

# Il sistema GNU/Linux

## Riflessioni sul sistema *GNU/Linux*

di

Rodolfo Giometti <giometti@linux.it>

(f|ht)tp://ftp.enneenne.com/docs/talks/gnu\_linux.p(s|df)

# Il legame Linux ↔ UNIX

## Linux è un sistema stile UNIX

Unix è un progetto del 1969 e nasce per gioco dall'esperienza di MULTICS.

Il sistema **Unics** (UNlprogrammed Computer System) è stato sviluppato nei laboratori AT&T ed è stato distribuito gratuitamente alle università (dal 1973).

UNIX è scritto in **C**, un linguaggio scritto appositamente per la programmazione di sistema, «C» è il successore di «B», che è una semplificazione di «BCPL».

La prima versione di UNIX era estremamente leggera, era stato sviluppato inizialmente su di un sistema con 16Kb di memoria.

In ambiente UNIX sono stati sviluppati diversi progetti:

- L'editor di testi **vi** è stato sviluppato nel 1977 sotto UNIX.
- All'inizio degli anni '80 **TCP/IP** è stato sviluppato sotto UNIX.

# L'astrazione «file»

L'astrazione file è un concetto molto potente che permette di ottenere potenti funzionalità

Tutto all'interno del sistema è visto come se fosse un file: i dischi, la tastiera, il video, ecc..

Grazie a questo tutti i programmi possono venir scritti senza preoccuparsi di «dove» questi dovranno operare.

Con uno stesso programma (ad esempio *cat*) è possibile leggere: un file (classico), i dati che provengono da un ADC, una istantanea scattata con un frame grabber o il contenuto di un floppy in modalità RAW, ecc..

Anche l'ingresso e l'uscita di un programma (che prendono i nomi di *stdin*, *stdout* e *stderr*) è vista come un file e quindi più programmi possono essere collegati in «cascata» per ottenere funzionalità più potenti di quelle ottenibili con i singoli programmi.

# I «filtri»

Sono programmi che fanno  
solo una cosa e la fanno  
bene!!!

Collegando in cascata più filtri si possono ottenere funzionalità molto complesse.

Il comando:

```
giometti@zaigor:~$ cat /dev/fgrab | yuv2jpg  
1024 768 | gzip -9 | uuencode mypic.jpg |  
mail -s "See my pic!" mom@home.org
```

scatta una foto e la invia per posta, mentre:

```
giometti@zaigor:~$ cat book.pdf | pdftops |  
psnup -4 | lpr -P rlp
```

legge un testo in formato PDF, lo reimpagina compattando 4 pagine in una e stampa il tutto via rete sul server di stampa.

# Il «filesystem»

## Tutto nei sistemi UNIX si appoggia al filesystem

Il filesystem contiene tutti i file del sistema e quindi anche i riferimenti alle periferiche.

Esistono diversi tipi di «filesystem» all'interno dei sistemi UNIX. Linux, ad esempio, possiede i filesystem:

- **ext2/ext3** propri di Linux;
- **FAT/VFAT** proprio dei sistemi MS;
- **NFS** o Network File System per la condivisione in rete tra sistemi UNIX;
- **SAMBA (smbfs)** per la condivisione in rete tra reti miste UNIX/MS;
- **procfs**, un filesystem virtuale per la «gestione» del sistema;
- ecc..

# I «processi»

I processi sono entità «dinamiche» al contrario di programmi che sono «statici»

Nei sistemi UNIX sopravvivono entità che evolvono nel tempo e che eseguono compiti (programmi) specifici: i processi.

Ogni processo è individuato univocamente all'interno del sistema grazie al suo **PID** (Process IDentificator).

Un processo può crearne altri per un meccanismo molto simile alla «clonazione»; in gergo i processi che si «clonano» si chiamano «padre» (o *parent*), mentre i «clonati» si chiamano «figlio» (o *child*).

Il primo processo che il nucleo esegue al boot è *init* che è anche il padre, diretto o indiretto, di tutti gli altri processi presenti nel sistema.

# Le «chiamate di sistema»

## I processi «comunicano» con il nucleo usando queste speciali funzioni

I driver che pilotano le periferiche eseguono nel nucleo e per comunicare con loro bisogna utilizzare le *syscall*.

Le «chiamate di sistema» non vanno confuse con le «funzioni» che eseguono solo nello spazio utente.

La suddivisione tra «spazio di nucleo» (*kernel space*) e «spazio utente» (*user space*) è alla base della teoria dei sistemi operativi e serve per implementare una serie di politiche di «supervisione» dell'attività dei processi: permessi di accesso ai file/periferiche, protezione di memoria, ecc..

Ogni volta che un processo deve utilizzare una risorsa di sistema lo fa utilizzando quindi una chiamata di sistema la quale verifica la validità della richiesta e in caso positivo la esegue.

# I «meccanismi» e le «politiche»

Le politiche di gestione di una risorsa si implementano con diversi meccanismi di accesso

Distinguere tra «meccanismi di accesso» alle risorse e le loro «politiche di utilizzo» permette di risolvere molti problemi in maniera elegante ed efficace.

Se implementiamo ad esempio un driver per uno speciale dispositivo in modo tale che questo sia «visto» dai processi come se fosse un file allora per accedervi basta utilizzare i normali tool di manipolazione dei file.

Se il driver di un hard-disk mi «virtualizza» il dispositivo come se fosse un vettore di caratteri e mi fornisce semplicemente le funzioni per leggere e scrivere tali caratteri allora è possibile poi implementare su tale dispositivo qualsiasi tipo di filesystem; cosa non possibile (o comunque molto difficile) se il driver implementasse già un filesystem ad esempio di tipo FAT.

# I «socket»

Sono dei canali di comunicazione che possono essere utilizzati per il *networking*

Su di un socket spesso si utilizza un «protocollo di comunicazione»: i più utilizzati sono il TCP/IP e l'UDP/IP.

I protocolli basati sul protocollo **IP** implementano il concetto di «porta» attraverso la quale si possono specificare diversi «servizi» che una macchina denominata «server» supporta. Ad esempio: il servizio WWW (**HTTP**) è sulla porta 80, l'**FTP** è sulle 20/21, il **POP3** è sulla 110 e l'**SMTP** è sulla 25.

Se i protocolli sono di tipo «testo» allora i client che li implementano possono essere semplicissimi!

```
giometti@zaigor:~$ telnet www.unsito.it 25
```

```
giometti@zaigor:~$ telnet www.unsito.it 110
```

# I «socket» come file

## Anche questi canali possono essere acceduti dei file

Utilizzando l'astrazione file anche con i socket si possono ottenere delle funzionalità molto potenti con poco sforzo.

Se utilizziamo una coppia di programmi che da una parte prendono lo *stdin* di un processo e lo mandano in **broadcast** su UDP in rete (Tx), e dall'altra leggono i pacchetti UDP e li stampano sul proprio *stdout* (Rx), in tutto  $\pm 30$  linee di codice C per programma, allora sul Tx con il comando:

```
giometti@zaigor:~$ telnet localhost | tee /dev/ttyq9 | udp-send 12345
```

e su gli Rx con i comandi:

```
veronica@mazinga:~$ udp-get 12345
```

```
rossi@goldrake:~$ udp-get 12345
```

```
lombardo@jeeg:~$ udp-get 12345
```

Si implementa una remotizzazione su più piattaforme di una console di testo!!

# Un esempio «classico»: `inetd`

Questa applicazione viene  
spesso chiamata  
«superserver»

Grazie a questa applicazione scrivere  
un server diventa semplicissimo.

Ad esempio un semplice server per eseguire remotamente  
dei comandi può essere definito dal seguente script:

```
#!/bin/bash
while true; do
    read cmd
    su nobody -c $cmd
done
```

Il superserver infatti, una volta configurato, si mette in attesa  
di una connessione su una porta e quindi «collega» lo *stdin*  
e lo *stdout* del programma che implementa il codice del  
processo server sul canale socket utilizzato per la  
comunicazione con il processo client.

# L'ambiente grafico: X-Window

Anche il sistema X-Window si basa sui modelli client/server e meccanismi/politiche

Abbiamo l'«X-server» e gli «X-client» e i «Window Manager» e le «Xlib».

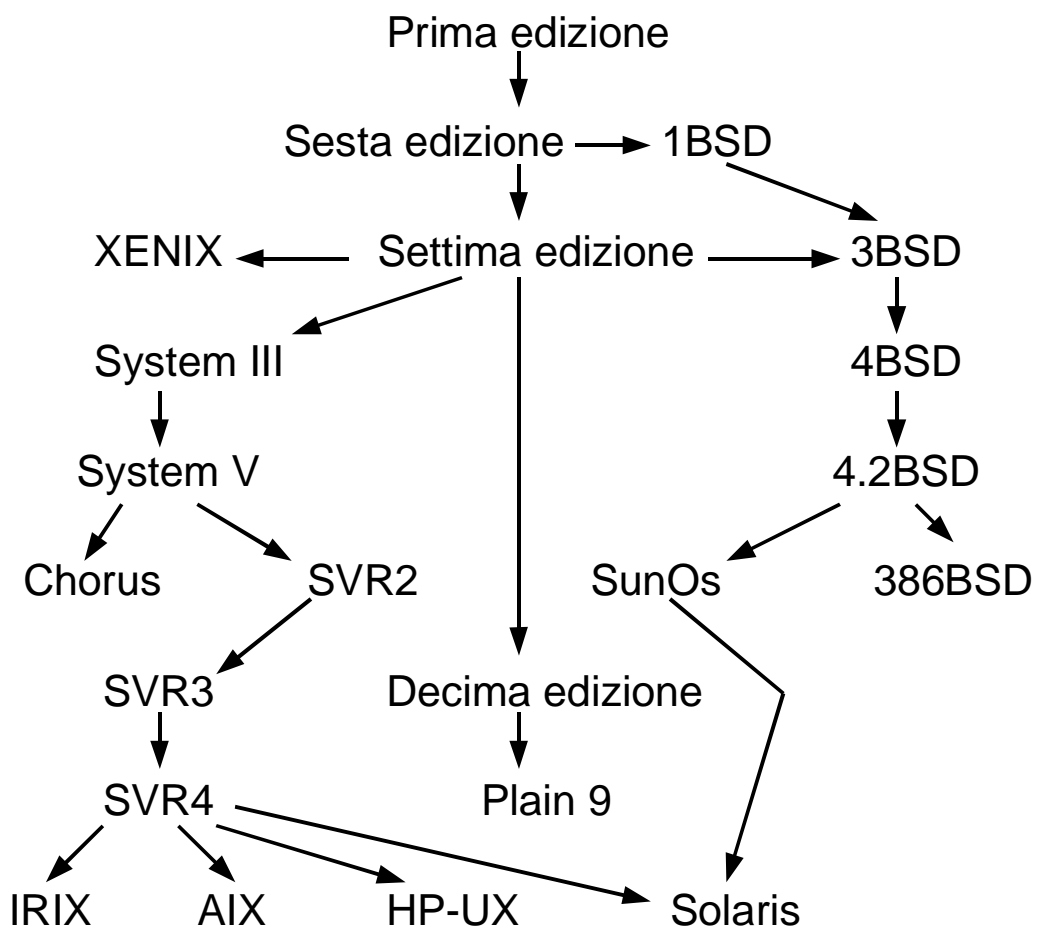
L'**X-server** si occupa della gestione dell'hardware grafico ed esegue i comandi che gli **X-client** gli inviano. Esiste in genere un X-server diverso a seconda dell'hardware grafico utilizzato.

Il **Window Manager** gestisce le finestre e la loro interazione con l'utente senza nessuna interazione con l'hardware grafico mentre le **Xlib** implementano i comandi di gestione dell'hardware che servono per «muovere» gli oggetti grafici.

In questo modo è possibile eseguire uno stesso client (applicazione grafica) su diverse piattaforme, anche remote, senza doversi preoccupare di gestire direttamente le diverse piattaforme grafiche.

# UNIX e le sue «evoluzioni»

La distribuzione in forma sorgente ha permesso la ramificazione e la «proprietarizzazione» ne ha poi impedito la riunificazione



# «GNU is Not UNIX»



## Il progetto GNU nasce per ritornare «alle origini»

Non chiamavamo il nostro software «software libero», poiché questa espressione ancora non esisteva, ma si trattava proprio di questo.

La situazione cambiò drasticamente all'inizio degli anni '80 [...] I moderni elaboratori di quell'epoca, [...] avevano il proprio sistema operativo, ma nessuno di questi era libero: si doveva firmare un accordo di non-diffusione persino per ottenerne una copia eseguibile.

Questo significava che il primo passo per usare un computer era promettere di negare aiuto al proprio vicino. Una comunità cooperante era vietata. La regola creata dai produttori di software proprietario era: «se condividi il software col tuo vicino sei un pirata. Se vuoi modifiche, pregaci di farle».

L'idea che la concezione sociale di software proprietario, cioè il sistema che impone che il software non possa essere condiviso o modificato, sia antisociale, contraria all'etica, semplicemente sbagliata, può apparire sorprendente a qualche lettore. Ma che altro possiamo dire di un sistema che si basa sul dividere utenti e lasciarli senza aiuto?

Allora cercai un modo in cui un programmatore potesse fare qualcosa di buono. Mi chiesi dunque: c'erano un programma o dei programmi che io potessi scrivere, per rendere nuovamente possibile l'esistenza di una comunità?

La risposta era semplice: innanzitutto serviva un sistema operativo. [...] Essendo un programmatore di sistemi, possedevo le competenze adeguate per questo lavoro. [...] Scelsi di rendere il sistema compatibile con Unix, in modo che fosse portabile, e che gli utenti Unix potessero passare facilmente ad esso. [R. M. Stallman]

# Hacker vs Cracker

I mezzi di comunicazione  
hanno creato il dualismo  
hacker ↔ pirata

Dal «Jargon File (4.3.3, 20 Set. 2002)»:

- ◆ ***hacker*** : 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.

L'uso del termine «hacker» nel senso di «pirata» è una confusione di termini creata dai mezzi di informazione. Gli hacker si rifiutano di riconoscere questo dualismo, e continuano ad utilizzare la parola nel senso di «uno che ama programmare, e a cui piaccia essere bravo a farlo».

La comunità hacker ha poi creato il termine «cracker» per indicare quelle persone che invece vogliono danneggiare il prossimo.

---

Lecture consigliate:

- Hackers di Steven Levy
- L'etica hacker di Pekka Himanen

# La licenza GPL

## È la licenza di ogni prodotto software del progetto GNU

L'acronimo GPL sta per General Public License e si basa su 4 libertà:

- Libertà 0, o libertà fondamentale. **La libertà di eseguire il programma per qualsiasi scopo.**
- Libertà 1. **La libertà di studiare il funzionamento del programma**, e adattarlo alle proprie esigenze.
- Libertà 2. **La libertà di redistribuire copie del programma.**
- Libertà 3. **La libertà di migliorare il programma**, e distribuirne i miglioramenti.

La licenza del progetto GNU, la GPL, non si limita a concedere all'utente del programma le 4 libertà suddette, ma si occupa anche di proteggerle: **chi modifichi un programma protetto da GPL e lo distribuisca con le modifiche, deve distribuirlo sotto licenza GPL.** È grazie a questa protezione che la GPL è ad oggi la licenza più usata per il software libero.

Quando si usa la GPL occorre bene avere in mente la distinzione che c'è tra il concetto di «normale utilizzo» del programma e la sua «modifica e/o distribuzione» poiché solo così sappiamo quando violiamo la licenza o no.

# «Software Libero» o «Free Software»?

Quando si parla di «free software» ci si riferisce alla «libertà», non al «prezzo»

«Free as free speech and not as free beer» [*R. M. Stallman*]

Le licenze GPL e LGPL sono progettate per assicurare la libertà di utilizzo, modifica o copia del software e di farsi anche pagare per questo (se si vuole), quindi **un programma coperto da licenza GPL non è automaticamente un programma gratis** (e viceversa)!!!.

La GPL poi dice anche che solo chi possiede già il programma può avere «diritti» su di esso, quindi **rilasciare un programma sotto GPL non significa che questo debba essere pubblicato necessariamente sul web!!!**

---

Per ulteriori informazioni si veda:

- [www.fsf.org](http://www.fsf.org)    [www.fsfeurope.org](http://www.fsfeurope.org)
- [www.gnu.org](http://www.gnu.org)
- [www.linux.it/GNU](http://www.linux.it/GNU)

# Risposta: software libero!

## La libertà del software: una necessità

[...] Ad oggi, il software libero è ampiamente diffuso in ambito accademico, industriale e fra gli appassionati di calcolatori, soprattutto grazie ai sistemi GNU/Linux. Questi sistemi liberi sono disponibili a costi molto bassi, ben inferiori a quelli di analoghi sistemi proprietari. Tuttavia, a causa delle loro caratteristiche, il loro uso richiede una buona cultura di base nel campo del software.

In ambito accademico viene molto apprezzata la possibilità di personalizzare ogni parte del sistema, visto che i programmi liberi sono liberamente modificabili (libertà numero uno). In ambito industriale, si apprezza l'affidabilità dei sistemi liberi, dovuta al fatto che quando un utente corregge un errore in un programma solitamente rende disponibile la correzione agli altri utenti (libertà numero tre). Gli appassionati di calcolatori apprezzano lo spirito di condivisione esistente fra gli utenti di software libero.

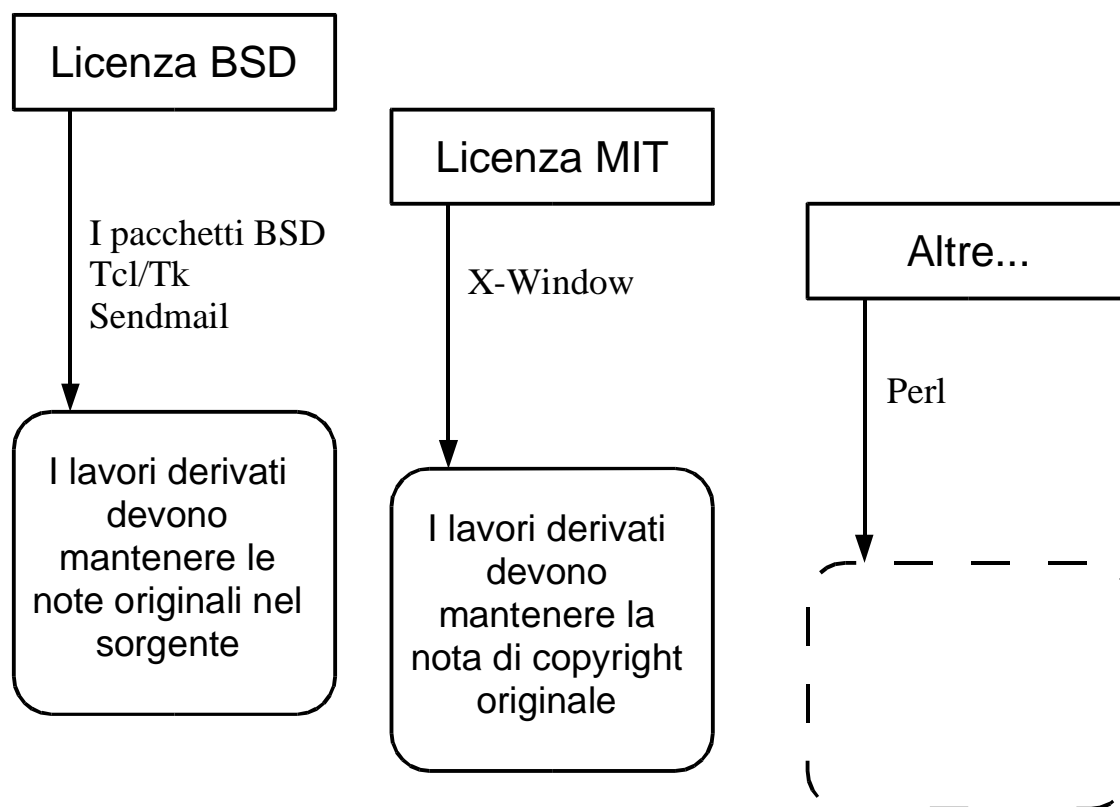
Ma le implicazioni dell'uso del software libero non sono soltanto tecniche ed economiche, perché il software da tempo ormai è avviato ad occupare un ruolo di primo piano nella nostra vita quotidiana, ed è destinato a cambiare in maniera profonda la società.

È per queste ragioni che la nostra libertà futura dipenderà anche dalla capacità di ognuno di noi di controllare il software. È per queste ragioni che ai tradizionali principi di libertà sessuale, di culto, di movimento, di espressione deve essere affiancata la libertà del software. È per queste ragioni che la nostra libertà futura dipenderà anche dall'uso di software libero.

[*Francesco Potorti -  
Associazione Software libero*]

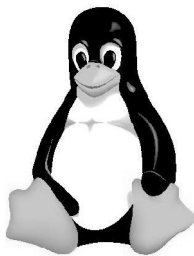
# Le «altre» licenze del software libero

Esistono diverse licenze che rispettano il concetto di «software libero»



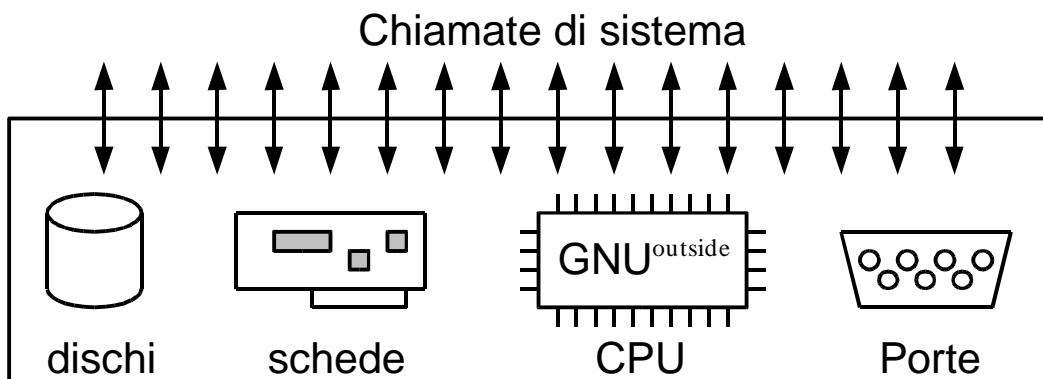
La licenza GPL, al contrario di altre licenze che rispettano comunque il concetto di «software libero», si preoccupa di proteggere il lavoro dei programmatori che hanno rilasciato il loro software come software libero.

# Linux è il nucleo del «sistema GNU»



Il progetto GNU doveva partire dal nucleo; cosa che non è accaduta

Il nucleo (o *kernel*) è quella parte di codice che implementa tutte le politiche di gestione dei dispositivi e dei processi presenti sul sistema.



Quando si dice che sul proprio computer si «è installato Linux» si commette un errore, bisognerebbe dire che si ha un sistema con diversi programmi del progetto GNU e come nucleo Linux.

Generalmente quando si vuole utilizzare la dizione corretta si dice che si utilizza sul proprio computer il sistema GNU/Linux.

# Le «distribuzioni»

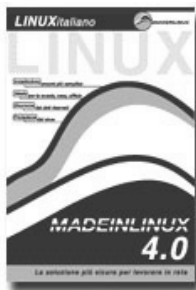


Molte distribuzioni usano applicazioni GNU e Linux come kernel

slackware  
linux



Linux  
Mandrake™



Ogni distribuzione ha le proprie caratteristiche: pacchetti supportati, gestione delle dipendenze, applicazioni di configurazione, *tool* di sviluppo, ecc..

Una distribuzione può contenere al suo interno software proprietario o libero, più questo rapporto si sposta verso il software libero e più si dice che la distribuzione si avvicina al modello «ideale» del sistema GNU/Linux.

# Lavorare con il software libero - 1

## Dal punto di vista del produttore di software

- Si riesce a vendere solo poche copie (tranne casi eccezionali).
- La commercializzazione è solitamente per contatto diretto con il cliente.
- Il ritorno economico avviene solitamente tramite i contratti di assistenza e/o consulenza.
- Si possono utilizzare le soluzioni software già adottate da altri diminuendo quindi i tempi di sviluppo.
- Si hanno a disposizione moltissimi *tool* di sviluppo a costi ridotti favorendo così anche l'iniziativa privata.
- Le conoscenze di base sono tutte disponibili in rete e di accesso immediato.
- Per poter lavorare occorre una conoscenza degli aspetti tecnici più elevata del normale per poter essere sempre «aggiornati» sulle continue innovazioni e fornire ai clienti consulenze di qualità.

# Lavorare con il software libero - 2

## Dal punto di vista dell'utente

- È possibile utilizzare il programma senza i vincoli (alle volte assurdi) delle licenze «proprietarie».
- I dati generati con i programmi liberi possono essere convertiti facilmente in formati diversi rendendoli quindi facilmente accessibili anche dopo molti anni.
- È possibile redistribuire il programma od utilizzarlo su più piattaforme senza vincoli.
- Il prodotto software è svincolato dall'assistenza tecnica che può quindi essere richiesta a fornitori diversi.
- Il prodotto software non «muore» se il produttore lo abbandona (volontariamente o involontariamente).
- L'eventuale costo del prodotto software, che è comunque commisurato al lavoro fatto e non è predeterminato a priori, o della sua assistenza può essere ripartito sul numero di sistemi che lo utilizzano.
- Si possono utilizzare anche macchine ormai considerate «obsolete» per implementare semplici sistemi di varia utilità.

# Riflessioni su alcune questioni «moralì»

Il mercato attuale dell'informatica:

- È teatro di inutile consumismo.
- Necessita di grossi investimenti anche per compiti semplici.
- È dominato dai colossi del settore e tende ad oscurare i singoli programmatori e le piccole imprese.

Quello che può fare il «software libero»:

- Limitare lo spreco di macchinari, e di conseguenza l'inquinamento, permettendo l'uso di macchine obsolete per i compiti poco onerosi.
- Aprire il mercato all'iniziativa individuale facendo prevalere «i più bravi» grazie alla possibilità di poter scegliere e giudicare in base alle caratteristiche del prodotto.
- Permettere una maggiore diffusione della cultura informatica in maniera più capillare sul territorio.
- Permettere il «dirottamento» di grossi capitali che ora vengono usati prevalentemente per acquistare licenze verso la ricerca e lo sviluppo.