Linux Base - Capitolo n. 2

Edizioni ByteMan (22-05-2005)

revisione: 25/01/2008

File System (parte 2)

Considerazioni sui File System

Un file system è, nel linguaggio corrente, un meccanismo con il quale i file sono immagazzinati e organizzati su un dispositivo di archiviazione, come un hard disk o un CD-ROM, e costituisce parte integrante di qualsiasi sistema operativo moderno. Più correttamente, un file system è l'insieme dei tipi di dati astratti necessari per la memorizzazione, l'organizzazione gerarchica, la manipolazione, la navigazione, l'accesso e la lettura dei dati.

I file system più comuni si appoggiano a dispositivi di archiviazione che offrono l'accesso ad un array di blocchi di dimensione fissa, generalmente chiamati settori, tipicamente di 512 byte l'uno. Il software che li supporta è responsabile dell'organizzazione di questi settori in file e directory, e tiene traccia dei settori che appartengono ai vari file e dei settori non utilizzati.

I file system tipicamente hanno tabelle che associano i nomi dei file con i file, usualmente collegando il nome del file ad un indice in una tabella di allocazione dei file (file allocation table) di qualche genere, come la FAT di un file system MS-DOS, o un inode in un file system di tipo Unix.
Le strutture delle directory possono essere ad un solo livello, oppure possono permettere una struttura gerarchica in cui delle directory possono contenere sottodirectory.

Il File System Journaled

I requisiti per un filesystem di moderna concezione sono: capacità di supportare grossi volumi fisici (oramai centinaia di gigabyte), integrità dei dati, riduzione della frammentazione interna ed esterna, recovery veloce in caso di crash e manipolazione in modo veloce ed efficiente sia dei file voluminosi sia dei file piccoli.

Un File System Journaled (con libro giornale) ha la capacità di recuperare lo stato consistente dell'hard disk in pochi secondi, la tecnica di recovery, ereditata dai database, si basa sulle transazioni. In caso di 'crash' del sistema, lo stato consistente del disco viene recuperato leggendo il Journal (libro giornale) e riapplicando le modifiche: quindi lo stato precedente.

In un filesystem non journaled lo stato consistente del disco passa attraverso l'analisi dei meta-data (i-node, directory, i-node map...) che richiede la scansione completa (blocco per blocco) del volume logico. Questo avviene in quanto la ricerca dei blocchi liberi avviene mappandoli in una sequenza di bit (detta comunemente 'struttura a bitmap'); più il filesystem cresce in dimensioni, più lo spazio occupato cresce di conseguenza: l'algoritmo di ricerca utilizzato è sequenziale.
ovviamente per filesystem di piccole dimensioni il sistema è comunque decisamente efficiente, e non penalizza le prestazioni del server.

Esempi di File System

L'elenco seguente include solo alcuni dei moltissimi tipi di file system conosciuti, è soprattutto orientato al mondo Linux, ma non può trascurare i prodotti Microsoft.

- **FAT32** (Microsoft) rappresenta una versione avanzata di **FAT** (**F**ile **A**llocation **T**able) che può essere utilizzata con dischi rigidi fino a **32GB**, ed offre la compatibilità con molti sistemi operativi. Viene spesso utilizzata in sistemi multi-boot come area di transito tra un sistema operativo e l'altro.
  L'utilizzo della versione a 16 bit **FAT16**, con il limite dei 2 GB, è ormai limitato solo a quei casi in cui sia indispensabile condividere informazioni anche con MS-DOS o Windows 3.11.
  Mentre la versione **FAT12** è stata utilizzata solo per i floppy disk.
  In questo genere di File System è necessario procedere periodicamente alla **deframmentazione**.

Non è previsto alcun meccanismo di riservatezza sui file, a parte l'attivazione dell'attributo **hidden** (file nascosti).

- **NTFS** (Microsoft), acronimo di **N**ew **T**echnology **F**ile **S**ystem, viene raccomandato per Windows XP ed usato con le varie versioni di Windows NT. Estende le caratteristiche di base di **FAT32** al fine di migliorarne la sicurezza, e la compressione; consente la gestione di hard-disk fino a **2TB** .
  Occorre tenere presente però che una partizione formattata con **NTFS** non sarà accessibile da parte di sistemi operativi come Win98 che lavorano su FAT32.
  NTFS è un file system journaled, ma ha bisogno di una **deframmentazione** periodica.
  Tutti i file e le directory possono essere protetti (lettura, scrittura, modifica, etc.), ma occore tenere presente che tutta questa sicurezza è bypassabile attraverso un boot disk Linux.
  Al fine di incrementare la velocità NTFS memorizza i file piccoli nella Master File Table (MFT); questa procedura favorisce l'accesso delle testine di lettura dell'hard disk nella posizioni iniziali (raggiungibili più velocemente) senza andare a cercare i dati nelle tracce centrali.
  Altra caratteristica di questo file system è la compressione dei dati memorizzati.

- **Ext2** (Linux), è un filesystem efficiente e veloce, che nel corso degli anni ha dimostrato di essere in grado (con i dovuti miglioramenti di cui ha goduto) di tenere il passo della tecnologia. Questo filesystem non è però journaled, e infatti i problemi che sono causati da un arresto forzato del sistema sono tristemente noti.
  Come tutti i file system per Linux dispone di un'ottimo sistema per garantire la riservatezza di file e directory (permessi di lettura, scrittura, esecuzione, etc.).

- **Ext3** (Linux), è l'evoluzione di Ext2, con l'aggiunta del supporto per il journaling. Effettivamente, Ext2 è sempre stato reputato un ottimo filesystem, con l'unico neo dato da una certa debolezza nei confronti delle interruzioni di sistema; il journaling risolve il problema al prezzo inevitabile di un certo degrado nelle prestazioni. Infatti, nonostante gli autori assicurino che sono state apportate delle migliorie rispetto ad Ext2 per quanto riguarda la velocità, gli accessi dovuti all'aggiornamento del journal rendono Ext3 piuttosto lento rispetto al suo antenato. RedHat assicura di aver condotto test approfonditi che ne garantirebbero la stabilità.
  Ext3 è un filesystem basato su blocchi, supporta nomi lunghi, link simbolici veloci, attributi dei file in stile UNIX, blocchi riservati per il superuser e partizioni fino a **4 TB**.

- **ReiserFS** (Linux), fu sviluppato in origine da Hans Reiser, ma è rapidamente cresciuto con l'apporto di molti altri sviluppatori. È stato il primo filesystem journaled per Linux a raggiungere ufficialmente una certa "stabilità". ReiserFS continua a distinguersi per la grande cura che i programmatori e i manutentori dedicano alla sua distribuzione.
  Tecnicamente, ReiserFS usa un sistema ad alberi bilanciati molto sofisticato che permette performance migliori rispetto all'approccio convenzionale a blocchi usato ad esempio da Ext2; inoltre, permette una strategia di immagazzinamento dei piccoli file di gran lunga migliore.
  Permette di creare partizioni e file grandi fino a **17 TB**, ovviamente questo è un limite teorico, perché il kernel impone vincoli che costringono (si fa per dire) a minori dimensioni. In ogni caso, se in futuro il kernel abbatterà questi vincoli, non occorreranno grandi cambiamenti al filesystem.

- **XFS** (Silicon Graphics), sviluppato originariamente (fin dal 1994) per IRIX è stato reso disponibile per architetture x86, ma è stato fatto girare anche sotto sistemi DEC Alpha, IA64 e Sparc64.
  Tecnicamente, XFS è un filesystem journaled interamente progettato a 64 bit; di per sé può teoricamente gestire partizioni grandi fino a **1 EsaByte** (un miliardo di Gigabyte!). Una caratteristica importante di XFS è la presenza degli strumenti per il backup ed il restore del filesystem, che per di più utilizzano un formato comune a Linux e IRIX: i backup fatti sotto un sistema saranno quindi pienamente portabili sull'altro, con evidenti vantaggi di flessibilità e migrazione.
  Silicon graphics riporta di aver condotto test esaustivi di stabilità su una grande varietà di macchine e configurazioni. Come sempre queste informazioni, soprattutto se date in maniera così generica, sono da prendere con le pinze; ma in effetti XFS appare un prodotto già abbastanza stabile.

- **JFS** (IBM), utilizzato da tempo su AIX e nei server enterprise IBM, è stato ufficialmente rilasciato alla comunità Linux. Questo atteggiamento di collaborazione è molto promettente, e dimostra davvero come e fino a che livello "Big Blue" creda nell'Open Source.
  Purtroppo da varie prove che sono state effettuate, JFS è risultato il filesystem meno stabile tra i quattro journaled di Linux che abbiamo preso in esame.
  Una peculiarità di JFS è l'uso della deframmentazione, strumento tipico di filesystem estranei a Linux, ma pare che nonostante JFS adotti strategie avanzate per limitare la frammentazione dei file, queste non siano sufficienti.
  La dimensione massima di un file in JFS è **512 TB** con blocchi di 512 byte.

Altri File System

In questo elenco, per nulla completo, vengono brevemente citati altri tipi di file system usati o riconosciuti da Linux.

- **Minix** Il più vecchio e per questo si presume che sia il più affidabile dei file system per Linux. E' però piuttosto limitato (mancano alcuni time stamp, e i nomi di file sono al massimo di 30 caratteri) e di capacità ristrette (al massimo 64 MB per filesystem).
- **Xia** Una versione modificata del filesystem Minix, che alza i limiti sulla lunghezza dei nomi di file e sulla dimensione dei filesystem, ma non introduce nuove caratteristiche. Non è molto conosciuto, ma si dice funzioni molto bene.
- **Ext** Una versione più vecchia dell'Ext2 che però non ha garantito la compatibilità in avanti. Non è quasi mai più usato in nuove installazioni, perché la maggior parte delle persone si è convertita all'Ext2.
- **MsDos** Compatibile con i filesystem FAT di MS-DOS (ed OS/2 e Windows NT).
- **UsmDos** Estende il driver MsDos sotto Linux, in modo da avere i nomi di file lunghi, i permessi, i link e i file di device, e da assegnare ciascun file ad un utente. In questo modo è possibile usare un normale filesystem MsDos come se fosse uno Linux, eliminando la necessità di avere una partizione separata per quest'ultimo.
- **Iso9660** Il filesystem standard per i CD-ROM: viene supportata automaticamente l'estensione Rock Ridge allo standard per i CD-ROM, che permette di avere i nomi dei file lunghi.
- **Nfs** Un filesystem di rete che permette la condivisione dei file tra vari computer per avervi un accesso più facile.
- **Hpfs** Il filesystem di OS/2.
- **SysV** I filesystem SystemV/386, Coherent e Xenix.

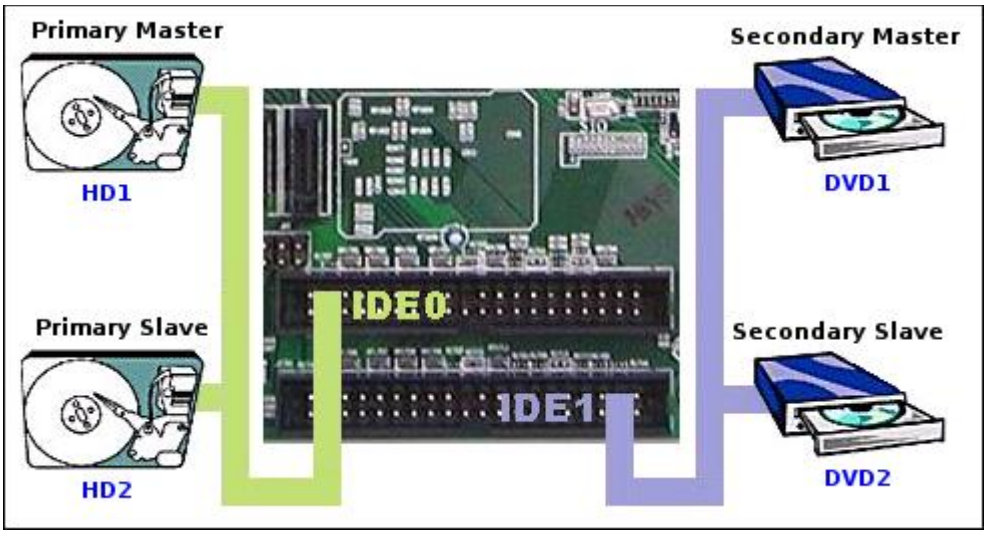
I nomi dei dispositivi

C'è un principio guida in Unix che è stato ereditato anche da Linux: **ogni cosa è un file**. Non si sottraggono a questa regola nemmeno i dispositivi: dischi, stampanti, etc. Infatti, a questo scopo, esiste una directory di nome **/dev** (devices) che contiene tutti i riferimenti file dei vari dispositivi.

I nomi dati da Linux a dischi e partizioni, per esempio, sono diversi da quelli usati da altri noti sistemi operativi.
In Windows abbiamo **A:** per indicare la prima unità floppy, **B:** per la seconda unità floppy, **C:** per il primo hard-disk o per la prima partizione del primo hard-disk.
In Linux la situazione è diversa e fa riferimento alla directory **/dev**, per esempio ecco i nomi di alcuni dispositivi:

/dev/fd0 primo dispositivo floppy
/dev/fd1 secondo dispositivo floppy
/dev/hda1 primo hard-disk

Prima di continuare è però opportuno fare alcune considerazioni sull'interfaccia più comune usata per controllare gli hard-disk, e cioè IDE, poi evolutasi in EIDE e ATA. I due connettori disponibili, identificati dalle sigle **IDE0** (Primario) e **IDE1** (Secondario), consentono la connessione di un massimo di 4 unità, due per ogni connettore. Le 2 periferiche che fanno capo a ciascun connettore sono identificate come **Master** e **Slave**.



Supponiamo che nel caso illustrato in figura i due hard-disk abbiano ciascuno una singola partizione, allora ecco come saranno contrassegnati i vari dispositivi:

	HD1	HD2	DVD1	DVD2
Windows	C:	D:	E:	F:
Linux	hda	hdb	hdc	hdd

Ora accade che per varie esigenze si decida di partizionare i due hard-disk nel seguente modo:
HD1 con una primaria ed una estesa con dentro due logiche.
HD2 con una primaria ed una estesa con dentro una logica.
La situazione allora cambierà come nella tabella seguente:

	HD1	HD2	DVD1	DVD2
Windows	C: E: F:	D: G:	H:	I:
Linux	hda1 hda5 hda6	hdb1 hdb5	hdc	hdd

Come si vede, nel caso di Windows, è accaduto una piccolo terremoto nell'assegnazione dei nomi, in

quanto è il sistema operativo che gestisce la nomenclatura secondo propri criteri.
Nel caso di Linux invece ciascun dispositivo ha mantenuto intatta la radice del nome, per esempio per l'hard-disk **HD1** le varie sue partizioni continueranno ad essere identificate dalla radice **hda** con il solo accorgimento che le partizioni primarie saranno numerate da **1** a **4**, mentre le partizioni logiche contenute nelle estese saranno numerate da **5** in avanti.

Comandi Base (parte 2)

Linux, al contrario di molti altri S.O., è in grado di gestire tutti i tipi di filesystem più comuni, dal tradizionale FAT di Windows 9x a NTFS di Windows NT, dalle partizioni BSD a quella di Solaris, ecc. Affinché siano utilizzabili da Linux, però, è necessario che il kernel sia compilato con il supporto per il tipo di fs che vorremo montare (to mount = montare, innestare). Un kernel standard, presente in una generica distribuzione, ha già compilato il supporto per i filesystem più utilizzati.
Abbiamo visto che la directory **/dev** contiene i riferimenti file dei vari dispositivi presenti nel sistema, la directory **/mnt**, presente nella maggior parte delle distribuzioni, serve invece per innestare (montare) nell'albero del file system i dispositivi che si desidera utilizzare.
Il comando che gestisce queste situazioni è **mount** la cui forma standard è la seguente:

```
mount -t type device dir
```

dove

- l'opzione **-t** serve a specificare il tipo di file system, e quindi **type** può assumere, per esempio, i seguenti valori: *auto, ext2, ext3, iso9660, msdos, ntfs, reiserfs, udf, usbfs, vfat,...*
- **device** indica il dispositivo da montare, per esempio: */dev/fd0, /dev/hdb2, ...*
- **dir** indica la directory su cui va innestato il dispositivo. Si noti che tale directory deve esistere, altrimenti deve essere prima creata, generalmente viene predisposta sotto la directory **/mnt**.

Ecco alcuni casi:

```
mount -t msdos /dev/fd0 /mnt/floppy        monta un floppy msdos
mount -t auto /dev/hdb2 /mnt/hdb2          monta la partizione hdb2 in automatico
mount -t iso9660 /dev/hdc /mnt/cdrom       monta il lettore di CD
```

Poichè alcuni **mount** possono essere di più frequente utilizzo, le distribuzioni maggiormente diffuse di GNU/Linux inseriscono i dati utilizzati nel file **/etc/fstab** per facilitare ed automatizzare le operazioni. Per esempio se nel file appena citato è presente una riga di configurazione simile:

```
/dev/hdc /mnt/cdrom iso9660 owner,noauto,ro 0 0
```

basterà utilizzare il comando mount con una serie ridotta di opzioni:

```
mount /dev/hdc
```

L'operazione inversa al comando mount viene svolta da **umount** che smonta il file system aggiunto.
Riprendendo l'esempio del cd-rom:

```
umount /dev/hdc
```

Per effettuare questa operazione, ovviamente bisogna trovarsi in una directory esterna al file system da smontare!
Per i dispositivi estraibili ricordarsi sempre di effettuare lo smontaggio prima di estrarre fisicamente il supporto. In alcuni casi (es. unità CD) il sistema operativo blocca il meccanismo di espulsione, in altri casi ciò non è possibile (es. floppy, chiavette USB). Occorre farci l'abitudine.

I permessi per l'accesso ai file

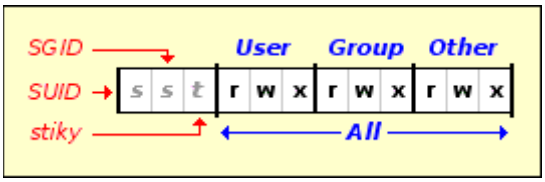
Ad ogni file Linux associa sempre l'utente che ne è proprietario (il cosiddetto owner) ed un gruppo di appartenenza, secondo il meccanismo degli identificatori di utente e gruppo (uid e gid).
Il controllo di accesso ai file segue un modello abbastanza semplice che prevede **tre permessi** fondamentali strutturati su **tre livelli** di accesso. I tre permessi di base associati ad ogni file sono:

- il permesso di lettura (indicato con la lettera **r**, dall'inglese **r**ead)
- il permesso di scrittura (indicato con la lettera **w**, dall'inglese **w**rite)
- il permesso di esecuzione (indicato con la lettera **x**, dall'inglese e**x**ecute)

mentre i tre livelli su cui sono divisi i privilegi sono:

- i privilegi per l'utente (**u**ser) proprietario del file
- i privilegi per un qualunque utente faccia parte del gruppo (**g**roup) cui appartiene il file
- i privilegi per tutti gli altri (**o**ther) utenti

L'insieme dei permessi viene espresso con un numero a **12 bit** (espresso in ottale); di questi i nove meno significativi sono usati a gruppi di tre per indicare i permessi base di lettura, scrittura ed esecuzione e sono applicati rispettivamente al proprietario, al gruppo, a tutti gli altri.



I 3 bit più significativi non rivestono, per il momento, una particolare importanza e saranno oggetto di attenzione prossimamente. Il significato del tipo di accesso dipende a seconda che venga applicato ai file o alle directory.
Per un **file**: l'accesso in lettura permette di leggerne il contenuto, l'accesso in scrittura permette di modificarne il contenuto, l'accesso in esecuzione permette di eseguirlo (ammesso che si tratti di un eseguibile binario o di uno script di qualunque tipo).
Per una **directory**: l'accesso in lettura permette di leggerne il contenuto (poter conoscere l'elenco dei file in esse contenuti, di qualunque tipo essi siano), l'accesso in scrittura permette di modificarne il contenuto (creare, eliminare e rinominare i file), l'accesso in esecuzione permette di attraversare una directory.

I permessi si possono esprimere in due forme diverse: o con una una stringa alfabetica o con un numero ottale. La stringa utilizza le lettere **r**, **w**, **x** per rappresentare i permessi di lettura, scrittura ed esecuzione, mentre quando si utilizza la notazione ottale, il numero **4** rappresenta un permesso in lettura, il numero **2** rappresenta un permesso in scrittura e il numero **1** rappresenta un permesso in esecuzione. Si ottiene la combinazione di più tipi di permesso di accesso sommando le cifre necessarie.

La notazione numerica ottale è preferibile rispetto a quella simbolica essendo più completa e immediata. In particolare, se il numero non utilizza tutte le cifre, si intende che manchino quelle anteriori e che queste siano semplicemente azzerate. Per esempio, il permesso **755** (pari a **0755**) indica che l'utente proprietario può leggere, modificare ed eseguire il file, mentre sia gli utenti del gruppo sia gli altri possono solo leggere ed eseguire il file.

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea e al gruppo principale dell'utente stesso. I permessi gli vengono attribuiti in base alla maschera dei permessi (**umask**). Questa maschera rappresenta i permessi che **non** vengono attribuiti. Di solito, il suo valore è **022** e con questo, non viene attribuito il permesso di scrittura (2) né al gruppo, né agli altri utenti.

E' evidente che la gestione dei permessi è affidata principalmente all'amministratore del sistema, il quale dispone di una serie di appositi comandi, ma anche i normali utenti (con qualche limitazione) possono effettuare delle modifiche.
Per modificare la modalità di accesso di un qualsiasi file di cui si sia proprietari si utilizza il comando

chmod (**ch**ange **mod**e). Questo strumento accetta due argomenti: una **operazione** che specifica i permessi da concedere/revocare, e i **nomi dei file** su cui lavorare. L'operazione si avvale dei seguenti tre operatori:

+ aggiunge permessi
- rimuove permessi
= fissa come unici permessi

`chmod go-w pippo`	*impedisce la scrittura al gruppo e agli altri*
`chmod a+rw pippo`	*autorizza tutti a leggere e scrivere*
`chmod a+x myscript`	*rende il file eseguibile da parte di tutti*
`chmod go=r pippo`	*rende il file a sola lettura per il gruppo e gli altri*
`chmod go= secret`	*rende il file privato, gruppo e altri senza permessi, nulla dopo l'operatore =*

Tips & Tricks - Autocompletamento e storico dei comandi

Autocompletamento

Una delle caratteristiche più belle della shell è l'autocompletamento. Quando occorre scrivere un comando o un nome di un file molto lunghi, si cominciano a scrivere le prime lettere e poi si preme **TAB** (una o 2 volte).
Se c'è solo un comando (o un file, dipende dal contesto) che inizia con quelle lettere, automaticamente il nome verrà completato.
Se invece ci sono più possibilità e quindi il completamento non è unico, verranno mostrate le varie possibilità, allora specificando qualche ulteriore lettera si toglierà l'ambiguità ed il comando verrà definitivamente completato.
Nota: Non si può ovviamente pretendere che la shell legga nel pensiero. Se si sta creando un nuovo file di nome *finchelabarcavalascialaandare* non si può sperare di scrivere *fin* premere TAB ed ottenerne l'autocompletamento!!

Storico

Lo storico dei comandi: usando le frecce in su e in giu si possono scorrere i comandi precedenti.
Mentre il comando **history** consente di avere una lista,numerata progressivamente, degli ultimi comandi effettuati. Digitando **!n** seguito da **invio** si richiede nuovamente il comando corrispondente.

```
history

!25                          ripete il comando n.25
```

Linux ce l'ha!

Passando ad un altro sistema operativo si è spesso presi da un senso di disorientamento causato dall'ansia di sapere se con il nuovo sistema sarà possibile fare le stesse cose che si facevano con il precedente.
La tabella seguente vuole solo raccogliere, in maniera sintetica, le corrispondenze tra alcuni noti prodotti Windows ed applicativi Open Source per Linux.
E' inoltre importante creare subito familiarità con i nomi dei nuovi applicativi, per evitare di smarrirsi nelle migliaia di applicazioni Open Source ormai facilmente disponibili tramite Internet.
Per un riferimento più completo si può consultare invece la voce del menù **Equivalenze Win/Lin**.

Funzione	**Windows**	**Linux**
Editor di testo elementare	NotePad	**KWrite**
Word Processor	Word	**AbiWord**
Web Editor	Crimson Editor	**Quanta Plus**
Lettore PDF	Acroread	**Acroread**
Browser per Internet	Internet Explorer	**Mozilla**
Gestore di posta	Outlook	**Evolution**
Downloader	Get Right	**D4X**
FTP manager	FTP Explorer	**gFTP**
Chat	ICQ	**LICQ**
Internet Relay Chat	mIRC	**KVirc**
Instant Messaging	ICQ, IRC,AIM, MSN,Yahoo	**Gaim**
Suite per ufficio	Office	**OpenOffice**
Foglio di calcolo	Excel	**GNumeric**
Contabilità familiare	Money	**GNUCash**
Grafica di base	Paint	**KolourPaint**
Grafica professionale	Photoshop	**Gimp**
Disegno vettoriale	Illustrator	**Sodipodi**
Browser di immagini	ACDSee	**GQView**
Sviluppo Flow Chart	Visio	**Dia**
Player Audio	WinAMP	**XMMS**
Player Video	Media Player	**MPlayer**
Masterizzazione	Nero	**K3B**
Masterizzazione	Nero	**XCDRoast**