non si trovano nel file /etc/passwd che è accessibile a tutti gli utenti ma si trovano in /etc/shadow, accessibile dal solo utente root.
- Vengono gestite le date di scadenza delle password degli utenti, in modo da garantirne il rinnovamento periodico

Il file /etc/shadow, analogamente a /etc/passwd è in formato ASCII e contiene una riga per ogni utente e le informazioni sono separate da un carattere ':' secondo questo formato:

Nome:Password:UltimoCambio:CambioMin:CambioMax:GiorniAvviso:GiorniInattivo:DataScadenza:NonUtilizzato

I campi specificano da sinistra a destra il nome utente, la password (offuscata secondo i criteri visti per il file /etc/passwd), la data dell'ultimo cambio password effettuato, i giorni che dovranno trascorrere prima di un ulteriore cambio password, i giorni che mancano prima che verrà richiesto dal sistema di cambiare la password, quanti giorni prima il sistema avviserà l'utente della scadenza della password, quanti giorni dopo la scadenza della password l'account rimmarrà attivo, la data di scadenza dell'account.

Sebbene ogni distribuzione Linux offra dei tool grafici per la gestione di gruppi ed utenti, allo stesso scopo esistono diversi comandi che possono essere lanciati dalla shell che possono risultare molto utili per automatizzare alcuni processi o quando non è disponibile una sessione desktop.

17.3 Aggiungere un nuovo utente

Per creare un nuovo utente va eseguito da root il comando:

```
# useradd [opzioni] nomeutente
```

Le opzioni principali da che il comando *useradd* accetta sono riassunte nella tabella seguente:

Opzione	Descrizione
-c '*commento*'	'*commento*' può essere una stringa qualunque, generalmente viene specificato il nome completo dell'utente.
-d *home_directory*	la home directory associata all'utente, se questa opzione non viene specificata il default è /home/nomeutente.
-e *data*	la data di scadenza dell'account, espressa nel formato YYYY-MM-DD (anno, mese, giorno).
-f *giorni*	una volta scaduta la password, il numero di giorni trascorsi i quali viene disattivato anche l'account.
-g *nomegruppo*	il gruppo principale dell'utente. Il gruppo deve già esistere prima di eseguire il comando.
-G *lista_gruppi*	una lista di gruppi separati da virgola cui assegnare l'utente. Tutti i gruppi della lista devono esistere prima di eseguire il comando.
-m	crea la home directory se non esiste già
-M	non creare la home directory
-N	non creare lo User Group privato per l'utente

L'utente appena creato solitamente non è attivo e non possiede una password, per cui il secondo passo consiste proprio nell'assegnargliene una:

```
# passwd nomeutente
```

A questo punto, se il sistema implementa il concetto di User Private Group sarà stato creato anche il gruppo nomeutente, che sarà il gruppo principale dell'utente appena creato. Possiamo vedere nel dettaglio cosa succede quando vengono utilizzati i comandi di creazione degli utenti, ad esempio:

```
# useradd alice -c Alice
```

per prima cosa viene aggiunta questa linea nel file /etc/passwd (per il significato dei singoli campi si rimanda al paragrafo precedente):

```
alice:x:1001:1001:Alice:/home/alice:/bin/bash
```

la presenza della x in corrispondenza del campo password è segno che il sistema utilizza le shadow password. E infatti subito dopo viene aggiunta questa linea nel file /etc/shadow (anche qui, per il significato dei singoli campi si rimanda al paragrafo precedente):

```
alice:!!:15577:0:99999:7:::
```

la presenza del doppio punto esclamativo (!!) nel secondo campo della riga sta a significare che l'utente non ha una password assegnata ed attualmente non può loggarsi nel sistema, mentre il valore *99999* in corrispondenza del campo *CambioMax* sta a significare che la password di fatto non ha una scadenza. Se poi eseguiamo da terminale il comando:

```
$ id alice
uid=1001(alice) gid=1001(alice) groups=1001(alice)
```

a seconda dell'output potremmo notare che contenstualmente è stato creato un gruppo con lo stesso nome (come in questo caso), segno che il sistema utilizza i gruppi privati. Per finire, possiamo notare che è stata creata nel filesystem una directory che diventerà la home dell'utente alice:

```
$ ls -l /home
drwx------. 4 alice alice 4096 Aug 25 19:39 alice
```

A questo punto l'utente esiste nel sistema anche se non può ancora loggarsi: infatti l'amministratore dovrà specificare una password per il primo accesso con il comando *passwd*, consentendo così all'utente di fare login e settare una password di propria scelta.

Per modificare le proprietà di un utente già presente nel sistema si può utilizzare il comando usermod. Ad esempio se si vuole modificare la shell utilizzata dall'utente alice da bash a csh si può eseguire da root il comando:

```
# usermod -s /bin/csh alice
```

17.4 Aggiungere un nuovo gruppo

Si può aggiungere un nuovo gruppo al sistema tramite il comando groupadd eseguito da root:

```
# groupadd [opzioni] nomegruppo
```

Le opzioni principali da che il comando *groupadd* accetta sono riassunte nella tabella seguente:

Opzione	Descrizione
-f, --force	quanto utilizzato insieme all'opzione -g, crea comunque il gruppo scegliendo un diverso GID
-g *group_id*	crea il nuovo gruppo utilizzando il GID specificato, che deve avere valore maggiore di 499
-o, --non-unique	consenti di avere dei gruppi con lo stesso nome

Tutte le informazioni sui gruppi vengono conservate nel file /etc/group, del tutto simile ai file /etc/passwd e /etc/shadow. Il file è in formato ASCII e contiene una riga per ogni gruppo nel sistema, secondo questo formato:

Nome:Password:GID:Utenti

Da sinistra a destra, il prino campo contiene il nome del gruppo, il secondo contiene l'hash della password del gruppo, poi viene specificato l'identificativo del gruppo ed infine la lista degli utenti che ne fanno parte, separati da virgola. Il concetto di password del gruppo è raramente utilizzato nella normale gestione del sistema. Quando viene specificato, di fatto un utente può autonomamente entrare a far parte di un certo gruppo a patto che ne conosca la password.

Una volta creato un gruppo, è possibile modificarne le proprietà attraverso il comando groupmod, ad esempio per cambiare il nome al gruppo *studenti* è sufficiente eseguire da root:

groupmod -*n nuovonome* studenti

17.5 Rimuovere utenti e gruppi

Per completare gli strumenti base per la gestione di utenti e gruppi, elenchiamo i comandi necessari per rimuoverli dal sistema. Per rimuovere un utente si può eseguire da root il comando:

userdel *nomeutente*

Se eseguito senza alcuna opzione, il comando rimuove l'utente e tutte le informazioni ad esso collegate ma non la sua home directory. Per rimuovere l'utente e contestualmente rimuovere dal filesystema la sua home directory occorre eseguire il comando con l'opzione -*r*:

userdel -r *nomeutente*

Analogamente, per rimuovere un gruppo basta eseguire da root il comando:

groupdel *nomegruppo*

Il comando non elimina gruppi che risultino essere il gruppo principale di un utente presente nel sistema, nel qual caso per procedere occorre prima eliminare l'utente oppure cambiare a quell'utente il gruppo principale.

ithum®

18 Gestione delle autorizzazioni e della proprietà del file

Linux, come la maggior parte degli attuali sistemi operativi, è un sistema operativo multiutente. Per garantire la riservatezza dei propri dati e assicurare che un utente non possa accedere ai file o alle directory di un altro utente e modificarli o cancellarli, sia accidentalmente che intenzionalmente, Linux utilizza le funzionalità del proprietario e dei permessi.

18.1 Proprietari di file/directory

Ogni file è associato ad un solo proprietario, abitualmente l'utente che ha creato il file, ed un solo gruppo, il gruppo primario dell'utente. Tale associazione avviene mediante la memorizzazione dello UID (*user ID number*) e del GID (*group ID number*). Come verrà descritto in seguito, i permessi dei file possono essere specificati indipendentemente per il proprietario del file, per il gruppo del file e per tutti gli altri utenti. Il proprietario esiste anche per i programmi in esecuzione. Solitamente un programma quando viene avviato crea un processo il cui proprietario è l'utente che lo ha eseguito. L'utente root può cambiare il proprietario ed il gruppo di qualsiasi file e directory, mentre il proprietario del file o della directory può modificarne esclusivamente il gruppo con un altro a cui l'utente deve necessariamente appartenere.

Il comando per modificare il proprietario di un file o di una directory è *chown* (che sta per *change file owner and group*). La sintassi del comando prevede il nome del nuovo proprietario seguito dal nome del file di cui modificare il proprietario:

```
chown [<options>] [<owner>][:[<group>]] <filename>
```

Ad esempio con:

```
# chown user miofile
```

viene assegnato come proprietario del file miofile l'utente user, mentre non viene modificato il gruppo.

Qualora si provi ad eseguire lo stesso comando con un utente regolare, si riceve un errore:

```
$ chown user miofile chown: changing ownership of 'miofile': Operation not
permitted
```

Con lo stesso comando è possibile cambiare contemporaneamente sia il proprietario del file che il gruppo, separando l'utente dal gruppo tramite i due punti (:).

```
# chown user:utenti miofile
```

In questo esempio viene assegnato, al file miofile, come proprietario l'utente user e come gruppo utenti. Per assegnare ad un file il gruppo primario del nuovo proprietario è anche possibile utilizzare la seguente sintassi:

```
# chown user: miofile
```

In questo altro esempio al file miofile viene assegnato come proprietario l'utente user e come gruppo quello primario di user.

Qualora non si specifichi il proprietario è possibile modificare esclusivamente il gruppo del file, ad esempio:

```
# chown :utenti miofile
```

Per modificare solamente il gruppo è anche possibile utilizzare il comando *chgrp* (che sta per *change group ownership*), simile a *chown* che prevede però come argomento unicamente il nome del gruppo, senza i due punti (:). La sintassi del comando è:

```
chown [<options>] <group> <filename>
```

Ad esempio:

```
$ chgrp utenti miofile
```

L'utente standard può cambiare il gruppo del file ma solamente se si è proprietari del file e si appartiene al nuovo gruppo. In caso contrario si riceve un errore:

```
$ chgrp adm miofile
chgrp: changing group of 'miofile': Operation not permitted
```

I comandi *chown* e *chgrp* supportano numerose opzioni per modificarne il comportamento. La più utile è *–R* (o *--recursive*) che cambia ricorsivamente la proprietà di una directory e di tutti i file e sottodirectory in essa contenuti. Ad esempio il comando:

```
# chown -R user /home/utente
```

assegna la proprietà all'utente user della directory /home/utente e di tutti i file e sottodirectory in essa contenuti.

La modifica del proprietario o del gruppo non influisce sui permessi del file o della directory.
È inoltre possibile modificare il proprietario ed il gruppo utilizzando l'interfaccia grafica dei file browser, come Nautilus e Konqueror.

18.2 Permessi di file/directory

La proprietà del file è inutile senza un meccanismo per specificare cosa gli utenti possano fare sui file propri o degli altri utenti. Per fare ciò si utilizzano i permessi.

I permessi previsti, che possono essere combinati tra di loro, sono: lettura (*read*), scrittura (*write*) ed esecuzione (*execute*). Linux gestisce permessi separati per il

proprietario del file (*owner*), per i membri del gruppo attribuito al file (*group*) e per tutti gli altri utenti (*others*). Il proprietario del file può assegnare i permessi separatamente ed individualmente per questi tre insiemi di utenti (*owner*, *group*, *others*). Lo stesso vale, ovviamente, anche per le directory, sebbene i permessi abbiano effetti differenti tra file e directory.

Relativamente ai file:

- Il permesso di lettura (*read*) consente di visualizzarne il contenuto del file.
- Il permesso di scrittura (*write*) consente di modificarne il contenuto, ma non cancellarlo! Quindi anche se è possibile intervenire sul contenuto, il solo permesso *write* (w) non consente di cancellare il file.
- Il permesso di esecuzione (*execute*) consente di eseguire i file come programmi. Qualora si tratti di script di shell, o altre tipologie di file da interpretare, è necessario anche il permesso di lettura. Abilitare il permesso di esecuzione su di un file che non è un programma non lo fa diventare un programma; il permesso permette solamente di eseguire un file che è già un programma.

Per quanto riguarda le directory:

- Il permesso di lettura (*read*) permette di visualizzarne, ad esempio con il comando *ls*, i file e le sottodirectory contenuti nella directory.
- Il permesso di scrittura (*write*) permette di creare, cancellare e rinominare i file e le sottodirectory contenuti nella directory. Come detto, non sono i permessi del file a determinare da chi esso può essere cancellato, ma sono i permessi della directory che lo contiene che lo stabiliscono. Il comando *rm* avvisa qualora si stia tentando di cancellare un file per il quale non si hanno i permessi di scrittura; nel caso in cui si risponda affermativamente alla domanda "*rm: remove write-protected regular file 'miofile'?*" e si abbiano i permessi sulla directory il file verrà cancellato, anche se non se ne è proprietari. Tale comportamento predefinito del permesso di scrittura può essere modificato tramite lo *sticky bit*, come descritto nel capitolo 5.4.
- Il permesso di esecuzione (*execute*) garantisce la navigazione della directory, permettendo di entrarne all'interno, attraversarla ed utilizzarne il relativo percorso. Non ha molto senso, tranne rari casi specifici, assegnare separatamente alle directory i permessi di lettura ed esecuzione. Il solo permesso di esecuzione permette di accedere alla directory ma non di elencarne il contenuto; si potrebbe però continuare ad accedere ai file tramite il loro nome qualora lo si conosca.

È importante comprendere il meccanismo utilizzato da Linux per stabilire i permessi di un utente rispetto ad un file. Per prima cosa viene verificato se l'utente è il proprietario: in tal caso all'utente si applicheranno i permessi di owner indipendentemente dagli altri permessi; in caso contrario viene verificato se l'utente fa parte del gruppo assegnato al file: in tal caso all'utente vengono applicati i permessi di group indipendentemente da

quelli di others. Nel caso in cui l'utente non sia né il proprietario né membro del gruppo del file, gli vengono assegnati i permessi di others. Questo vuol dire che se il gruppo del file non ha i permessi di lettura ma li ha others, allora un membro del gruppo del file non potrà visualizzare il contenuto del file.

Le informazioni sul proprietario e sui permessi del file sono specifiche dei filesystem Linux e non vengono preservate quando vengono trasferite su supporti removibili che utilizzano filesystem diversi, come *File Allocation Table* (*FAT*) o *New Technology File System* (*NTFS*).

Per visualizzare il proprietario ed i permessi dei file si utilizza il comando *ls* con l'opzione *–l*, che sta per *long list* cioè una lista dettagliata dei file presenti in una directory (si rimanda al capitolo 2.3 per una spiegazione approfondita del comando *ls* e delle sue opzioni più comuni):

> $ ls -l -rw-r--r-- 1 user utenti 23 Aug 8 14:25 filemio drwxr-x--- 2 user utenti 4096 Aug 8 14:26 sottodir

L'output del comando è formato, per ogni file, da diverse sezioni:

- Permessi: la prima colonna rappresenta, in modalità simbolica, i permessi del file;
- Numero di *hard link*: la seconda colonna riporta il numero di *hard link* (*link counter*) al file (argomento che verrà trattato e approfondito nel capitolo 5.4);
- Proprietario: la terza colonna identifica il nome utente del proprietario del file;
- Gruppo: la quarta colonna riporta il nome del gruppo assegnato al file;
- Dimensione del file: la quinta colonna mostra la grandezza del file espressa in byte;
- Data e ora di ultima modifica: la sesta colonna indica il giorno e l'orario di ultima modifica del file;
- Nome del file: infine l'ultima colonna mostra il nome del file.

Per quanto riguarda i dieci caratteri che costituiscono i permessi, essi sono suddivisi nel seguente modo:

- il primo carattere rappresenta la tipologia di file: - per i file regolari, d per le directory, l per i link simbolici, b e c per i device file, etc.); poiché in realtà non è un permesso viene omesso nell'assegnazione dei permessi ai file;
- tre serie (una per il proprietario, una per il gruppo ed una per gli altri) di tre caratteri ciascuna che identificano la presenza dei tre permessi visti in precedenza, rispettivamente:
  - "r" per la lettura,
  - "w" per la scrittura,
  - "x" per l'esecuzione; o la loro relativa assenza:
  - "-" un segno meno al posto del permesso corrispondente.

Nella Tabella 1 è riportata la suddivisione dei dieci caratteri che costituiscono i permessi di un file.

Tabella 17 - Rappresentazione dei permessi di un file

Tipologia di file	Permessi per il proprietario (owner: u)			Permessi per il gruppo (group: g)			Permessi per gli altri (others: o)		
-	r	w	x	r	-	x	r	-	x
	lettura (read)	scrittura (write)	esecuzione (execute)	lettura (read)	scrittura (write)	esecuzione (execute)	lettura (read)	scrittura (write)	esecuzione (execute)

Perciò "*rwxr-xr-x*" indica che il proprietario del file ha i permessi di lettura e scrittura mentre i membri del gruppo e tutti gli altri utenti solamente la lettura e l'esecuzione.

Questa modalità di rappresentare i permessi dei file, tramite le lettere, è chiamata rappresentazione simbolica.

Esiste un'altra modalità più compatta per specificare i permessi dei file; tale modalità utilizza, per esprimere i permessi per le tre classi di utenza (proprietario, gruppo, altri), un numero di tre cifre. Ogni cifra è in base ottale, può assumere cioè un valore da 0 a 7. I permessi vengono quindi specificati tramite un numero di tre cifre in base otto. Tale rappresentazione si chiama infatti ottale.

Ciascuna delle tre cifre viene calcolata tramite la somma dei numeri dei singoli permessi abilitati, che sono:

- 4 per la lettura (r);
- 2 per la scrittura (w);
- 1 per l'esecuzione (x).

La somma dei valori per i permessi abilitati fornisce il numero che rappresenta il permesso complessivo relativo ad una classe di utenza.

Gli otto valori che è possibile ottenere combinando i permessi, con la rispettiva rappresentazione simbolica, sono:

- 0 (---) nessun permesso;
- 1 (--x) esecuzione;
- 2 (-w-) scrittura;
- 3 (-wx) scrittura ed esecuzione (2+1);
- 4 (r--) lettura;
- 5 (r-x) lettura ed esecuzione (4+1);
- 6 (rw-) lettura e scrittura (4+2);
- 7 (rwx) lettura, scrittura ed esecuzione (4+2+1).

La concatenazione di questi valori forma un numero di tre cifre che rappresenta i permessi complessivi del file: per il proprietario, per il gruppo e per gli altri.

Ci sono cinquecentododici possibili combinazioni di permessi; di eseguito alcuni di quelli più utilizzati con la loro rappresentazione ottale, simbolica e spiegazione:

- 777 (rwxrwxrwx) lettura, scrittura ed esecuzione per tutti gli utenti;
- 755 (rwxr-xr-x) lettura, scrittura ed esecuzione per il proprietario, solo lettura ed esecuzione per il gruppo e per gli altri;
- 700 (rwx------) lettura, scrittura ed esecuzione per il proprietario, nessun permesso per il gruppo e per gli altri;
- 664 (rw-rw-r--) lettura e scrittura per il proprietario ed il gruppo, solo lettura per gli altri;
- 644 (rw-r--r--) lettura e scrittura per il proprietario, solo lettura per il gruppo e per gli altri;
- 600 (rw-------) lettura e scrittura per il proprietario, nessun permesso per il gruppo e per gli altri;
- 550 (r-xr-x---) lettura ed esecuzione per il proprietario e per il gruppo, nessun permesso per gli altri;
- 400 (r--------) solo lettura per il proprietario, nessun permesso per gli altri.

Le regole dei permessi qui esposte non hanno effetto per l'utente root che può sempre, anche qualora un file abbia permessi 000, leggere e scrivere tutti i file. Anche l'utente root ha però necessità del permesso di esecuzione attivo sui file per poterli eseguire.

Per poter cambiare i permessi di un file o di una directory si può utilizzare il comando *chmod* (che sta per *change file mode bits*). Il comando può essere eseguito esclusivamente dal proprietario del file, oltre che dall'utente root. È possibile specificare i permessi tramite le due rappresentazioni precedentemente esposte: forma simbolica o numero ottale.

Per modificare i permessi utilizzando la rappresentazione ottale basta specificare il valore dei permessi ed il nome del file. Ad esempio:

```
$ chmod 644 filemio
```

assegna i permessi di lettura e scrittura per il proprietario e di sola lettura per il gruppo e per gli altri (rw-r--r--) al file filemio.

Questa modalità di assegnazione dei permessi è di tipo assoluta poiché sovrascrive totalmente i permessi esistenti del file e richiede di specificare i permessi per tutte le classi di utenza.

Tramite la rappresentazione simbolica i permessi vengono specificati attraverso una serie di uno o più caratteri separati dalla virgola. Ogni valore è composto da una o più lettere che rappresentano la classe, o le classi, di utenza di cui si vuole modificare il permesso.

Le lettere che identificano le classi sono:

- u per il proprietario;
- g per il gruppo;

- per gli altri;
- a (o ugo o nessuna lettera) per tutte le categorie.

Dopo la classe/le classi è necessario indicare, tramite un apposito carattere, l'operazione da eseguire su tale classe/classi:

- + per aggiungere i permessi specificati senza alterare gli altri i permessi già presenti;
- per rimuove i permessi specificati senza alterare gli altri permessi già presenti;
- = per impostare i permessi specificati, rimuovendo gli altri permessi già presenti.

Infine si devono specificare i permessi attraverso la rappresentazione simbolica (r, w, x) già vista. In Figura 1 è riportata la rappresentazione simbolica per l'assegnazione dei permessi.

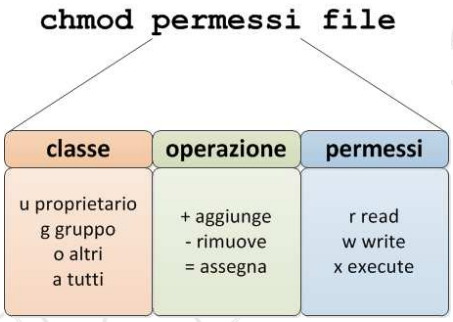


Figura 10- Assegnazione dei permessi in modalità simbolica

Qualora sia necessario assegnare permessi differenti per le diverse classi, è necessario dividere ogni serie così composta, tramite la virgola ",".

La rappresentazione simbolica è più flessibile in quanto è possibile utilizzare sia la modalità di assegnazione dei permessi assoluta che quella relativa, in cui cioè si può modificare un singolo permesso ed una singola classe di utenza senza alterare gli altri permessi presenti.

Ad esempio per assegnare in modalità assoluta i permessi di lettura e scrittura per il proprietario e di sola lettura per gli altri, come nel caso precedente, si può utilizzare il comando:

```
$ chmod u=rw,go=r filemio
```

La modalità relativa si realizza utilizzando i simboli "+" e "-" che aggiungono o rimuovono specifici permessi o con il simbolo "=" ma senza indicare tutte le classi di utenza in modo tale che quelle non specificate non vengano alterate. Per poter stabilire i permessi finali, assegnati tramite modalità relativa, è necessario quindi conoscere i permessi iniziali del file.

Ad esempio se il file filemio ha i permessi "*rw-r--r--*" e si esegue il comando:

```
$ chmod a+x filemio
```

che abilita l'esecuzione per tutti, i nuovi permessi saranno: *rwxr-xr-x*.

Se ora si esegue il comando:

```
$ chmod og=rw filemio
```

i nuovi permessi saranno: *rwxrw-rw-*.

A questo punto eseguendo il comando:

```
$ chmod u-wx,og-w filemio
```

i nuovi permessi saranno: *r--r--r--*.

La sintassi generale del comando *chmod* è:

```
chmod [⟨options⟩] ⟨permissions⟩ ⟨filename⟩
```

È consentito specificare, alla fine della riga, i nomi di più file e directory.

Una opzione utile è *--reference=<sourcefile>* che permette di assegnare al file filename gli stessi permessi del file sourcefile. Ovviamente in questo caso non è necessario specificare da linea di comando la parte *<permissions>* relativa ai permessi.

Come per il comando *chown* e *chgrp* è possibile utilizzare l'opzione *–R* (o *--recursive*) per modificare ricorsivamente i permessi di tutti i file contenuti in una directory e nelle sue sottodirectory.

I permessi predefiniti quando viene creato un file sono 666 (rw-rw-rw-) mentre per una directory sono 777 (rwxrwxrwx). A questi permessi vanno però rimossi i permessi definiti nella *umask* (che sta per *user mask*). Abitualmente la *umask* ha valore 022 (----w--w-), ma può essere modificata. I permessi risultanti saranno allora 644 (rw-r--r--) per i file e 755 (rwxr-xr-x) per le directory. Benché in questo caso la sottrazione della *umask* effettuata sia in forma ottale che in forma simbolica coincidono, per essere sicuri di non commettere errori la sottrazione della *umask* dai permessi predefiniti va effettuata bit a bit tramite la modalità simbolica. Se ad esempio il valore della *umask* fosse 037 i permessi risultanti per i file sarebbero 640 (rw-r-----) e non, come risulterebbe tramite la sottrazione in modalità ottale, 666-037 che produrrebbe per il gruppo il permesso 3 (-wx) e per gli altri -1, entrambi valori errati o senza senso. Alcune distribuzioni utilizzano come predefinito per la *umask*, anziché 022, il valore 002.

L'argomento della *umask* non viene qui ulteriormente approfondito; si riporta solamente il comando per modificare il valore della *umask*:

```
umask <valore>
```

precisando che tale modifica ha effetto esclusivamente sui file creati da quel momento in poi e non su quelli già creati.

Comandi trattati

- *chgrp* - Modifica il gruppo proprietario di file e directory
- *chmod* - Imposta i permessi per file e directory
- *chown* - Modifica il proprietario e/o il gruppo di file e directory

Esercitazione proposta

1. Con il proprio utente standard (non root), creare un file e scrivere qualcosa al suo interno.

2. Visualizzare i permessi del file. Quali sono? Perché? Qual è il suo gruppo?

3. Provare ad assegnare al file un gruppo di cui non si è membri. Cosa accade?

4. Aggiungere al file, senza modificare gli altri permessi attuali, i permessi di esecuzione per tutte le classi di utenza.

5. Assegnare al file, tramite la rappresentazione ottale, i seguenti permessi: lettura e scrittura per il proprietario, nessun permesso per il gruppo, lettura per gli altri.

6. Provare con un altro utente appartenente allo stesso gruppo del file a visualizzare il contenuto del file. Cosa accade?

7. Provare con un altro utente NON appartenente allo stesso gruppo del file a visualizzare il contenuto del file. Cosa accade?

8. Con il proprio utente standard, assegnare al file, in modalità simbolica, i seguenti permessi: lettura e scrittura per il proprietario, nessun permesso per il gruppo e per gli altri.

9. Provare a cambiare il proprietario del file. Cosa accade?

10. Con l'utente root provare a visualizzare il contento del file. Cosa accade?

11. Sempre con l'utente root cambiare il proprietario ed il gruppo del file. Cosa accade?

Soluzione dell'esercitazione

1. *$ echo "ciao" > esercizio;*

2. *$ ls –l;*
I permessi possono variare in base alla distribuzione Linux utilizzata. Il proprietario ed il gruppo del file sono gli stessi dell'utente corrente.

3. *$ chgrp sys esercizio;*
Errore: "*Operation not permitted*".

4. *$ chmod a+x esercizio;* oppure *$ chmod ugo+x esercizio;* oppure *$ chmod u+x,g+x,o+x esercizio;*

5. *$ chmod 604 esercizio;*

6. *$ more esercizio;*
Errore: "*Permission denied*".

7. *$ more esercizio;*
Ora è possibile visualizzare il file.

8. *$ chmod u=rw,go= esercizio;*

ISBN: 9788866850090

9. *$ chown root esercizio;*
Errore: "*chown: changing ownership of 'esercizio': Operation not permitted*"

10. *# more esercizio;*
L'utente root può visualizzare e scrivere qualunque file anche non avendone i relative permessi.

11. *# chown user2 esercizio;*

Domande

1. Quale comando permette di cambiare il proprietario del file relazione da alice a bob?

A. *chown alice:bob relazione*
B. *chmod relazione bob*
C. *chown relazione bob*
D. *chown bob relazione*
E. *chmod bob relazione*

2. Quali comandi permettono di cambiare il gruppo del file? (scegliere più risposte)

A. *groupadd*
B. *groupmod*
C. *chmod*
D. *chgrp*
E. *chown*

3. Solo l'utente root può cambiare il proprietario del file?

A. Vero B. Falso

4. Solo l'utente root può cambiare il gruppo assegnato al file?

A. Vero B. Falso

5. Il proprietario del file può cambiare il gruppo assegnato al file con qualsiasi altro gruppo?

A. Vero B. Falso

6. Quale opzione permette di cambiare il proprietario di una directory e di tutti i file e sottodirectory?

A. *–R* B. *–t* C. *–d* D. *–r*

7. L'output del comando *ls –l* è il seguente:
drwxr-x--- 2 user staff 4096 Aug 8 14:26 archivio
Quali delle seguenti affermazioni sono vere? (scegliere più risposte)

A. archivio è un link simbolico
B. archivio è un programma eseguibile.
C. archivio è una directory.

D. archivio può essere letto da tutti gli utenti del sistema.
E. staff è il nome del proprietario di archivio.
F. staff è il nome del gruppo assegnato a archivio.
G. archivio può essere letto dai membri del gruppo del file.

8. Il permesso 755 su di un file consente a qualsiasi utente di leggere il contenuto del file?

A. Vero B. Falso

9. Il permesso 755 su di un file consente a qualsiasi utente di modificare il contenuto del file?

A. Vero B. Falso

10. Il permesso 666 su di un file consente a qualsiasi utente di cancellare il file?

A. Vero B. Falso

11. Qual è la rappresentazione ottale dei seguenti permessi: *rwxr-xr—*

A. 765 B. 754 C. 457 D. 321 E. 751

12. Qual è la rappresentazione simbolica dei seguenti permessi: 640

A. *rwxr-x---* B. *-wx-wx---* C. *--x-wxrwx* D. *rw-r-----*

13. Qual è il comando per assegnare al proprietario i permessi 6, aggiungere (senza modificare il resto) il permesso di scrittura al gruppo e togliere (senza modificare il resto) il permesso di lettura per gli altri al file archive?

A. *chmod archive u=rw,g+w,o-r*
B. *chmod u=rw,g+w,o-r archive*
C. *chmod o=rw,g+w,a-r archive*
D. *chmod u=rw-,g+-w-,o-r-- archive*

14. Solo l'utente root può utilizzare il comando *chmod*?

A. Vero B. Falso

Risposte

1. D
2. D, E
3. A
4. B
5. B
6. A
7. C, F, G
8. A
9. B
10. B

ISBN: 9788866850090

11. B
12. D
13. B
14. B

ithum®

19 Directory e file speciali

19.1 Gerarchia del filesystem e file di sistema

Come già ricordato, Linux è un sistema operativo multi-utente che può ospitare centinaia o migliaia di utenti. Per evitare che un utente possa modificare dei file di configurazione ed alterare il funzionamento del sistema Linux prevede una distinzione tra i file degli utenti e quelli di sistema.

I file di sistema sono quei file che controllano e modificano il funzionamento del computer, come: script, demoni, programmi, file di configurazione, file di log, etc.

Molti dei file di sistema sono di proprietà dell'utente root o di altri account amministrativi di sistema per scopi specifici. Gli utenti potrebbero essere in grado di leggere alcuni di questi file anche se molti sono protetti anche dalla lettura. È evidente quindi che, anche in un sistema mono-utente, per effettuare la maggior parte dei compiti amministrativi e di manutenzione sono necessari i privilegi di root. Questo aumenta la protezione da errori accidentali eseguiti dall'utente standard.

Subito dopo aver installato un sistema Linux la quasi totalità dei file e delle directory presenti sono di sistema, mentre solamente pochi (come */home* e */tmp*) sono per i file degli utenti.

Linux utilizza una gerarchia di directory (o *filesystem*) unificata, in cui cioè ogni disco, partizione, condivisione e dispositivo è accessibile come se fosse una normale directory del *filesystem*. Per chi è abituato a lavorare con Windows, in cui ogni disco o partizione è identificato da una propria lettera (come *A:*, *C:*, *D:*, etc.), questa è una differenza sostanziale.

Un'altra differenza importante tra i due sistemi operativi riguarda la locazione dei file e delle directory. Abitualmente in Windows un programma e tutti i file relativi, come i dati e le configurazioni, risiedono all'interno della stessa directory sotto *C:\Programmi*. La gerarchia delle directory di Linux, invece, è strutturata in modo tale da posizionare i file e le directory in base alla loro funzione o al loro scopo: programmi, librerie, configurazioni, file utente, etc. Tali locazioni sono indipendenti da dove esse vengono fisicamente memorizzate: normale sottodirectory, diversa partizione o altro disco. Conoscere l'organizzazione del *filesystem* Linux è dunque importante.

Ogni distribuzione Linux ha una propria modalità di strutturare il *filesystem*. Esistono però alcune convenzioni per l'organizzazione delle directory; ciò consente ad esempio agli amministratori di sistema e ai programmi di trovare nella stessa posizione i file di configurazione di sistema in modo tale che i comandi possano funzionare indipendentemente dalla distribuzione in cui ci si trova a lavorare. Lo standard principale per tale esigenza è il *Filesystem Hierarchy Standard* (*FHS*) a cui la maggior parte delle

distribuzioni Linux aderisce, sebbene con alcune differenze tra le diverse distribuzioni. *Filesystem Hierarchy Standard* definisce il nome, la locazione e lo scopo di molte directory e sottodirectory di un sistema Linux.

Nell'organizzazione delle directory *FHS* effettua una distinzione tra le directory che contengono i file che possono essere condivisi con altri sistemi attraverso la rete e quelle che invece devono necessariamente risiedere localmente. Un'altra distinzione viene fatta tra le directory contenenti file statici (che non variano nell'utilizzo ordinario del sistema) e dinamici (i file variabili).

Di seguito si fornirà una descrizione delle più importanti directory che contengono, approssimativamente, gli stessi file in tutte le distribuzioni:

- */* La directory root (da non confondersi con la directory */root*) contiene tutte le directory e sottodirectory.
- */etc* Directory particolarmente importante nell'amministrazione del sistema poiché contiene la maggior parte dei file di configurazione specifici della macchina (come i già visti *passwd*, *shadow* e *group*) con sottodirectory specifiche per servizi e sottosistemi complessi.
- */boot* Contiene il kernel del sistema operativo ed i file relativi all'avvio del sistema come quelli necessari per il boot loader (*LILO* o *GRUB*).
- */bin* Contiene i principali programmi essenziali (*binaries*) che un utente normale abitualmente può eseguire (es. *ls*, *cp*, *mkdir*, …).
- */sbin* Contiene i programmi per l'amministrazione del sistema (*system binaries*) che abitualmente esegue l'utente root (es. *init*, *mount*, …).
- */lib* Contiene le librerie (collezioni di file con codice condiviso per il funzionamento degli altri programmi) che sono essenziali per i binari presenti in */bin* e */sbin*.
- */usr* Gerarchia secondaria in cui risiedono i programmi e dati utilizzati nell'operatività normale del sistema ma che non sono essenziali per l'avvio del sistema. Tale directory è nata per essere condivisa tra i sistemi e contiene diverse sottodirectory che rispecchiano parti della gerarchia di root, come ad esempio */usr/bin*, */usr/sbin* e */usr/lib*.
- */var* Contiene i file variabili, quei file cioè che vengono abitualmente modificati dinamicamente durante l'operatività ordinaria del sistema, come ad esempio i log e lo *spool* di stampa. La sottodirectory */var/tmp* contiene file temporanei che non devono essere cancellati al riavvio del sistema.
- */opt* Contiene i pacchetti software opzionali e aggiuntivi sviluppati da terze parti.
- */srv* Contiene i dati per i servizi offerti dal sistema (come i server ftp, web, etc.).
- */tmp* Directory, accessibile a tutti gli utenti, contenente i file temporanei che vengono abitualmente cancellati dalla maggior parte delle distribuzioni Linux al riavvio del sistema.
- */mnt* Contiene i punti di accesso (*mount points*) per i *filesystem* montati temporaneamente.

- */media* Contiene i punti di accesso (*mount points*) per i media rimovibili come i CD-ROM ed i floppy.
- */dev* Contiene una moltitudine di file per l'accesso di basso livello ai dispositivi (*devices files*). Tali file non sono file standard con un contenuto classico come gli altri file ma contengono i *device number* che identificano il driver che il *kernel* deve utilizzare per accedere a tale periferica. Sono presenti *character device file*, *block device file* e *pseudo device* (come */dev/null* e */dev/zero* che non sono veri e propri dispositivi).
- */home* Contiene le *home* directory degli utenti in cui essi possono creare i propri file. Abitualmente tale directory è collocata su una partizione o su un disco differente per isolare tali dati rispetto a quelli di sistema.
- */root* Home directory dell'utente root.

La maggior parte di queste directory sono di sistema; le principali eccezioni sono */home*, */tmp*, */mnt* e */media*.

Tutte le directory possono risiedere su di un'unica partizione oppure essere collocate su partizioni o dischi differenti. Per un utente normale questo è trasparente. Vedrà una directory e non saprà se è una normale directory contenuta nella stessa partizione del *filesystem* root (/) oppure se è un punto di accesso (*mount point*) per un'altra partizione.

Se un disco contiene diverse partizioni, una di queste diventerà il *filesystem* root (/) mentre le altre saranno montate, attraverso un *mount point*, in una specifica directory all'interno dell'unica struttura gerarchica. La stessa cosa avviene quando si montano CD-ROM, DVD, dispositivi di memorizzazione USB o altri dispositivi rimovibili.

19.2 File speciali

Linux permette di creare dei collegamenti (*link*) ai file ed assegnare quindi differenti nomi allo stesso file. Questo, ad esempio, può essere utile per creare una scorciatoia ad una directory o ad un file riferendosi ad esso in maniera più comoda.

Esistono due differenti tipologie di link in Linux: *soft link* (o link simbolici) e *hard link*.

19.2.1 Link simbolici

I link simbolici (o *soft link*) sono dei file particolari che memorizzano al loro interno il percorso, assoluto o relativo, del file o della directory a cui sono collegati. I link simbolici ridirezionano le operazioni di lettura e scrittura sul file a cui sono collegati, il quale non è a conoscenza del link simbolico. Infatti la creazione e la rimozione di un link simbolico non hanno alcun effetto sul file destinazione. Se però il file destinazione viene rimosso, il link simbolico continua ad esistere ma diventa "orfano" e non punta più a nulla.

Differentemente dagli *hard link*, che verranno descritti successivamente, i link simbolici permettono di effettuare un collegamento sia a file che a directory, anche residenti su partizioni differenti. Vengono abitualmente utilizzati per mantenere la retrocompatibilità

o la compatibilità tra distribuzioni differenti in cui lo stesso file o la stessa directory può trovarsi in percorsi differenti.

Per creare i link si utilizza il comando *ln* (che sta per *link*). La sintassi del comando è:

ln [<options>] <target> <linkname>

Per i link simbolici si utilizza l'opzione *–s* (che sta per *soft link*). Ad esempio, per creare nella directory corrente un link simbolico di nome hUser alla home directory dell'utente user, si utilizza il comando:

$ ln -s /home/user hUser

È possibile specificare, come destinazione del link simbolico, percorsi sia relativi che assoluti, come visto nel capitolo 2.3.

Per visualizzare il link simbolico e capire il file a cui esso è collegato si utilizza il solito comando *ls –l*:

$ ls -l lrwxrwxrwx 1 user utenti 12 Aug 8 17:05 hUser -> /home/user

Si può vedere dall'output del comando come la tipologia del file sia "*l*" che sta per *soft link*, ed il percorso del file a cui punta il link simbolico. Inoltre la grandezza del file (12 byte) corrisponde al numero dei caratteri del percorso a cui fa riferimento il link.

Per rimuovere un link simbolico si utilizza il consueto comando *rm*. Tale operazione ha effetto esclusivamente sul link e non sulla destinazione del collegamento.

$ rm hUser

I permessi dei link simbolici sono sempre 777 (*rwxrwxrwx*); questo permesso ha effetto sul link e non sulla destinazione del link. Perciò tutti gli utenti possono visualizzare il percorso del link, ma sono i permessi configurati sulla destinazione a stabilire se un utente può leggere, scrivere o eseguire il file. È necessario porre attenzione poiché la modifica dei permessi di un link simbolico ha effetto non sul link stesso, bensì sul file collegato.

19.2.2 Hard link

L'altra tipologia di link si chiama *hard link* ed anziché utilizzare il percorso del file a cui collegarsi, utilizza il numero di *inode* (o *inode number*). Ogni file ha infatti un nome che è associato con un *inode*. L'*inode* è una struttura dati del *filesystem* che memorizza le seguenti informazioni del file: dimensione, proprietario, gruppo di appartenenza, permessi, informazioni temporali, contatore di *hard link* che referenziano l'*inode*, puntatore alla locazione fisica dei blocchi del disco che memorizzano il contenuto del file. L'unica informazione che non è presente nell'*inode* è il nome del file. All'interno di ogni *filesystem* gli *inode* sono numerati in maniera univoca. Quando si lavora con un file, Linux recupera il corrispondente *inode* dal *filesystem* per poter interagire con il file. I file non sono altro che *hard link* (anche detti collegamenti fisici) agli *inode* tramite appunto l'*inode number*. La Figura 1 mostra la relazione esistente tra i nomi dei file ed i relativi *inode*.

Per creare un hard link si utilizza ancora il comando *ln* ma senza l'opzione *-s*. È necessario solamente specificare il file esistente a si vuole associare il collegamento ed il nome del link:

```
$ ln fileEsistente fileNuovo
$ ls -l -rwxr--r-- 2 user utenti 5 Aug  8 17:11 fileEsistente -rwxr--r-- 2 user utenti 5
Aug  8 17:11 fileNuovo
```

Eseguendo il comando *ls –l* è possibile visualizzare i due file che non sono più file simbolici ma file regolari. Dall'output del comando si evince, seconda colonna, che il contatore dei link (*link counter*) è 2, visto che abbiamo appena creato un *hard link* al file. Per verificare che i due nomi facciano riferimento allo stesso file è necessario utilizzare il comando *ls –i* che mostra l'*inode* del file, cioè l'identificatore univoco del file all'interno della partizione su cui si trova il file.

```
$ ls -i 18945 fileEsistente 18945 fileNuovo
```

L'*inode* identifica totalmente il file: proprietario, gruppo, permessi e contenuto. In questo esempio è possibile verificare che fileEsistente e fileNuovo sono effettivamente il medesimo file poiché posseggono numero di *inode* identico (18945): hanno infatti grandezza uguale, lo stesso proprietario, lo stesso gruppo, la stessa data di modifica e gli stessi permessi; abbiamo semplicemente dato al medesimo file un nuovo nome. Modificando il contenuto di un file viene quindi modificato anche l'altro (e lo stesso vale per il proprietario, per il gruppo e per i permessi del file). Il file è ora accessibile attraverso due nomi differenti. Un file con due o più *hard link* ha quindi più nomi ma nessuno di essi è quello "vero"; per il sistema sono tutti equivalenti ed indistinguibili.

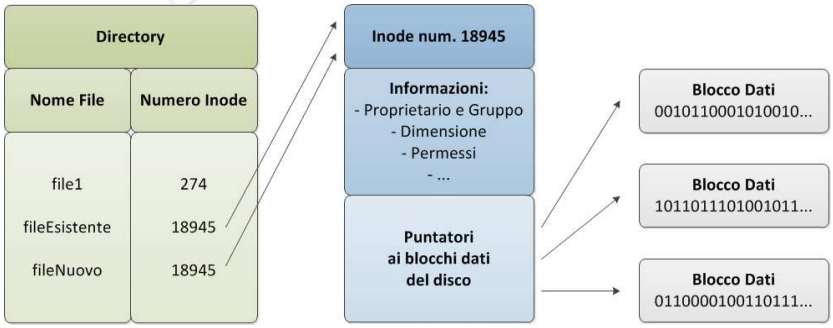


Figura 11 - Relazione tra nomi dei file e inode

Benché in questo esempio i due file, fileEsistente e fileNuovo, siano contenuti nella stessa directory è possibile creare o spostare gli *hard link* anche in directory differenti, a patto che si resti all'interno dello stesso *filesystem*.

Cancellando un file questo non viene effettivamente rimosso dall'hard disk ma ne viene decrementato il contatore di *hard link* dall'*inode*. Solo quando il contatore raggiunge il

valore 0, e quindi non ci sono altri nomi per riferirsi ad esso, il file viene eliminato dall'hard disk.

Ad esempio:

```
$ rm fileEsistente
$ ls -li 18945 -rwxr--r-- 1 user utenti 5 Aug  8 17:11 fileNuovo
$ rm fileNuovo
$ ls -li
```

Come già accennato, la numerazione degli *inode* è effettuata a livello di *filesystem* fisico, cioè di partizione; per tale motivo è possibile creare gli *hard link* esclusivamente all'interno della stessa partizione, a differenza dei link simbolici che possono far riferimento anche a file su partizioni differenti poiché memorizzano il percorso del file. Inoltre, mentre i link simbolici possono fare riferimento sia ad un file che ad una directory, non è possibile creare *hard link* a directory (le uniche eccezioni sono "." e ".." che il sistema crea per ogni directory).

Si dice che nei sistemi operativi Unix-like "tutto è un file". Anche le directory sono dei file il cui contenuto è una corrispondenza tra i nomi dei file ed i corrispondenti numeri di *inode*. Questa corrispondenza, tra nome e *inode*, è chiamata appunto *hard link*. La Figura 1 mostra tale corrispondenza.

Quando viene creato un file, anche tramite il comando *ln*, viene aggiunto nella directory un *hard link* tra il nome del file e l'*inode*. Quando invece il file viene rimosso, nella directory viene eliminato l'*hard link*.

19.3 Permessi speciali

Nel capitolo 5.3 sono stati descritti i permessi standard presenti in Linux. Oltre a questi ne esistono altri speciali che sono ugualmente importanti: *Sticky bit*, *setuid* e *setgid*.

Questi permessi avanzati permettono di modificare alcuni comportamenti predefiniti dei file e delle directory su cui vengono abilitati. È necessario prestare attenzione quando si incontrano questi file poiché si discostano da quanto ci si potrebbe aspettare sulla base di quanto visto nei capitoli precedenti.

Prima di trattarli in dettaglio è però utile comprendere come sia possibile visualizzare facilmente i permessi assegnati ad una directory.

19.3.1 Visualizzare i permessi di una directory

Qualora sia necessario visualizzare le informazioni di una directory e si esegue *ls –l*, passandogli come argomento il nome della directory, l'output del comando mostrerà i dettagli dei file contenuti nella directory e non le informazioni relative alla directory. Per sopperire a questa esigenza si utilizza il comando *ls* con le opzioni *–ld* (dove *d* sta per *directory*). Il funzionamento di tale comando è mostrato nel seguente esempio:

```
$ ls -l dir*
dir1: total 8 -rw-r--r-- 1 user utenti 6483 Aug  8 17:21 file1
dir2: total 4 -rw-r--r-- 1 user utenti 5 Aug  8 17:21 file2
-rw-r--r-- 1 user utenti 1 Aug  8 17:21 file3

$ ls -ld dir*
drwxr-xr-x 2 user utenti 4096 Aug  8 17:21 dir1
drwxr-xr-x 2 user utenti 4096 Aug  8 17:21 dir2
```

19.3.2 Sticky bit

È già stato evidenziato come un utente, pur non avendo i permessi di scrittura su di un file, lo possa cancellare qualora abbia i permessi di scrittura per la directory contenente il file. Questa non è una contraddizione; come visto, le directory sono speciali tipologie di file che contengono la corrispondenza tra il nome del file e l'*inode*. Per tale motivo la modifica di un file richiede i permessi di scrittura sul file ma la creazione o l'eliminazione di un file richiede i permessi di scrittura nella directory in cui esso risiede.

Questo comportamento standard di Linux potrebbe non essere sempre desiderabile; per modificarlo si utilizza un permesso speciale, chiamato *sticky bit*. Abilitando lo *sticky bit* su una directory si permette solamente al proprietario del file e al proprietario della directory contenente il file di cancellarlo; i permessi di scrittura sulla directory non sono più sufficienti. Questo, quindi, impedisce ad un utente di rimuovere i file altrui dentro una directory modificabile da più utenti. L'utente root è ovviamente ancora in grado di cancellarli.

È possibile abilitare lo *sticky bit* attraverso il comando *chmod* tramite le due modalità di rappresentazione: simbolica e ottale.

La lettera che identifica lo *sticky bit* nella rappresentazione simbolica è la "*t*" che si può utilizzare come gli altri permessi, specificando come classe di utenza quella per gli altri (*o*) e la specifica operazione da eseguire ("+","-" o "=").

Per aggiungere lo sticky bit basta quindi eseguire:

```
$ chmod o+t dir
```

Analogamente per rimuoverlo:

```
$ chmod o-t dir
```

Per quanto riguarda la rappresentazione ottale, per abilitare i permessi speciali si utilizza un'altra cifra prima delle tre consuete. In questo modo i permessi vengono specificati tramite quattro cifre, la cui prima identifica i permessi speciali. Per abilitare lo *sticky bit* si utilizza il valore 1. Perciò i permessi relativi ad una directory con lo sticky bit attivo e con tutti i permessi per il proprietario, per il gruppo e per gli altri, saranno 1777. Per disabilitarlo basta specificare il valore 0 ed i permessi conseguenti saranno 0777. Altri valori vengono utilizzati per gli altri permessi speciali, come descritto in seguito.

È possibile notare la presenza dello *sticky bit* in una directory utilizzando il comando *ls –ld dir*.

```
$ ls –ld dir
drwxrwxrwt 2 user utenti 4096 Aug  8 17:30 dir
```

Dall'output del comando si nota che il permesso di esecuzione per gli altri non è la solita *x* ma una *t*, la quale indica l'abilitazione dello *sticky bit*.

Lo *sticky bit* è particolarmente utile per le directory condivise tra molti utenti e pubbliche, in cui gli utenti possono cioè scrivere. È solitamente abilitato per le directory */tmp* e */var/tmp* poiché in esse diversi utenti possono memorizzare file temporanei ma non si vuole che un utente sia in grado di eliminare i file temporanei di un altro utente.

19.3.3 setuid e setgid per i file eseguibili

Non soltanto i file ma anche i processi in esecuzione hanno un proprietario ed un gruppo. Nel capitolo 4.3 sono stati trattati i processi ed i comandi per visualizzare quelli in esecuzione con i relativi proprietari.

La maggior parte dei programmi quando vengono avviati creano un processo il cui proprietario è l'utente che lo ha invocato; questo perché utilizzano le stesse credenziali dell'utente utilizzato per eseguirli. Questa identità, assieme alla proprietà ed i permessi dei file, determina se un programma può leggere o modificare un file. Tale comportamento è abitualmente corretto ed è un meccanismo per la protezione e la sicurezza del sistema.

Alcune volte però può essere utile, o necessario, eseguire un programma con privilegi diversi. L'esempio classico è il comando *passwd*, utilizzato per impostare la password degli utenti, che anche quando invocato da un utente normale deve essere eseguito con le credenziali dell'utente root per poter leggere e scrivere i relativi file di configurazione come */etc/shadow*.

Per realizzare ciò esistono altri due permessi speciali, oltre lo *sticky bit* precedentemente descritto:

Set user ID (*setuid*): un file eseguibile con abilitato il *setuid* viene eseguito con le credenziali del proprietario del file piuttosto che dell'utente che lo ha eseguito. Se ad esempio il file ha come proprietario l'utente root ed ha il *setuid* abilitato, il programma andrà in esecuzione con i privilegi dell'utente root e sarà quindi in grado di accedere a tutti i file del sistema. I file con il *setuid* abilitato sono riconoscibili dall'output del comando *ls –l* per la presenza della lettera *s* nel permesso di esecuzione per il proprietario, come ad esempio:

$ ls -l /usr/bin/passwd -rwsr-xr-x 1 root root 41284 Apr 9 04:40 /usr/bin/passwd

Set group ID (*setgid*): è simile al *setuid* ma mentre il *setuid* modifica il proprietario, il *setgid* modifica il gruppo del programma in esecuzione con quello del file piuttosto che con quello dell'utente che ha eseguito il programma. Può essere utile per consentire al programma di accedere ai file e alle directory assegnati al medesimo gruppo del proprietario del file. I file con il *setgid* abilitato sono riconoscibili dall'output del comando *ls –l* per la presenza della lettera *s* nel permesso di esecuzione del gruppo, come ad esempio: *rwx-r-sr-x*.

È possibile abilitare tali permessi speciali utilizzando il comando *chmod* sia con la rappresentazione simbolica che ottale.

Nella rappresentazione simbolica si utilizza la lettera *s* per il proprietario (*u*) o per il gruppo (*g*) per abilitare rispettivamente il *setuid* o il *setgid*.

Ad esempio è possibile abilitare il *setuid* con il comando:

```
$ chmod u+s programma
```

mentre per il *setgid* il comando sarà:

```
$ chmod g+s programma
```

Utilizzando il simbolo "–" è possibile disabilitarli, come visto nel capitolo 5.3 per i permessi standard. È anche possibile abilitarli e disabilitarli contemporaneamente; ad esempio con il comando

```
chmod ug-s programma
```

si disabilitano sia il *setuid* che il *setgid* per il file di nome programma.

Nella rappresentazione ottale i due permessi si abilitano in maniera simile allo *sticky bit*, utilizzando un apposito valore che precede i tre permessi classici:

- per lo *sticky bit* il valore da utilizzare era l'1;
- per il *setuid* è il 4;
- per il *setgid* è il 2.

Ovviamente tali valori possono essere combinati come per i permessi classici; ad esempio con il comando

```
$ chmod 6755 programma
```

si abilita sia il *setuid* (4) che il *setgid* (2) ed i permessi sono di lettura, scrittura ed esecuzione per il proprietario e di sola lettura ed esecuzione per il gruppo e per gli altri.

Abitualmente non è necessario abilitare o rimuovere questi permessi, anzi è necessario fare molta attenzione ai file eseguibili con il *setuid* o il *setgid* abilitato e di proprietà dell'utente root: un programma con tali permessi, girando come root, può cancellare e modificare i file di sistema e compromettere l'operatività del sistema. È consigliato abilitarli solamente sui programmi di cui si nutre una certa fiducia o che si compilano personalmente potendone visualizzare il codice sorgente al fine di verificarne il funzionamento. È anche importante evitare di assegnare i permessi di scrittura a tali file per impedire che qualcuno li possa alterare con una versione opportunamente modificata per eseguire azioni malevole. Se ad esempio fosse possibile modificare il comando *passwd*, un utente potrebbe cambiare non solo la propria password ma anche quella di un altro utente o di root. Per questi motivi di sicurezza alcune shell considerano l'effetto di tali permessi solo quando applicati a programmi binari e li ignora quando sono invece applicati a shell script.

19.3.4 setgid per le directory

Mentre il *setuid* non ha effetto per le directory, il *setgid* ha un comportamento differente rispetto a quanto visto per i file eseguibili qualora lo si attivi sulle directory.

Il *setgid* su di una directory infatti stabilisce che i file creati in tale directory devono appartenere allo stesso gruppo della directory piuttosto che al gruppo dell'utente proprietario del file. Questo vale anche se l'utente non è membro del gruppo della directory. Può essere utile ad esempio per le directory condvise che devono contenere dei file relativi ad un gruppo di lavoro. È necessario però che gli utenti abbiano, anche solo come gruppo secondario, lo stesso gruppo di lavoro della directory. Per poter creare un file in tale directory gli utenti devono ugualmente avere i permessi di scrittura su di essa.

Il *setgid* si assegna allo stesso modo di come visto precedentemente per i file eseguibili, l'unica differenza è che ora il comando lo si esegue su di una directory.

```
$ chmod u=rwx,g=srwx /home/progetto
$ chgrp responsabili /home/progetto
$ touch /tmp/prova
$ ls -l /tmp/prova -rw-r--r-- 1 user utenti 0 Aug  8 17:48 /tmp/prova
$ touch /home/progetto/relazione
$ ls -l /home/progetto/relazione -rw-r--r-- 1 user responsabili 0 Aug  8 17:49
progetto/relazione
```

Come si può vedere dall'esempio precedente i due file creati dallo stesso utente nelle due directory (*/tmp* e */home/progetto*) hanno un gruppo differente dovuto al fatto che la directory */home/progetto* ha il *setgid* abilitato. Per permettere agli altri utenti del gruppo di lavoro di modificare il file è necessario cambiare i permessi, come visto nel capitolo 5.3, o modificare la *umask* per i nuovi file.

La modifica del *setgid* ha effetto esclusivamente per i file creati da quel momento in poi e non per quelli già presenti nella directory. Inoltre il proprietario del file può ugualmente modificare il gruppo del file tramite il comando *chgrp* ma deve ancora appartenere al gruppo di assegnazione. Perciò se un utente non è membro del gruppo della directory non può, anche se la directory ha il *setgid* abilitato, assegnare manualmente tale gruppo ai suoi file.

Comandi

ln - Crea link simbolici e hard link

Esercitazione proposta 1 (ove non specificato diversamente utilizzare la propria utenza)

1. Con il proprio utente creare un file di nome file1 e contenuto a scelta.

2. Creare un *hard link* a file1 di nome file2.

3. Creare un link simbolico a file2 di nome link1.

4. Verificare dall'output del comando *ls* che i file siano stati creati correttamente ed i relativi *inode*. Qual è la grandezza del file link1? Quali sono gli *inode* dei tre file?

5. Modificare il contenuto del file2.

6. Visualizzare il contenuto dei file file1 e link1. È cambiato? Perché?

7. Modificare i permessi di link1 in 555.

8. Verificare la modifica cosa ha provocato.

9. Modificare il proprietario del file file2 con un altro utente qualsiasi. (tramite utente root)

10. Cambiari i permessi del file file2 con altri a scelta. (tramite utente root)

11. Rinominare file2 in file3. (tramite utente root)

12. Visualizzare tramite il comando *ls* i permessi dei tre file. Ci sono state modifiche nei permessi anche negli altri file? In quali? Perché?

13. Visualizzare il contenuto del file file1. Funziona? Perché?

14. Visualizzare il contenuto del file link1. Funziona? Perché?

Soluzione dell'esercitazione 1

1. *$ ls –l / > file1;*

2. *$ ln file1 file2;*

3. *$ ln –s file2 link1;*

4. *$ ls –li;*
La dimensione di link1 è pari al numero di caratteri del percorso utilizzato per creare il link. I file file1 e file2 condividono lo stesso *inode* essendo *hard link*, mentre link1 essendo un link simbolico ha un altro inode.

5. *$ echo prova > file2;*

6. *$ more file1; $ more link1;*
Essendo tutti i file collegati le modifiche si sono ripercosse su tutti e tre i file.

7. *$ chmod 555 link1;*

8. *$ ls –l;*
Il comando *chmod* sul link simbolico ha modificato i permessi di file2 (e file1) e non quelli del file link1 che sono rimasti come erano prima (777).

9. *# chown user2 file2;*

10. *# chmod 777 file2;*

11. *# mv file2 file3;*

12. *$ ls –l;*
Le modifiche hanno interessato anche file1 poiché è un *hard link*.

13. *$ more file1;*
Funziona perché file1 e file3 sono *hard link* allo stesso *inode*.

14. *$ more link1;*
link1: No such file or directory
Non funziona perché link1 era un link simbolico a file2, il quale essendo stato rinominato ha provocato che ora link1 è orfano.

Esercitazione proposta 2 (ove non specificato diversamente utilizzare la propria utenza)

1. Con la propria utenza creare una directory.

2. Assegnare i permessi di scrittura a tutti per la directory.

3. Verificare i permessi della directory.

4. Creare un file di nome fileProva1 all'interno di questa directory con contenuto a scelta.

5. Rimuovere i permessi di scrittura per il gruppo e per gli altri per il file fileProva1.

6. Provare con l'utente user2 a modificare il contenuto del file fileProva1. È possibile?

7. Provare con l'utente user2 a cancellare il file fileProva1. È possibile?

8. Con la propria utenza abilitare lo *sticky bit* per la directory.

9. Creare un file di nome fileProva2 all'interno della directory con contenuto a scelta.

10. Assegnare i permessi di scrittura per il gruppo e per gli altri per il file fileProva2.

11. Provare con l'utente user2 a modificare il contenuto del file fileProva2. È possibile?

12. Provare con l'utente user2 a cancellare il file fileProva. È possibile?

Soluzione dell'esercitazione 2

1. *$ mkdir /tmp/provaSB;*

2. *$ chmod 777 /tmp/provaSB;*

3. *$ ls –ld /tmp/provaSB;*

4. *$ ls –l / > /tmp/provaSB /fileProva1;*

5. *$ chmod g-w,o-w /tmp/provaSB /fileProva1;*

6. *$ echo prova > /tmp/provaSB/fileProva1;*
bash: /tmp/provaSB/fileProva: Permission denied
Il file è protetto da scrittura e non è possibile modificarlo.

7. *$ rm /tmp/provaSB /fileProva1;*
rm: remove write-protected regular file `/tmp/provaSB/fileProva1'? y
Pur non avendo i permessi di scrittura sul file fileProva1 è possibile cancellarlo.

8. *$ chmod o+t /tmp/provaSB;*

9. *$ ls –l / > /tmp/provaSB /fileProva2;*

10. *$ chmod g+w,o+w /tmp/provaSB /fileProva2;*

11. *$ echo prova > /tmp/provaSB/fileProva2;*
È possibile modificare il file in quanto tutti hanno i permessi di scrittura.

12. *$ rm /tmp/provaSB /fileProva2;*
rm: cannot remove `/tmp/provaSB/fileProva2': Operation not permitted
Pur avendo i permessi di scrittura sul file fileProva2 (e potendo quindi modificarne il contenuto) non è possibile cancellarlo poiché la directory ha lo *sticky bit* abilitato.

Esercitazione proposta 3 (ove non specificato diversamente utilizzare la propria utenza)

1. Con la propria utenza creare una directory, abilitare il *setgid* e dare i permessi di scrittura a tutti.

2. Assegnare alla directory un gruppo diverso da quello principale dell'utente user2.

3. Verificare i permessi della directory.

4. Come utente user2 creare un file di nome fileProva3 in quella directory.

5. Verificare il proprietario ed il gruppo del file. Chi è il proprietario? Qual è il gruppo?

6. Come utente user2 cambiare il gruppo del proprio file con il proprio gruppo. È possibile farlo?

7. Verificare che la modifica sia avvenuta con successo.

8. Come utente user2 ripristinare il gruppo del proprio file con quello originario della directory. È possibile farlo?

Soluzione dell'esercitazione 3

1. *$ mkdir /tmp/provaSG; chmod g+s,a+w /tmp/provaSG;*

2. *$ chgrp utenti /tmp/provaSG;*

3. *$ ls –ld /tmp/provaSG;*

4. *$ touch /tmp/provaSG/fileProva3;*

5. *$ ls –l /tmp/provaSG;*
Il proprietario è l'utente user2 mentre il gruppo non è quello di user2 ma è quello impostato per la directory.

6. *$ chgrp user2 /tmp/provaSG/fileProva3;*
Si il proprietario del file può ancora modificare il gruppo del file.

7. *$ ls –l /tmp/provaSG;*

8. *$ chgrp utenti /tmp/provaSG/fileProva3;*
Dipende: se l'utente user2 ha come affiliazione secondaria il gruppo della directory può farlo, altrimenti non può più ripristinare il gruppo del file come era originariamente non appena creato.

Domande

1. In quali casi potrebbe essere necessario per un programma, il cui proprietario è root, aver abilitato il *setuid*?

A. Sempre, tale permesso è necessario per i file eseguibili.
B. Quando il programma deve avere accesso ad un *device file*.
C. Solamente quanto richiede i privilegi di root per poter funzionare correttamente.
D. Quando il programma deve poter accedere agli *userID* degli utenti.
E. Mai, questo permesso è una grave falla di sicurezza.

2. Quali delle seguenti voci possono essere memorizzate nella directory */var*? (scegliere più risposte)

A. Librerie
B. File di log
C. File di configurazione
D. File eseguibili per l'amministrazione del sistema
E. File relativi allo *spool* di stampa

3. Abitualmente una periferica di memorizzazione USB viene montata in una sottodirectory di? (scegliere più risposte) A. */mount*

B. */mnt* C. */usb* D. */media*

4. Dove vengono solitamente memorizzate le librerie necessarie al funzionamenti dei comandi presenti in */bin* e */sbin*?

A. */lib* B. */bin* C. */sbin* D. */usr/lib*

5. Solitamente, in merito alle directory */tmp* e */var/tmp*? (scegliere più risposte)

A. in */tmp* vengono memorizzati i file temporanei che persistono al riavvio del sistema.
B. in */var/tmp* vengono memorizzati i file temporanei che persistono al riavvio del sistema.
C. in */tmp* vengono memorizzati i file temporanei che possono essere cancellati durante il riavvio del sistema.

D. in */var/tmp* vengono memorizzati i file temporanei che possono essere cancellati durante il riavvio del sistema.

6. Quali comandi possono essere utilizzati per abilitare lo *sticky bit* su di una directory? (scegliere più risposte)

A. *chmod o+t dir*
B. *chmod u+t dir*
C. *chmod o=rt dir*
D. *chmod 1755 dir*
E. *chmod 5755 dir*

7. L'output del commando *ls –l* mostra il seguente campo: *drwxrwsr-t*
Quali delle seguenti affermazioni sono vere? (scegliere più risposte)

A. Si tratta di un file
B. Si tratta di una directory
C. Ha lo *sticky bit* abilitato
D. Ha il *setuid* abilitato
E. Ha il *setgid* abilitato

8. Quali dei seguenti output del comando *ls –l* sono corretti per un file con il *setuid* abilitato? (scegliere più risposte)

A. *rwxrwsrwx*
B. *rwsrwxrwx*
C. *rwxrwxrwt*
D. *rwsrws---*
E. *rwtrwxrwx*

9. Quale dei seguenti comandi (e relativa opzione) permette di creare un link simbolico?

A. *ln* B. *ln –s* C. *ln –h* D. *ls –d*

10. Quale dei seguenti comandi (e relativa opzione) permette di creare un *hard link*?

A. *ln* B. *ln –s* C. *ln –h* D. *ls –d*

11. In quale directory sono presenti i più importanti file di configurazione del sistema?

A. */root* B. */var* C. */bin* D. */etc* E. */sbin*

12. Quale opzione/gruppo di opzioni del comando *ls* permette, passandogli come argomento il nome di una directory, di visualizzare i permessi della directory stessa anziché dei file in essi contenuti?

A. *–l* B. *–ld* C. *–il* D. *–al*

13. Quale comando può essere utilizzato per abilitare contemporaneamente per un file il *setuid* ed il *setgid*

ISBN: 9788866850090

A. *chmod 1555 programma*
B. *chmod 4555 programma*
C. *chmod 5555 programma*
D. *chmod 6555 programma*

14. Quali comandi permettono di abilitare il *setuid* su un file? (scegliere più risposte)

A. *chmod 1555 programma*
B. *chmod 4555 programma*
C. *chmod 2555 programma*
D. *chmod u+s programma*
E. *chmod g+s programma*
F. *chmod u+t programma*

Risposte

1. C
2. B, E
3. B, D
4. A
5. B, C
6. A, D
7. B, C, E
8. B, D
9. B
10. A
11. D
12. B
13. D
14. B, D

Appendice 1 - Il syllabus LPI Linux Essentials

Si riporta di seguito il syllabus della certificazione LPI Essentials sia nella versione inglese che in italiano.

Tali documenti sono stati presi da:

- http://www.lpi.org/linux-certifications/introductory-programs/linux-essentials (versione inglese)
- http://www.lpi-italia.org/certificazione/programmi/linux-essentials/ (versione italiana)

Ulteriori link utili sono:

- http://wiki.lpi.org/wiki/LinuxEssentials (in inglese)

http://wiki.lpi.org/wiki/LinuxEssentials(IT) (in italiano)

Argomento 1: La comunità Linux e le carriere lavorative nell'open source

1.1 Evoluzione di Linux e diffusione dei Sistemi Operativi

Peso: 2

Descrizione: Conoscenza dello sviluppo di Linux e le principali distribuzioni

Aree di conoscenza chiave

- Filosofia open source
- Distribuzioni
- Sistemi Embedded

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Android
- Debian
- CentOS

1.2 Principali applicazioni open source

Peso: 2

Descrizione: Conoscenza dei principali applicativi e loro usi.

Aree di conoscenza chiave

- Applicativi Desktop
- Applicativi Server

- Applicativi Mobile
- Sviluppo Linguaggi
- Tool di gestione pacchetti e repository

Quella che segue è una lista parziale dei file, termini ed utilità usate

- OpenOffice.org, LibreOffice, Thunderbird, Firefox
- Blender, Gimp, Audacity, ImageMagick
- Apache, MySQL, PostgreSQL
- NFS, Samba, OpenLDAP, Postfix, DNS, DHCP
- C, Java, Perl, shell, Python, PHP

1.3 Conoscenza del Software open source e tipi di Licenze

Peso: 1

Descrizione: Comunità aperte e licenze open source per il business.

Aree di conoscenza chiave

- Licenze
- Free Software Foundation (FSF), open source Initiative (OSI)

Quella che segue è una lista parziale dei file, termini ed utilità usate

- GPL, BSD, Creative Commons
- Free Software, open source Software, FOSS, FLOSS
- Modelli di business open source

1.4 Abilità ICT e lavoro su Linux

Peso: 2

Descrizione: Abilità base su Tecnologie dell'informazione e della comunicazione (ICT) e lavoro su Linux.

Aree di conoscenza chiave

- Desktop
- La linea di comando
- Settori che utilizzano Linux, Cloud Computing e Virtualizzazione

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Usare un browser, concetti di privacy, opzioni di configurazione, ricerca sul web e salvataggio dei contenuti
- Terminale e Console
- Password Privacy e tool relativi
- Uso dei comuni applicativi open source per presentazioni e progetti

2.1 Riga di comando base

Peso 2

Descrizione Basi sull'uso della linea di comando Linux.

Aree di conoscenza chiave

- Shell base
- Comandi di formattazione
- Lavorare con le opzioni
- Variabili
- Globbing
- Quoting

***Quella che segue è una lista parziale dei file, termini ed utilità usate*:**

- echo
- history
- variabili
- PATH env
- which

2.2 Usare la linea di comando per cercare aiuto

Peso 2

Descrizione Lanciare comandi help e navigare tra i vari sistemi di aiuto.

Aree di conoscenza chiave

- Man
- Info

Quella che segue è una lista parziale dei file, termini ed utilità usate

- man
- info
- Man pages
- /usr/share/doc
- locate

2.3 Usare directory ed elencare file

Peso 2

Descrizione Navigazione della direcory home e del sistema, elencare i file in differenti posizioni.

Aree di conoscenza chiave

- File e directory
- File e directory nascosti
- Home
- Percorsi relativi ed assoluti

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Opzioni comuni per ls
- Elenchi ricorsivi
- cd
- . e ..
- home e ~

2.4 Creare, Spostare e Cancellare file

Peso 2

Descrizione Creare, spostare e cancellare file e directory nella home directory.

Aree di conoscenza chiave

- File e directory
- Case sensitivity
- Globbing e quoting

Quella che segue è una lista parziale dei file, termini ed utilità usate

- mv, cp, rm, touch
- mkdir, rmdir

Argomento 3: La potenza della linea di comando

3.1 Archiviazione file sulla linea di comando

Peso 2

Descrizione Archiviazione file nella directory home.

Aree di conoscenza chiave

- File e directory
- Archivi e compressione

Quella che segue è una lista parziale dei file, termini ed utilità usate

- tar
- Opzioni comuni di tar
- gzip, bzip2
- zip, unzip

3.2 Ricerca ed Estrazione dati da file

Peso 4

Descrizione Ricerca ed estrazione dati da file nella directory home.

Aree di conoscenza chiave

- Pipes nella linea di comando
- Redirezione dell'I/O
- Basi delle espressioni regolari POSIX (., [ ], *, ?)

Quella che segue è una lista parziale dei file, termini ed utilità usate

- find
- grep
- less
- head, tail
- sort
- cut
- wc

3.3 Inserire comandi negli script

Peso 4

Descrizione Inserire comandi ripetitivi in un semplice script.

Aree di conoscenza chiave

- Text editing base
- Shell scripting base

Quella che segue è una lista parziale dei file, termini ed utilità usate

- /bin/sh
- Variabili
- Argomenti
- for loops
- echo
- Exit status

Argomento 4: Il sistema operativo Linux

4.1 Scegliere un sistema operativo

Peso 1

Descrizione Conoscenza dei principali sistemi operativi e distribuzioni Linux.

Aree di conoscenza chiave

- Differenze Windows, Mac, Linux
- Ciclo di vita delle distribuzioni

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Interfaccia grafica versus linea di comando, configurazione desktop
- Cicli di mantenimento, Beta e versioni stabili

4.2 Comprendere l'hardware

Peso 2

Descrizione Familiarità con i componenti che vanno a costruire un Desktop o un Server.

Aree di conoscenza chiave

- Hardware

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Hard drive e partizioni, schede madri, processori, alimentatori, drive ottici, periferiche
- Tipi di display
- Driver

4.3 Dove archiviare i dati

Peso 3

Descrizione Dove i vari tipi di informazioni vengono archiviati su Linux.

Aree di conoscenza chiave

- Kernel
- Processi
- syslog, klog, dmesg
- /lib, /usr/lib, /etc, /var/log

Quella che segue è una lista parziale dei file, termini ed utilità usate

- Programmi, librerie, pacchetti e database pacchetti, configurazione di sistema
- Processi e tabelle dei processi, indirizzi memoria, messaging e logging del sistema
- ps, top, free

4.4 Il tuo computer in Rete

Peso 2

Descrizione Interrogazione delle impostazioni di rete vitali, determinare i requisiti base per un computer in una LAN.

Aree di conoscenza chiave

- Internet, network, router
- Domain Name Service
- Configurazione di rete

Quella che segue è una lista parziale dei file, termini ed utilità usate

- route
- resolv.conf
- IPv4, IPv6
- ifconfig
- netstat
- ping

Argomento 5: Sicurezza e permessi dei file (peso: 7)

5.1 Sicurezza base ed identificazione dei tipi di utenti

Peso 2

Descrizione Vari tipi di utenti su un sistema Linux.

Aree di conoscenza chiave

- Utenti root e standard
- Utenti di sistema

Quella che segue è una lista parziale dei file, termini ed utilità usate

- /etc/passwd, /etc/group
- id, who, w
- sudo

5.2 Creare utenti e gruppi

Peso 2

Descrizione Creare utenti e gruppi su un sistema Linux.

Aree di conoscenza base:

- Comandi User e group
- ID utenti

Quella che segue è una lista parziale dei file, termini ed utilità usate

- /etc/passwd, /etc/shadow, /etc/group
- id, last useradd, groupadd
- passwd

5.3 Gestione permessi file e proprietari

Peso 2

Descrizione Comprensione e manipolazione dei permessi dei file e configurazione delle proprietà.

Aree di conoscenza chiave

- Permessi e proprietari di file/directory

Quella che segue è una lista parziale dei file, termini ed utilità usate

- ls -l
- chmod, chown

5.4 Directory e File speciali

Peso 1

Descrizione Directory e file speciali su un sistema Linux compresi permessi speciali.

Aree di conoscenza chiave

- File di sistema, librerie
- Link simbolici

Quella che segue è una lista parziale dei file, termini ed utilità usate

- /etc, /var
- /tmp, /var/tmp e Sticky Bit
- ls -d
- ln -s

Appendice 2 - Licenza per Documentazione Libera GNU

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. 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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

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 under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.