Sistemi operativi

Appunti introduttivi ai sistemi operativi

Author: Andrea Manni

Version: 1.5

Indice degli argomenti

Generato con: http://docutils.sourceforge.net/rst.html

Sistema operativo

Concetti introduttivi

Un elaboratore e' una macchina in grado di seguire una serie di istruzioni al fine di permettere all'utente di raggiungere il risultato voluto. Possiamo partire da alcuni concetti di base per poterne studiare le caratteristiche.

Hardware:

Tutto quanto c'e' di *fisico* in un computer, quanto ha un peso (in senso materiale: grammi o kilogrammi) e occupa dello spazio. Il monitor, la tastiera, i supporti di storaggio dati, il computer stesso in ogni suo singolo componente interno e periferico sono l'**hardware** del computer, dall'inglese *ferraglia*.

Software:

Tutti i dati e i programmi (questi ultimi altro non sono che sequenze di istruzioni, quindi dati) che vengono elaborati o prodotti nel corso dell'esecuzione del programma. I software e i dati tipicamente risiedono su una memoria RAM volatile al momento dell'esecuzione, ma vengono archiviati sui *supporti di storaggio* (hard disk, CD-Rom, floppy...) in base alle necessita' del momento.

Input

I flussi di dati in ingresso inseriti generalmente dall'utente tramite le perifieriche dedicate a queste funzioni. Tastiera e mouse sono perifieriche di *solo Input*, una scheda di rete invece e' una periferica che puo' ricevere e trasmette dati. Questi dati sono utilizzati per modificare il corso dei programmi, nel caso ci siano piu' programmi utilizzati contemporaneamente per raggiungere il risultato voluto dall'operatore potra' accadere che il nostro programma riceva Input da un altro programma. Nei sistemi operativi *Unix si fa riferimento allo **standard Input** (stdin, la sua collocazione piu' elementare in un sistema basato sul kernel *Linux si basa sul device `/dev/input`).

Output

I flussi di dati in uscita, normalmente i risultati aspettati dall'operatore. Tipiche periferiche di output dedicate sono i monitor, le casse audio, le stampanti. Come nel caso dell'input possiamo considerare periferiche come le schede di rete come di input/output. Nei sistemi operativi *Unix si fa riferimento allo **standard output** (stdout, fisicamente puo' essere dirottato sul diverse periferiche a seconda della natura dei dati (pensiamo a un file audio: verra' mandato alla scheda audio e da questa alle casse. Sarebbe poco significativo se visualizzato a schermo, e poco gradevole il caso inverso: ad es. telefonare a qualcuno e sentirsi rispondere da un FAX.) ma normalmente ci aspettiamo di vedere i risultati *a schermo*: sul nostro terminale `/dev/tty*` o equivalente.).

Errors

Non sempre le nostre aspettative nei confronti dell'elebarotore vengono soddisfatte, talvolta il nostro programma genera errori e un risultato solo parziale. Nei sistemi *Unix questo tipo di *output* viene

indicato come **standard error** (stderr) e puo' essere reindirizzato su una canale diverso dal normale output. Ad esempio suonando una serie di brani musicali l'output viene indirizzato fino alle casse del computer, ma se un brano risulta illeggibile viene visualizzato un errore a *schermo*. Situazione simile nel caso di una stampa: l'output va' alla stampante me se durante la stampa si generano messaggi di errore questi non compariranno sulla carta.

Questa possibilita' di canalizzare lo stdout e lo stderr verra' utile in seguito quando si avra' l'esigenza di pianificare operazioni svolte in automatico (es. script di back-up) durante le quali l'operatore non e' disponibile per leggere gli errori, che potranno essere re-indirizzati su un file di log o altro.

Interfaccia utente

Il sistema formato da periferiche (es tastiera e monitor) che permette lo scambio di informazioni tra l'utente e l'elaboratore. Tutti i sistemi multifunzione (cioe' che possono svolgere piu' di un singolo compito) sono dotati di una interfaccia utente. I sistemi *multi-pourpose* hanno interfacce utente piu' sofisticate rispetto alle macchine dedicate: ad es. un personal computer ha un'interfaccia utente piu' completa rispetto ad una console per videogiochi o a un cellulare. Le interfacce utente possono essere di diverso tipo:

Riga di comando

CLI (Common Line Interface). Interfaccia testuale: input e output sono solo o prevalentemente caratteri, quindi niente grafica, finestre, icone. E' ad esempio l'interfaccia che ci si trova davanti a un terminale *Unix (CTR + ALT + F1 sotto Gnu/Linux) oppure stabilendo una sessione remota con SSH o telnet.

Interfaccia grafica

Permette la visualizzazione di icone finestre e quant'altro, grazie ad un server grafico che trasforma le informazioni piu' semplici in una rappresentazione grafica. Generalmente piu' *pesante* dal punto di visto delle risorse di sistema del computer, permette di eseguire solo le operazioni espressamente previste e rese disponibili dai vari *menu*, a differenza della CLI che puo' permettere la combinazioni di singoli comandi (pipes) e la creazione di script per automatizzare operazioni anche di relativa complessita'.

Generalmente si accompagna a un sistema di puntamento come un mouse, touch pad, tavoletta grafica. track ball...

La superiore complessita' la rende sconsigliabile per le sessioni remote, per lo meno quando e' possibile svolgere le stesse operazione tramite un'interfaccia CLI. Questo per non sottrarre risorse ai software di nostro interesse e per motivi di sicurezza (piu' una cosa e' complessa e piu' e' soggetta a problemi di sicurezza).

Touch screen

Schermi sensibili al tocco, come su palmari, cellulari IPhone o i *tablet PC*. Le caratteristiche sono analoghe a una interfaccia grafica, cambia solo il sistema di puntamento (non avete un mouse o un touch pad ma si tocca direttamente lo schermo con una penna o le dita).

Caratteristiche di un Sistema Operativo

Alcune delle caratteristiche dei sistemi operativi ci aiuteranno a capire la funzione prevalente di questo in rapporto con gli altri programmi in esecuzione sul computer.

Multitasking

Capacita' di un OS di eseguire piu' programmi contemporaneamente.

Un personal computer generalmente ha in esecuzione centinaia di singoli programmi (o per la precisione: *processi*), oltre a quelli direttamente lanciati dall'utente ce ne saranno altri dedicati alla gestione dell'hardware o al mantenimento del sistema operativo.

Una cellulare (si pensi ai vecchi modelli) o una piccola calcolatrice invece possono eseguire solo un compito (task) alla volta.

Tipicamente risiede nel kernel la capacita' del sistema di poter eseguire con stabilita' piu' processi, utilizzando uno scheduler.

Multiutenza

> I sistemi multiutenti possono avere piu' utenti attivi contemporaneamente (ma anche uno alla volta a seconda della disponibilita' di input).

Il sistema e' comunque in grado di distinguere tra gli utenti: ad es. mia sorella non ha la possibilita' di eliminare le MIE foto, e viceversa. La multiutenza sotto il profilo tecnico si appoggia su un software di autenticazione / log-in per distinguere gli utenti, oltre a funzionalita' delegate al file system per limitare accessi ed esecuzione dei file ai diversi utenti di sistema. Ad esempio il filesystem FAT32 dei vecchi sistemi Windows (e usato sulla maggior parte delle chiavette USB e memory card varie) non permette la gestione delle propieta' dei files.

La presenza di piu' utenti in genere prevede una gerarchia tra questi (ad esempio *user, power user, administrator* sotto sistemi Windows o l'utente *root* per sistemi Unix). In genere si ha un solo utente abilitato alle modifiche delle funzionalita' del sistema (l'amministratore, che puo' abilitare le periferiche o installare nuovo software nel sistema) e semplici *utenti di sistema* (`system users`) che possono solo usufruire dei programmi messi a loro disposizione. Questa soluzione, oltre che a garantire la presenza di un account in grado di rimediare in caso di emergenza a eventuali errori fatti da altri, permette di avere un ambiente di lavoro piu' sicuro dato che una volta loggati come utenti di sistema non si ha la possibilita' di causare danni gravi all'intero sistema per una semplice distrazione. *Mai lavorare come amministratori quando non e' assolutamente necessario.*

Sistema operativo

Possiamo ora cercare di dare una definizione di sistema operativo:

> Il Sistema Operativo (OS: Operative System) e' quell'insieme di software che servono per far funzionare il sistema *in generale* piu' che svolgere un compito particolare come modificare un file o visualizzare un video. E' l'*ambiente* in cui potranno convivere ed essere utilizzati tutti i software applicativi usati dall'utente.

Sara' il sistema operativo a identificare gli utenti al momento di cominciare una sessione di lavoro, e garantisce gli accessi ai files in base a queste credenziali (e alle caratteristiche del file system, cosa che approfondiremo successivamente).

L'OS si pone poi come tramite tra i singoli applicativi e le risorse di sistema. Ad esempio se sono disponibili diversi programmi in grado di produrre stampe e una stampante, sara' il sistema operativo a gestire le code di stampa in modo che non si intralcino (a dire il vero l'esempio potrebbe non essere tecnicamente esatto. Ma rende l'idea.) I singoli programmi non utilizzeranno direttamente la stampante, ma semplicemente si *interfacceranno* all'OS al momento della stampa.

Quando installiamo un nuovo editor di testo non ci preoccupiamo di fornirgli un *driver* per la stampa. Sappiamo che questo e' gia' disponibile al sistema operativo, e i singoli applicativi si appoggeranno a questo. Questo "*layer di astrazione*" sull'utilizzo delle periferiche (al quale partecipa anche il kernel) semplifica di molto la realizzazione e l'installazione dei singoli software, garantendo maggiore stabilita' all'intero *sistema*.

In modo simile tutti gli applicativi che utilizziamo si *appoggiano* sul sistema operativo, tanto che siamo abituati ad avere *versioni diverse* degli stessi software rilasciate per i *diversi sistemi* (c'e' una versione di Openoffice.org per sistemi Microsoft, GnuLinux, Apple e cosi' via). Ovviamente la versione per il sistema X non funzionerebbe sul sistema Y.

I sistemi operativi, come del resto i singoli applicativi, sono rilasciati (quando possibile) in versioni a 32 o 64 bit, oppure per architetture diverse (x86, PPC, Arm, RISC...). In genere si indica la possibilita' di un OS di *girare* su architetture hardware diverse col termine *portabilita'*. Una maggiore portabilita', oltre che per l'intrinseco vantaggio, e' spesso indice di maggiore stabilita' in quanto il test del sistema in ambienti diversi permette di evidenziare *bug* difficilmente riscontrabili altrimenti.

I vari Sistemi Operativi

Per ulteriori informazioni sulle caratteristiche di alcuni dei sistemi operativi attualemnte in uso si seguano i seguenti link:

MS Windows

- http://it.wikipedia.org/wiki/Windows

Unix

http://it.wikipedia.org/wiki/Unix

Gnu/Linux

- http://it.wikipedia.org/wiki/Gnu/Linux

BSD

- http://it.wikipedia.org/wiki/Berkeley_Software_Distribution
- http://it.wikipedia.org/wiki/FreeBSD
- http://it.wikipedia.org/wiki/Openbsd

Apple Mac OS

- http://it.wikipedia.org/wiki/Mac_OS
- http://it.wikipedia.org/wiki/Mac_OS_X

Kernel

> In informatica, il **kernel** Costituisce il nucleo di un sistema operativo. Si tratta di un software avente il compito di fornire ai processi in esecuzione sull'elaboratore un accesso sicuro e controllato all'hardware. Dato che possono esserne eseguiti simultaneamente più di uno, il kernel ha anche la responsabilita'à di assegnare una porzione di tempo-macchina e di accesso all'hardware a ciascun programma (multitasking).

Tra tutti i processi in esecuzione su un computer uno ha un'importanza speciale: il kernel.

Il kernel ha il compito di far funzionare l'hardware (tastiera, monitor, RAM, CPU) e di attribuire le risorse di sistema agli altri processi, ne consegue che un malfunzionamento nel kernel (*kernel panic*) puo' compromettere qualunque altro processo in esecuzione. Considerando che il kernel gestisce anche le periferiche di input / output, nel caso di blocco del kernel l'intera macchina potrebbe diventare irraggiungibile. In questo caso l'unita soluzione sarebbe un *rese* hardware della scheda madre, con conseguente perdita di tutti i dati in 3 e possibile corruzione dei filesystem.

Il kernel gestisce l'attribuzione della memoria RAM tra i vari processi in esecuzione: tramite la protezione della memoria si evita che un applicativo andato fuori controllo possa trasbordare nello spazio RAM di un altro, compromettendo anche questo e magari l'intero sistema. La protezione della memoria e' una funzione necessaria per garantire l'efficenza del multitasking.

Debian utilizza un Kernel Linux. Windows attualmente usa un kernel di derivazione NT. Mac OSX utilizza un kernel "Darwin" su un sistema operativo derivato da BSD (Unix). Il sistema operativo e' composto da una serie di processi che girano sopra il kernel. I software applicativi a loro volta girano sul sistema operativo.

Il kernel e' sostituibile e aggiornabile rispetto al sistema operativo.

Gnu/Linux	OS
Linux	kernel

Microsoft Windows XP	OS
NT	kernel di MS Windows XP

Le distribuzioni

Differenze principali tra le varie distribuzioni:

- utility di installazione
- gestione dei pacchetti: es dpkg e RPM (portage e altri)
- numero e qualita' dei pacchetti
- licenze
- target di utilizzo
- tempi di rilascio

- http://it.wikipedia.org/wiki/Distribuzione_Linux
- http://it.wikipedia.org/wiki/Categoria:Distribuzioni_Linux

Memoria virtuale

O memoria di swap. Si utilizza quando il sistema esaurisce la RAM fisicamente disponibile. Si utilizza un altro supporto di storaggio per sopperire alla mancanza di RAM. Tipicamente viene usato l'hard disk, che e' piu' lento della RAM.

Con il termine swap si intende, in informatica, l'estensione della capacita'à della memoria volatile complessiva del computer, oltre il limite imposto dalla quantita'à di RAM installata, attraverso l'utilizzo di uno spazio su un altro supporto fisico, ad esempio il disco fisso.

A seconda del sistema operativo utilizzato è possibile avere file di swap (chiamato anche 'Memoria virtuale'), residenti nel normale file system del sistema, oppure partizioni di swap, cioe'è sezioni di disco integralmente dedicate allo swap ed inizializzate con un proprio specifico tipo di file system. L'uso della partizione è generalmente migliore dal punto vista delle prestazioni, perché evita che lo swap vada soggetto alla tipica di alcuni file system. Per contro, occupa una delle (poche) partizioni disponibili sul disco fisso.

In pratica, quando la memoria RAM libera non è più sufficiente per contenere tutte le informazioni che servono ai programmi, il sistema operativo si fa carico di spostare una certa quantita'à di dati (quelli meno recentemente utilizzati) dalla memoria volatile a quella di massa, liberando quindi una parte della RAM per permettere il corretto funzionamento dei programmi. È chiaro che nel momento in cui si rende necessaria tale operazione, le prestazioni del sistema crollano bruscamente, essendo la scrittura su hard disk molto più lenta di quella in RAM (oltre cento volte, al 2006).

Swap space per Linux

Sui sistemi basati sul kernel di Linux viene creata una partizione con un file-system dedicato per la memoria virtuale: una o piu' cosidette *partizioni di swap*.

Per le normali work-station si segue la regola di creare una partizione di swap pari al doppio della memoria RAM, ma non superiore a `512MB` (questo per via della buona ottimizzazione del gestore della memoria di Linux: raramente nell'uso normale si arriva ad aver bisogno di piu' di `512MB` di swap.

Caso particolare sono i portatili: vista la lentezza degli Hard Disk dei portatili e il maggior consumo elettrico causato dall'utilizzo di questi supporti per lo swap si tende ad avere un po' piu' di RAM (`512MB - 1GB`) ed evitare completamente lo spazio di swap. Ma attenzione: se si vuole utilizzare il *Suspend to disk* (S4) servira' una partizione di swap pari a circa il doppio della RAM.

Devices a blocchi

I device a blocchi sono suddividisibili in blocchi (unita) ed e' possibile scrivere e/o leggere su questi (input/output) Ad esempio gli hard disk o i supporti di storaggio in generale sono visti dal kernel come block device. Tipicamente per poterli utilizzare al fine dello storaggio dati sara' necessario creare un filesystem su di essi.

Devices a caratteri (seriali)

I device seriali sono continui, e non prevedono il "ritorno" (si pensi a una stampante, o all'output delle casse, a un terminale seriale) [1].

Dati

Dato

Il termine dati e' fondamentale nell'informatica, essendo i dati l'oggetto e il risultato di tutto il nostro lavoro.

Potremmo definire un dato come un'informazione elementare, significativa ed autonoma.

Si prenda ad esempio un sistema realizzato per gestire elettronicamente la contabilita': il guadagno del mese di Gennaio 2009 potrebbe essere `1200`. Oppure l'indirizzo di un contatto `Via Mazzini 8`.

Questi *valori* possono essere l'oggetto di elaborazioni, o il risultato di procedure che hanno manipolato altri dati.

Metadato

I metadati sono informazioni che qualificano i dati. Essi non hanno un'esistenza e un valore autonomo, sono funzionali ai dati che descrivono e senza di essi non avrebbero valore.

Tornando al precedente esempio del software per la contabilita', `Gennaio 2009` potrebbe essere un metadato, in quanto la sua funzione e' di descrivere il periodo a cui fa riferimento il dato `1200`. Senza quest'ultimo non ci interesserebbe sprecare risorse di storaggio per conservare `Gennaio 2009`.

Dato	***Metadato***
1200	Gennaio 2009

Spesso i metadati vengono utilizzati per stabilire le relazioni tra i vari dati, in modo da poterli caratterizzare e distinguere tra loro. Si pensi a un supporto di storaggio in cui vengano mantenuti terabytes di dati: senza la possibilita' di distinguerli l'uno dagli altri diventerebbero inutili (o meglio inutilizzabili), maggiore e' il numero dei dati tanto piu' si avverte la necessita' di caratterizzarli in modo da poterli *gestire* piu' facilmente e *incrociarli* tra loro in modo da ottenere le informazioni che veramente ci interessano.

Si tenga conto che buona parte dell'informatica consiste nello storaggio di *enormi* quantitativi di dati (che bisognera' essere in grado di distinguere e recuperare), e la funzione fondamentale degli elaboratori e' la capacita' di confrontare rapidamente questi dati tra loro. Quindi con l'aumentare del numero dei dati i metadati diventano sempre piu' importanti e *significativi*. Si pensi alla natura dei database.

I filesystem

File

Un file e' un insieme di dati con un inizio e una fine identificabile in mezzo ad altri.

Quindi avremo:

- un identificatore, come un `nome_file` accoppiato al suo *percorso* sul file system (possono esserci molti file con lo stesso nome ma non possono esserci due file on lo stesso nome nella stessa cartella).

• Un qualche *sistema* per recuperare quel file in mezzo ad altri. Ad esempio potremmo pensare, in un sistema molto rudimentale, a una coppia di coordinate che ci permettano di identificare il punto di inizio e il termine su un block device.

Filestem elementare		
Identificatore	***inizio***	***fine***
pippo	12	21

Filesystem

Il file system e' una struttura logica che ci permette di individuare i diversi file. Sostanzialmente e' un'insieme di metadati che caratterizzano i singoli file piu' quanto necessario per poterli gestire.

Alcuni esempi: e' il filesytem a poter caratterizzare i file con propietari, gruppi, permessi (lettura, scrittura, esecuzione), data di creazione e cosi' via. Alcuni filesytem permettono certe funzionalita', altri (sopratutto quelli piu' datati) sono piu' *rudimentali*.

Tipicamente i filesystem sono strutturati secondo un modello gerarchico basato su *files e cartelle*: quindi per poter identificare un file ci serve il suo *nome proprio* piu' il suo *percorso* nella struttura del filesystem. Il progressivo aumento del numero dei files sta cominciando a far sentire i limiti di questo modello.

Con l'aumentare delle informazioni tendono ad essere piu' efficaci modelli relazionali, che permettono di interagire coi file in modo simile a un database. Ad esempio pensiamo alle librerie che contengono centinaia di migliaia di brani musicali MP3, oppure migliaia di fotografie: cercare di gestirli tramite nome del file e cartelle diventa poco pratico. Meglio raggrupparli per autore, data, album o quant'altro si adatti alla loro natura e al loro utilizzo.

Tabella di allocazione files

non solo su device fisici: esempio NFS

File system distribuito: citati il NBD e ATA over ethernet

Gestione dei Pacchetti

Un Sistema di gestione dei pacchetti e' una collezione di strumenti che automatizzano il processo di installazione, aggiornamento, configurazione e rimozione dei pacchetti software di un computer.

Tali strumenti sono diffusi tipicamenti sui sistemi basati su software libero, in quanto i gestori della distribuzione (cosidetti *mantainers*) avendo accesso al codice sorgente del software hanno la possibilita' di prepararne una versione ottimizzata e adattata all'infrastruttura del sistema operativo (si pensi a librerie condivise, API, standard per la collocazione dei files di importanza particolare).

Cerchiamo di capire meglio il processo: L'autore / sviluppatore di un software (ad esempio un browser web come Mozilla Firefox) rilascia il suo lavoro con una licenza libera, che prevede la possibilita' di ottenere il codice sorgente, modificarlo e redistribuirlo. A questo punto un **mantainer** della distribuzione, che non deve per forza essere l'autore del software originale, scarica il sorgente, lo modifica in modo che si integri con gli altri software del sistema operativo, preoccupandosi che i file di configurazione, temporanei, ecc. vengano collocati sul file-system in modo analogo agli altri software del sistema operativo, in modo da ottenere un sistema il piu' possibile omogeneo e *strutturato*. Il mantainer quindi non introduce nuove funzionalita' nel software (per lo meno non nella distribuzione di cui si occupa, nel caso provvederebbe a fornire le modifiche all'autore in modo che sia lui stesse ad integrarle nella versione *ufficiale*, cosi' che poi anche chi non usa quella determinata distribuzione possa avantaggiarsene): si limita ad *adattarlo* perche possa ben funzionare nella distribuzione.

Quindi possiamo dire che un software come ad es. Apache o Firefox sono funzionalmente identici in ogni distribuzione, cambia solo il loro livello di integrazione nel sistema operativo. Come gia' detto le distribuzioni si caratterizzano per la *qualita'* di questa integrazione, il numero di pacchetti che riescono a gestire, l'efficacia dei software di installazione / aggiornamento / rimozione.

Non ultima per importanza: l'attenzione alle problematiche legate alla sicurezza. Infatti e' bene tener presente che sara' il mantainer a dover provvedere tempestivamente all'aggiornamento del pacchetto in caso di problemi di sicurezza, dato che l'utente finale non interagisce direttamente con l'autore del

software. Il che puo' porre il seguente problema: quando l'autore rileva un problema tende spesso a risolverlo con una versione aggiornata del software (quindi si passa ad es dalla versione 1.0 alla 1.1 , e gia' che si rilascia magari si coglie l'occasione per introdurre nuove funzionalita'). Ma una distribuzione per essere definita *stabile* **deve** rimanere costante, altrimenti diventa impossibile controllare possibili conflitti tra i diversi software. In questo caso, le i mantainers delle distribuzioni dedicate al *lato server* o comunque con particolare vocazione alla sicurezza, si attivano per *portare* loro stessi i soli aggiornamente della sicurezza alla versione precedente del software in modo che eventuali nuove funzionalita' non possano rischiare di compromettere l'interazione del software con altri presenti sul sistema.

Distribuzioni come *Debian* lavorano in questo modo, altre preferiscono un approccio piu' *rilassato*: tipicamente perche mettono a disposizione dell'utente software aggiornato piu' frequentemente, atteggiamento molto gradito sulle *work station* degli utenti che possono cosi' avvantaggiarsi piu' rapidamente di nuove funzionalita'. Se il software viene aggiornato *molto rapidamente* l'utente fianale non puo' aspettarsi che l'integrazione tra i vari pacchetti sia stata testata per lunghi periodi!

Cio' non toglie che gli utenti di Debian possano decidere di utilizzare le release *Sid/unstable* o *testing* che vengono aggiornate quotidianamente. Ma sulle macchine che devono restare stabili senza creare problemi quotidianamente si puo' contare sulla versione *stable*, che garantisce due o tre anni di *ragionevole quiete senza sorprese*. Dal momento in cui si raggiunge la stabilita' del sistema con le funzionalita' richieste, in ambiente *aziendale* non si avverte la necessita' di avere sempre a disposizione l'ultima release disponibile di un software col rischio che si possano verificare problemi. Un software appena rilasciato non puo' vantare stabilita': questa infatti non e' un concetto *assoluto*: la stabilita' si ottiene facendo girare sempre la stessa versione del software sul piu' ampio bacino di utenza in modo da poter individuare e risolvere gli eventuali problemi che mano a mano si individuano. La stabilita', per quanto riguarda il software *consumer*, e' relativa (allo stesso modo la *sicurezza* non e' uno *status* ma piuttosto un *processo*: i problemi si possono sempre verificare, tutto sta nella capacita' e tempi di reazione per risolverli con aggiornamenti).

Ci sono 3 principali gestori di pacchetti in generale:

Gestori di pacchetti		
Nome	***Distro***	***estensione***
DPKG	Debian	.deb
RPM	Red Hat	.rpm
Portage	Unix	---

Lo stesso software viene pacchettizzato in *formati* diversi (deb,rpm,...)

In alternativa c'e' generalmente una versione binaria generica del software, che viene distribuita con un archivio (targz). Questo non usufruisce delle agevolazioni e integrazioni offerte dal gestore dei pacchetti della distro.

In alternativa per tutto il software libero e' sempre disponibile il codice sorgente da compilare in un binario eseguibile. Permette l'ottimizzazione e la customizzazione del software. Portage si basa su questo sistema: si scaricano i sorgenti modificati e poi si compilano. Si faccia attenzione all'ottimizzazione: dipende da chi la fa...

Per la gestione dei pacchetti su distribuzioni Debian o derivate (quindi anche Ubuntu) si legga: l'Apt Howto = http://www.debian.org/doc/manuals/apt-howto/index.it.html

DPKG
: Il gestore dei pacchetti. Il software che si occupa dell'istallazione, rimozione e risoluzione dei conflitti e dipendenze. Apt e' il *front-end* di DPKG.

Elementi:

Dipendenze

Talvolta un pacchetto per poter funzionare necessita di un altro pacchetto. Si deve quindi installare anche l'altro oltre a quello richiesto.

Il software di installazione si preoccupa quindi dell'ordine in cui installare i vari pacchetti (dato che il pacchetto dovrebbe essere un'entita' indipendente / modulare rispetto agli altri pacchetti) e la configurazione di tutti e del sistema. Per i pacchetti rimossi vengono tolte anche le dipendenze quando queste non compromettano altri software, in caso di incomprensioni si puo' usare il comando `apt-get autoremove` (su sistemi dotati di DPKG).

Si tenga presente che Debian, di default, quando rimuove un pacchetto non elimina i file di configurazione dello stesso, nel caso si deve usare l'opzione `--purge`. In questo modo se l'utente rimuove un particolare software per provare un alternativa o liberare dello spazio temporaneamente, al momento opportuno re-installando lo stesso software questi si comportera' esattamente come quando era presente sul sistema. E' doverso far notare che in caso di problemi di configurazione di un programma l'idea di rimuoverlo per poi re-installarlo sia poco seria: i problemi vanno tracciati e risolti, non evitati. In caso contrario ripeteranno. Piuttosto si faccia sempre un una copia di *back-up* del file di configurazione **prima** di cominciare a modificarlo, in modo che in caso non si riesca ad arrivare ad arrivare nei tempi richiesti ad una nuova configurazione funzionale si possa per lo meno tornare alla precedente.

Stessa cosa per la rimozione per dpkg, ma Portage non permette la rimozione di software.

Conflitti

Alcuni software non possono essere presenti sul sistema contemporaneamente ad altri. Si pensi a un server di posta (o a qualunque servizio che ascolta su una porta determinata), o a versioni incompatibili dello *stesso* software. In questo caso il gestore di pacchetti avvertira' l'utente che al momento dell'installazione del software richiesto verranno rimossi i pacchetti in conflitto.

---

1 Questa definizione di device a *caratteri* non e' esatta, ma allo stato attuale delle nostre conoscenze e necessita' risulta piu' utile caratterizzare i due tipi di device dal punto di vista funzionale. Una definizione piu' *tecnica* di questo particolare tipo di device dovrebbe citare il fatto che questi in genere non supportano buffers e non permettono di accedere ai dati in modalita' random_access. Oltre al fatto che trattano singoli caratteri per volta invece che per *blocchi* di dati.