Operazione Rif. PA 2022-17295/RER approvata con DGR 1379/2022 del 01/08/2022 finanziata con risorse del Programma Fondo sociale europeo Plus 2021-2027 della Regione Emilia –Romagna.

Progetto n. **1** - Edizione n. **1**

MODULO: N. 6

Titolo: SICUREZZA DEI SISTEMI INFORMATICI

DOCENTE: MARCO PRANDINI

FACOLTATIVO - Breve storia delle licenze e di GNU/Linux

Un po' di storia

- Cronologia dello sviluppo dei sistemi operativi moderni, ed in particolare di UNIX
- Open source e free software
- La Free Software Foundation ed il progetto GNU
- Avvento e sviluppo del kernel Linux

Cos'è Linux?

Linux è un *sistema operativo* *Unix-like* *multitasking* *multiuser* *32 e 64 bit* che funziona su un'ampia varietà di piattaforme hardware ed è distribuito sotto una *licenza open source*

Per capire meglio i termini sottolineati della definizione, può essere interessante ripercorrere rapidamente alcune tappe della storia del calcolo automatico.

L'era del PC

Nel 1981 compare il PC IBM

- realizzato con componenti commerciali, potè essere imitato a basso costo in tutto il mondo, cosicchè la sua diffusione fu travolgente

Foto di *Boffy b,* rilasciata sotto licenza CC-BY-SA.
http://commons.wikimedia.org/wiki/Image:IBM_PC_5150.jpg

Un passo indietro

Mentre negli anni '80 UNIX progrediva...

- uscito dai Bell Labs viene riscritto dall'Università di California originando il ramo BSD
- i grandi produttori sviluppano le proprie varianti (AIX di IBM, HPUX di HP, IRIX di SGI, Solaris di SUN)
- raggiunge la maturità sui grandi sistemi multiprocessore
- viene dotato di un sistema grafico estremamente flessibile (X, sviluppato all'MIT)
- si presta a sperimentazioni sul kernel (Mach, di CMU, fu sviluppato come replacement del kernel di BSD ed oggi BSD/Mach è il cuore di MacOS X)

... sui PC viene installato un DOS

Tempi maturi

HW economico e potente, software proprietario e costoso

I tempi sono maturi perchè si manifestino le spinte sociali verso il software *free* ed *open*

- Spinta economica (free beer)
- Spinta idealistica (free speech)
- Spinta tecnica (open = learn & modify)

RMS

In quegli anni uno straordinario programmatore di nome Richard Matthew Stallman (RMS) lavorava agli MIT Artificial Intelligence Labs in un contesto di estrema libertà di condivisione del codice, finalizzata allo studio ed al miglioramento dell'esistente

All'inizio degli anni '80 il Lab acquistò nuovo HW corredato di SW proprietario, utilizzabile solo sotto un non-disclosure agreement.

RMS ed il progetto GNU

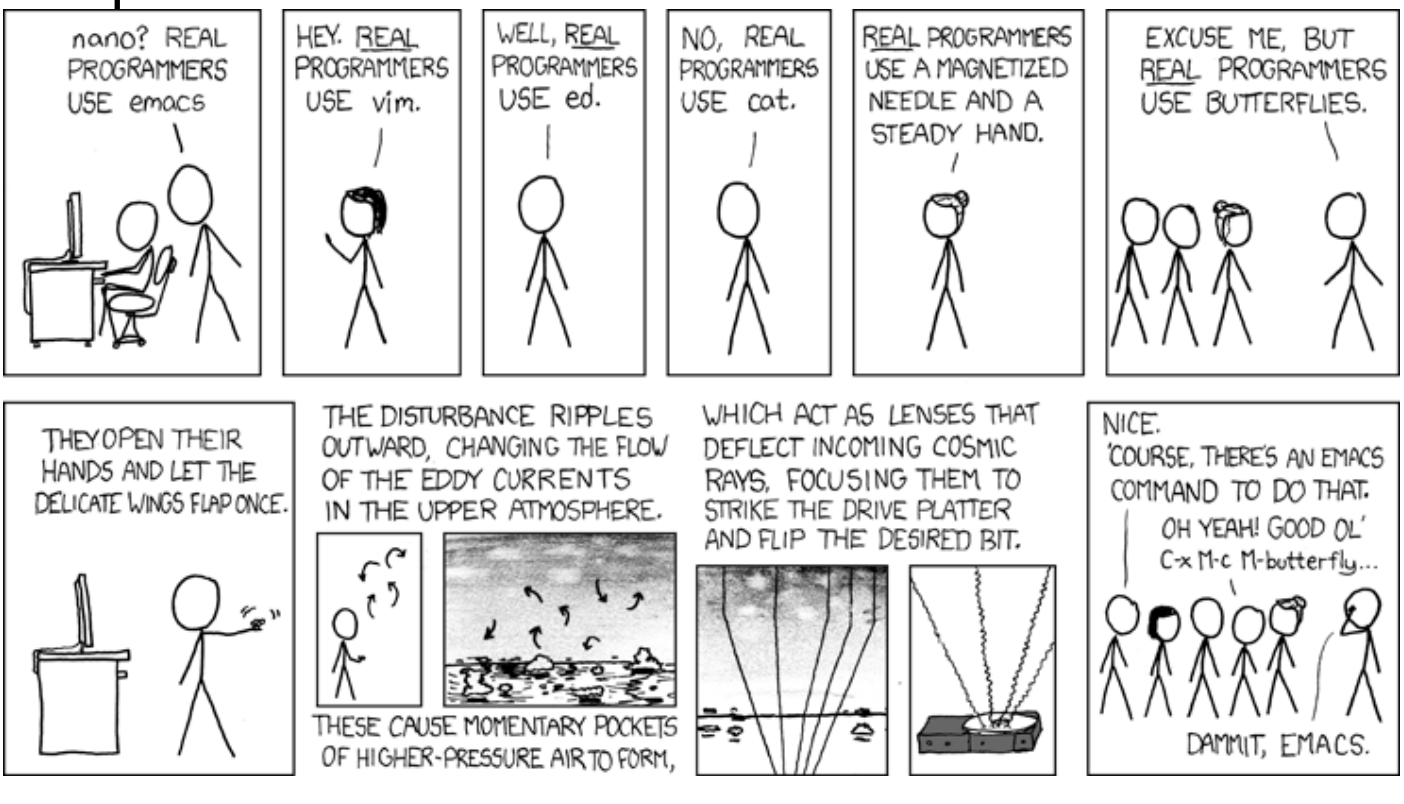
RMS fu deluso in modo profondo dal cambiamento:

> The rule made by the owners of proprietary software was, “If you share with your neighbour, you are a pirate. If you want any changes, beg us to make them.”

Abbandonò l'MIT per fondare il progetto GNU, convinto che per proseguire il suo lavoro come fatto fino a quel momento fosse indispensabile la totale libertà di scambio del software, a costo di dover ripartire da zero.

GNU EMACS

http://xkcd.com/378/



Il progetto GNU

Nell'ottica del progetto di software libero di RMS, naturalmente, non avrebbe senso pensare di sviluppare applicazioni se non disponendo di un sistema libero alla base

RMS si dedica quindi a GNU (GNU's Not Unix), un sistema operativo completo, libero e compatibile con Unix.

Un sistema di questo genere è ben più di un kernel, deve includere compilatori, editor, applicazioni di rete.

Tra il 1984 ed il 1990 furono sviluppati tutti i componenti, tranne il kernel, forse a causa dell'approccio molto ambizioso di RMS di scrivere un microkernel allo stato dell'arte, invece di portare oltre alle interfacce anche i meccanismi di Unix.

1991

All'inizio degli anni '90, quindi

- sono disponibili PC economici e realmente potenti per le necessità dell'epoca (avviene il salto generazionale da 16 a 32 bit con l'Intel i386)
- ma per queste piattaforme sono popolari solo S.O.
  - completamente chiusi e davvero limitati
    - (March 1993) MS-DOS 6.0
    - (December 1993) Windows for Workgroups 3.11
  - completamente chiusi ed embrionalmente evoluti
    - (October 1993) Windows NT 3.1 Workstation & Server
  - simil-Unix ma limitati, gratis ma non modificabili
    - minix

Linus Torvalds

Uno studente finlandese di nome Linus Torvalds, stanco delle limitazioni di Minix e desideroso di studiare le funzionalità dei nuovi processori a 32bit, si imbarca in un progetto che si trasforma rapidamente in un kernel ispirato a Unix.

L'annuncio è dato il 25 agosto 1991:

> ”Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.”

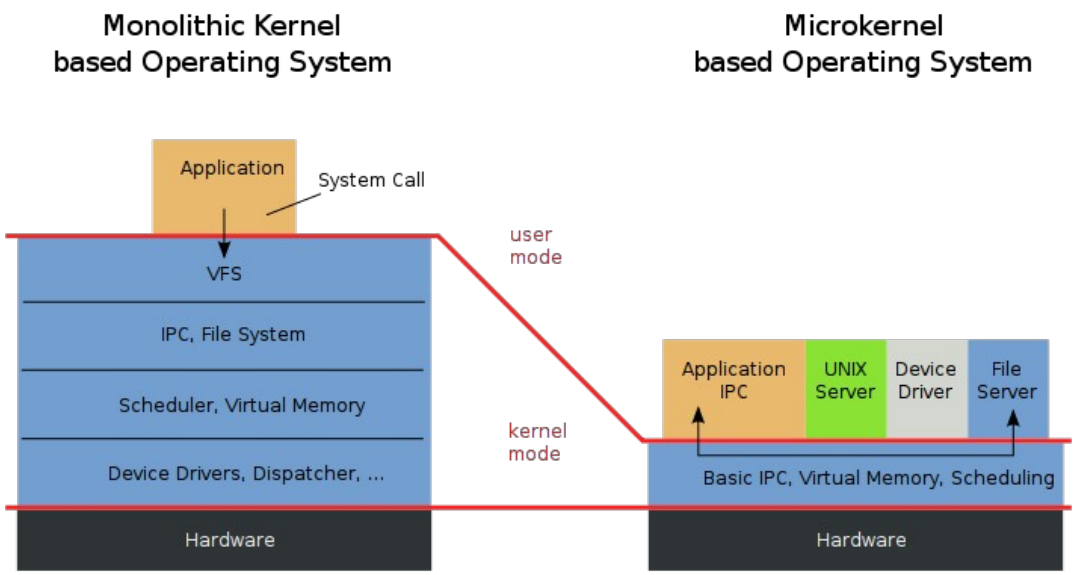
Linux

La disponibilità di tutti i tool GNU, principalmente il compilatore GCC, rende l'opera di Torvalds enormemente più semplice

Inizialmente l'autorità indiscussa dell'epoca, Andrew S. Tanenbaum, docente universitario ed autore di Minix, scartò Linux come malamente progettato

- il design sembrava troppo legato al processore
- il modello di sviluppo aperto alla comunità fu da AST tacciato di ingestibilità
- all'epoca sembrava tramontata l'era dei kernel monolitici, Minix e Hurd erano microkernel

Microkernel vs. Monolitico



Linux - i primi sviluppi

Nel giro di un anno e mezzo, Linux riceve sufficiente attenzione da giungere ad una versione numerata 0.99 di eccellente qualità

Questa viene rilasciata nel dicembre '92 con licenza analoga a GNU, e questa decisione decreta il suo successo

- si possono legalmente integrare i due progetti
- l'uno ha tutto ciò che manca all'altro
- dopo qualche controversia, viene quindi accettato globalmente che la denominazione corretta del sistema risultante sia GNU/Linux

Nel 1993 conta più di 100 sviluppatori attivi

Nel 1994 Torvalds lo considera abbastanza stabile e completo da meritare la versione 1.0

Le versioni del kernel

I kernel sono numerati con una terna *x.y.z* (più recentemente si è aggiunto un quarto numero)

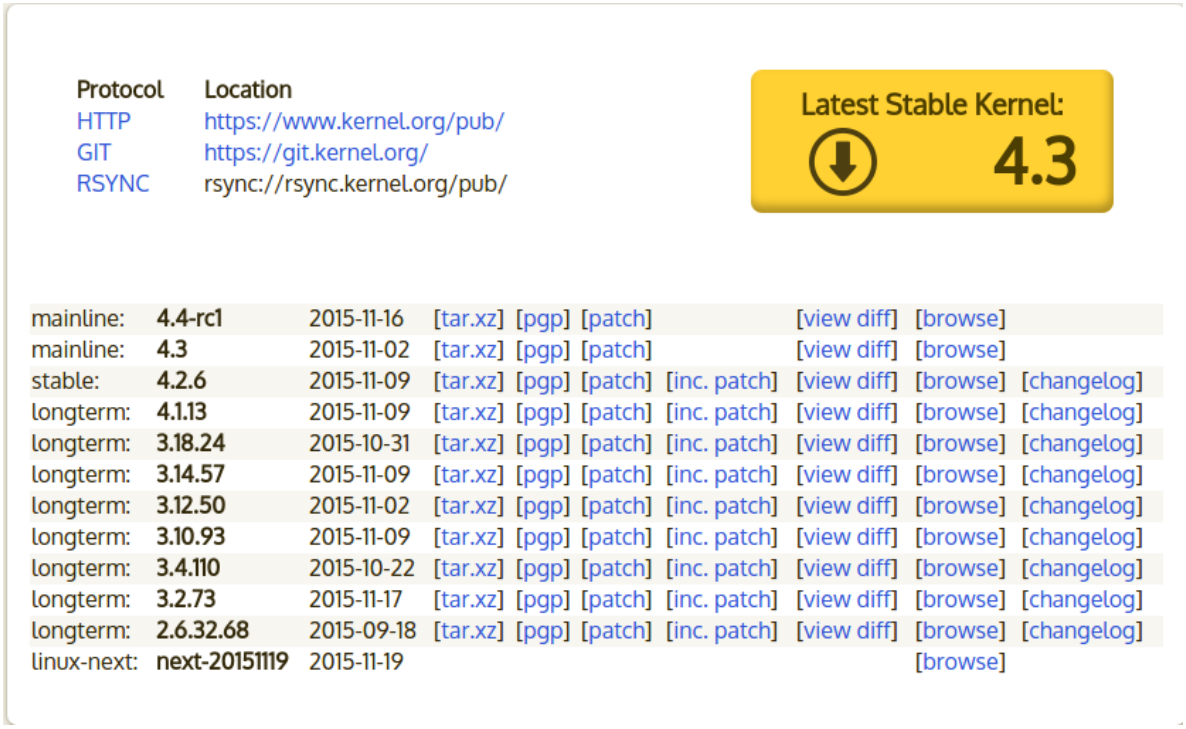
- Prima della versione 2.6.16, la prima cifra è cambiata solo in due momenti di svolta, da 0 a 1 e da 1 a 2
- La seconda cifra, fino alla versione 2.5, indicava un ramo stabile se pari e di testing se dispari
  - es. se il kernel stabile è il 2.4.x, il testing avviene sul 2.5.y; le migliorie considerate stabili porteranno a pubblicare il 2.4.(x+1)
  - dal 2.6, a causa delle dimensioni il ciclo di rilascio basato su major release divenne troppo lento, ed il testing fu ”spostato alla terza cifra”, cioè non esiste un 2.7 ma le 2.6.pari.x sono stabili e le 2.6.dispari.y sono di test
  - Dal 2008 in poi, il modello si è fatto più fluido, con rilasci frequenti di versioni stabili e un branch di sviluppo detto *linux-next*

Le versioni del kernel

Le principali innovazioni:

- 1.0 → 1.2 ('95): porting su DEC, Sparc e MIPS
- 1.2 → 2.0 ('96): modularizzazione, supporto SMP
- 2.0 → 2.2 ('99): forte miglioramento del supporto SMP, porting su m68k e PowerPC, nuovi filesystem
- 2.2 → 2.4 ('01): supporto a hw ”dinamico” (PnP, USB, PCcard), filesystem complessi (RAID, LVM)
- 2.4 → 2.6 ('03): innumerevoli aggiunte (passaggio da 16 a 32 bit di uid e pid, supporto proc. 64bit, kernel-based vm, vari filesystem, SELinux, ...)

www.kernel.org



Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel: 4.3

mainline:	4.4-rc1	2015-11-16	[tar.xz]	[pgp]	[patch]	[view diff]	[browse]	
mainline:	4.3	2015-11-02	[tar.xz]	[pgp]	[patch]	[view diff]	[browse]	
stable:	4.2.6	2015-11-09	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.1.13	2015-11-09	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.18.24	2015-10-31	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.14.57	2015-11-09	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.12.50	2015-11-02	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.10.93	2015-11-09	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.4.110	2015-10-22	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.2.73	2015-11-17	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
longterm:	2.6.32.68	2015-09-18	[tar.xz]	[pgp]	[patch] [inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20151119	2015-11-19					[browse]	

Ad oggi

Le architetture supportate sono praticamente tutte quelle dotate di una PMMU: Alpha AXP, Sun SPARC, Motorola 68000, PowerPC, ARM, Hitachi SuperH, IBM S/390, MIPS, HP PA-RISC, Intel i386, Intel IA-64, AMD x86-64, AXIS CRIS, Renesas M32R, Atmel AVR32, Renesas H8/300, NEC V850, Tensilica Xtensa, Analog Devices Blackfin

Il progetto uClinux ha portato il kernel su molti microcontrollori senza PMMU per sistemi embedded

Android è fortemente basato su Linux

Diverse distribuzioni girano su Raspberry PI

Linino è un porting di Linux su Arduino …

Quanto vale Linux?

Nel 2006, uno studio finanziato dall'UE ha stimato il costo teorico per riscrivere da zero il kernel versione 2.6.8 in **882 milioni di €**. La rapida crescita di complessità ha fatto lievitare la stima nel 2011 a **2,2 miliardi di €**

Nel 2008 la Linux Foundation ha stimato che per ricreare un intero sistema come Fedora 9 (204.5 millioni di linee di codice in 5547 pacchetti applicativi) sarebbero necessari circa **60,000 anni-uomo**, distribuiti nell'arco di **25 anni**, con un budget di **10,8 miliardi di $**

Linux vs. UNIX

Linux mira alla compatibilità POSIX (standard IEEE 1003 - ISO/IEC 9945), che certifica nel mondo UNIX l'uniformità ed interoperabilità

- delle API che il S.O. espone alle applicazioni
- dei comandi di utilità (centinaia!)

La conformità POSIX non è facile da garantire per un progetto open source a causa del costo della documentazione

- The Open Group ha quindi emesso uno standard aperto analogo (Single UNIX Specification)
- Il documento di riferimento per Linux è comunque la Linux Standard Base (LSB)

Linux vs. UNIX

In termini pratici, si può affermare che

- *ogni* UNIX ha una certa consistenza, mentre Linux ha preso le varianti migliori da ciascuno
- allo stesso modo gli UNIX commerciali tendono ad essere ottimizzati per le piattaforme HW corrispondenti, mentre Linux pur essendo nato su i386 è il sistema più portabile che ci sia

Linux vs. Windows

È piuttosto difficile indicare le differenze tecniche, visto quante sono...

Compatibilità

- In rari (ma notevoli) casi i produttori di HW non rilasciano le specifiche e scrivono driver proprietari. Ovviamente per Windows li realizzano tutti, mentre Linux è tuttora trascurato
- È comunque più frequente il caso in cui, grazie alla natura open di Linux, esista solo per quest'ultimo il supporto a dispositivi o filesystem esotici

Linux vs. Windows

Efficienza:

- In Linux non ci sono componenti applicativi legati al kernel, mentre in Windows l'interfaccia grafica è parte integrante del S.O.
- Similmente la dipendenza di molte applicazioni Windows dalle librerie di sistema è intricata, mentre in Linux le applicazioni sono pressochè autocontenute

Sicurezza

- Benchè da tempo Windows sia seriamente multiutente, la filosofia d'uso del sistema da parte della maggior parte degli utenti (e degli sviluppatori di applicazioni) si è formata prima
- Nel mondo UNIX (o fin dai tempi di Multics) per contro, la filosofia di separare i privilegi ed assegnarli a utenti diversi è innata

FLOSS

FLOSS: Free/Libre Open Source software

Cosa si intende per SW libero e SW open source

Chi si occupa di difendere il SW che non ha un proprietario?

Il modello di sviluppo e di business

La Free Software Foundation

È importante definire il significato di ”libero” - in inglese il termine *free* infatti significa anche gratuito, ma non è questo il significato legato alla dicitura *free software*

> "Free Software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech", not as in "free beer." --RMS

Per diffondere la cultura del software libero e garantirne i diritti, RMS fonda la Free Software Foundation

La Free Software Foundation

La definizione di software libero della FSF poggia su 4 principi:

- Libertà di usare l'applicazione per qualsiasi scopo;
- Libertà di modificare l'applicazione per adattarla alle proprie necessità;
- Libertà di ridistribuire copie dell'applicazione;
- Libertà di modificare l'applicazione e distribuire la versione modificata a chi la voglia usare.

Difesa del Software Libero

Per difendere il software libero, la FSF utilizza a suo vantaggio la legislazione sulla protezione della proprietà intellettuale (copyright)

Il software è accompagnato da una licenza vincolante (copyleft o Permesso d'Autore)

- garantisce le libertà definite dalla licenza
- vincola l'utilizzatore al rispetto delle stesse libertà

CopyLeft

Il diritto d'autore in Italia proteggere le opere dell'ingegno di carattere creativo. E' una legge che ha più di 50 anni e con il tempo sono state aggiunte altri tipi di opere da essa tutelate.

Il software oggi è tutelato dalla legge sul diritto d'autore così come lo è un'opera letteraria, un brano musicale o un film.

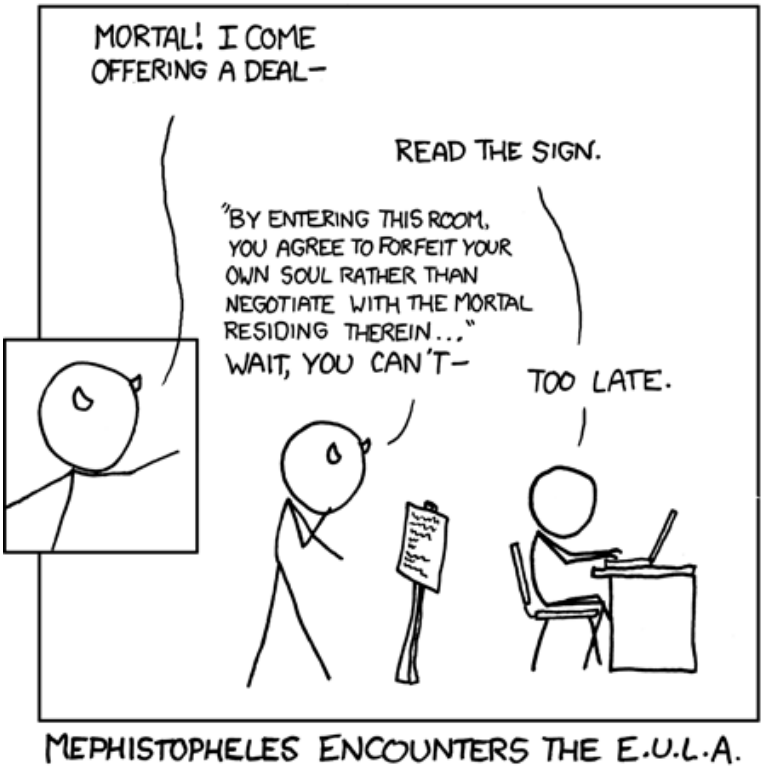
L'autore di un software nel distribuire il suo programma definisce gli utilizzi consentiti del software stesso tramite un contratto chiamato licenza d'uso software.

Il permesso d'autore non è una licenza specifica ma un meccanismo da applicare ad una licenza.

Il permesso d'autore usa le leggi sul diritto d'autore, ma le capovolge per ottenere lo scopo opposto a quello abituale: invece che un metodo per privatizzare il software, diventa infatti un mezzo per mantenerlo libero.

Licenze

http://xkcd.com/501/



Licenze

Troppe per citarle tutte...

http://www.gnu.org/philosophy/license-list.html

la proliferazione delle licenze è uno dei problemi del software libero, perchè complica il meccanismo fondamentale della creazione di nuovi prodotti per integrazione di quelli esistenti

In sintesi estrema:

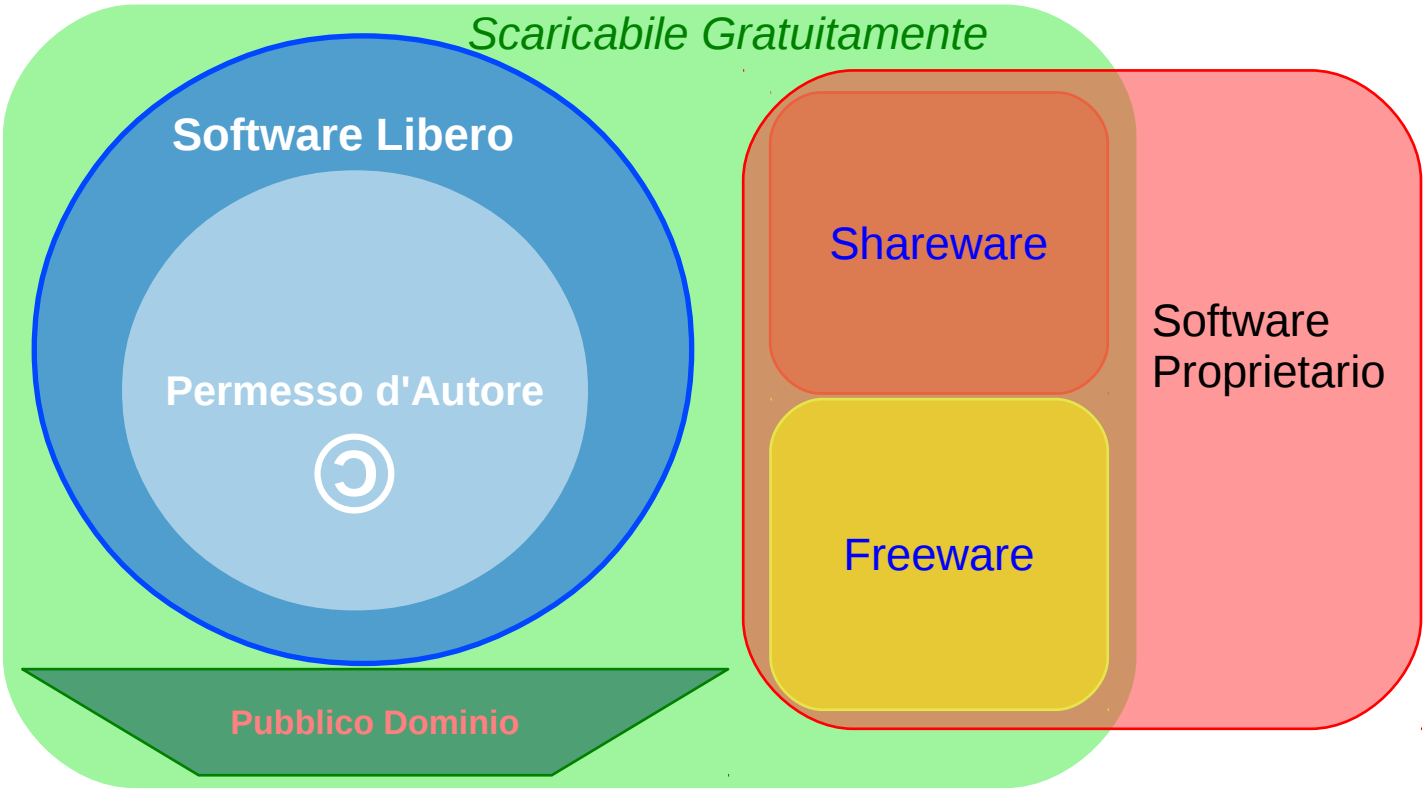
copyleft: attestano le libertà fondamentali ed impongono che i prodotti derivati siano distribuiti sotto la stessa licenza

free: attestano le libertà fondamentali

public domain: non impongono nessuna licenza al prodotto

http://www.gnu.org/licenses/licenses.it.html

Le tipologie di software



Open Source Initiative

Nel 1998 viene fondata la OSI per sostenere il mondo del software con sorgente aperto (open source), con un approccio diverso e più vicino al mondo economico tradizionale

- La FSF sostiene l'importanza sociale della propria causa: consentire alle persone l'uso libero e aperto del software a vantaggio dell'umanità
- L'OSI sostiene che l'accesso al codice sorgente e la possibilità di modificare il software ricevuto per migliorarlo sia una necessità pratica, non un diritto sociale.

È di base uno sforzo di chiarire l'eterno malinteso sul *free* software

Definizione di Open Source Software secondo OSI

OSS non è semplice accesso al codice sorgente, ma rilascio sotto una Open Source License (OSL) che garantisce all'utente altre libertà

Le proprietà delle OSL

1. Free redistribution

   The application can be redistributed, either on its own or as a part of a bundle of other applications. No royalty or other fees are required to do this.

2. Source Code

   The Source Code for the project must be easily available, if it is not downloaded with the compiled code (human readable code compiled into code the machine can read) clear instructions must be given as to where and how the source code can be obtained.

Le proprietà delle OSL

3. Derived Works

   The license must allow applications to be modified and distributed again with the same license as the original application.

4. Integrity of the Author's Source Code

   If the author wishes to keep their Source Code as is, they may stipulate that modified Source Code cannot be distributed, only if they then allow files (patches) to be distributed with the application that will modify it at runtime.

   Questa clausola è essenzialmente a difesa della riconoscibilità del contributo originale

Le proprietà delle OSL

5. No Discrimination against Persons or Groups

   The author may not discriminate against any person or group; the product must be made available to all people who want to use it.

6. No Discrimination against fields of Endeavour

   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Le proprietà delle OSL

7. Distribution of License

   The same license applies to all the people who the application is redistributed to.

8. License must not be specific to a Product

   If an application is taken from a bundle of applications that was released as Open Content, then that application has the same license as the original bundle of applications.

Le proprietà delle OSL

9. License must not restrict other Software

   The License the application is released under cannot specify that it can only be distributed or used in conjunction with Open Source software.

10. License must be Technology Neutral

    For example: the license cannot specify that to use the software you must download it from a web page, or from a CD-ROM. The license must allow modification of the application so that it can be used in all environments.

GPLv3

Esaminiamo alcuni aspetti fondamentali della licenza sotto cui è distribuita la gran parte di un sistema GNU/Linux

GNU General Public License version 3

http://www.gnu.org/licenses/gpl-faq.html

Come interagiscono i termini di questa licenza nel processo di sviluppo di una software house?

GPLv3

Binari/sorgenti

Software GPL distribuito in forma binaria sotto qualunque forma (ad esempio in un dispositivo hardware) deve essere accompagnato dal codice sorgente, o almeno dalle istruzioni su come ottenerlo

Estensioni ed aggregati

ogni estensione di un programma GPL deve essere rilasciata sotto la stessa licenza (attenzione alle librerie!)

un aggregato è un set di componenti chiaramente separabili, ad esempio un'applicazione che gira su un kernel; un aggregato può contenere parti proprietarie e parti GPL, purchè sia possibile per chi lo riceve far valere su queste ultime i diritti sanciti dalla licenza

GPLv3

Strumenti e prodotti

- Software GPL (es. editor, compilatori, ...) può essere usato per sviluppare software proprietario
- Attenzione ai casi in cui nell'ouput è incluso parte dello strumento (es. script+interprete, come nel caso di bison)

Librerie

- Un programma che usa librerie ”pure GPL” deve essere rilasciato sotto una licenza GPL-compatibile.
- La stragrande maggioranza delle librerie GNU, però, sono rilasciate
  - sotto la GPL + una specifica eccezione che consente il link con qualunque applicazione, anche non free
  - sotto la **GNU Lesser GPL**

LGPL

La Gnu Lesser GPL (LGPL, GPL attenuata) è stata studiata per consentire la creazione di software proprietario basato su librerie free

- secondo la GPL infatti, poichè effettivamente all'atto dell'esecuzione il codice comprende la libreria, il totale sarebbe un'estensione e non un aggregato
- la LGPL impone di rispettare alcuni vincoli, essenzialmente a garanzia che le librerie usate non vengano ”chiuse” per effetto della distribuzione con l'applicazione proprietaria
- http://www.gnu.org/licenses/lgpl.html

Open Content

Il problema di garantire la libertà della documentazione è affine al problema della libertà del software, e più generale

La prima associazione a sviluppare licenze per qualsiasi prodotto della creatività è stata Open Content, fondata nel 1998

Nel 2003 è stata chiusa, poichè il fondatore ritenne che la neonata Creative Commons stesse riuscendo meglio nello stesso obiettivo, specialmente in termini di solidità legale.

Licenze Creative Commons

Rilasciare un'opera sotto una licenza CC non significa rinunciare al copyright

- anzi, il copyright non è cedibile l'attribuzione va sempre indicata   *ecchia versione*
- vengono però concessi alcuni diritti agli utilizzatori sotto precise condizioni
- se si vuole rinunciare ad ogni diritto si deve rilasciare l'opera nel pubblico dominio 
- le icone usate qui e nelle slide successive sono ”by: Creative Commons http://www.creativecommons.it/”

Licenze Creative Commons

Ci sono 6 licenze - tutte prevedono i seguenti tratti comuni

è richiesto all'utilizzatore di:

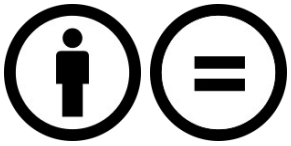
- chiedere il permesso di fare cose impedite dalla licenza
- mantenere intatti eventuali avvisi di copyright su tutte le copie dell'opera
- fornire un link alla licenza da tutte le copie dell'opera
- non alterare i termini della licenza
- non usare tecnologie per impedire ad altri utilizzatori usi leciti dell'opera

è concesso all'utilizzatore, nel rispetto delle condizioni di licenza, di:

- copiare l'opera
- ridistribuirla
- mostrarla o eseguirla pubblicamente
- diffonderla digitalmente
- cambiarla di forma senza alterarne il contenuto

Licenze Creative Commons

Attribuzione

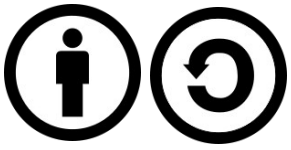


Attribuzione - Non opere derivate

Attribuzione - Non commerciale - Non opere derivate

Attribuzione - Non commerciale

Attribuzione - Non commerciale - Condividi allo stesso modo



Attribuzione - Condividi allo stesso modo

Il metodo di sviluppo

Gli straordinari progressi fatti da Linux in così breve tempo sono una realtà. Cerchiamo di spiegarne le cause.

- Il software proprietario è tipicamente sviluppato da piccoli gruppi di persone che lavorano a stretto contatto, e rilasciato quando ritenuto privo di problemi
- questo porta a tempi di rilascio lunghi (e i problemi restano)
- Il software aperto usa grandi comunità di sviluppatori e si basa sugli utenti per migliorare il codice. Questo modello impone di rilasciare spesso nuove versioni, che così, in presenza di una comunità competente ed entusiasta, progrediscono rapidamente.
- La differenza sui tempi di rilascio è persino più importante della competenza degli sviluppatori (vedi GNU vs. Linux)

La cattedrale ed il bazaar

Nel suo libro ”La Cattedrale ed il Bazaar” scritto tra il 1997 ed il 1999, Eric S. Raymond analizza scientificamente il fenomeno

- l'approccio che adotta è sperimentale, decide di sviluppare allo stesso modo di Linux un progetto che stava iniziando, *fetchmail*
- ESR definisce il metodo tradizionale ”cattedrale” - gruppi ristretti, isolati, rilascio pubblico di codice idealmente perfetto, e quello di Linux ”bazaar” - comunità aperta, delega di ogni attività possibile, rilasci frequenti piuttosto che accurati

La cattedrale ed il bazaar

Nel seguito mostriamo le principali ragioni per cui ESR ritiene che il modello ”bazaar” sia superiore a quello ”cattedrale”

Il testo è reperibile liberamente in originale ed in italiano

http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/

http://it.wikisource.org/wiki/La_cattedrale_e_il_bazaar

La cattedrale ed il bazaar

1. Ogni buon lavoro software inizia ”grattando un prurito” personale dello sviluppatore

   La motivazione è fondamentale per poter mandare avanti un progetto, e nel mondo open source se uno sviluppatore sente un'esigenza può facilmente trovarne altri che l'hanno uguale o simile e possono contribuire

2. I bravi programmatori sanno cosa scrivere. I grandi programmatori sanno cosa *ri*scrivere e cosa riusare.

   Con la disponibilità dei sorgenti, reinventare la ruota non è più necessario. ESR studiò 9 progetti di POP client, decise che nessuno aveva le caratteristiche che cercava, ma comunque basò fetchmail su uno di essi (fetchpop)

La cattedrale ed il bazaar

3. "Plan to throw one away; you will, anyhow." (Fred Brooks, The Mythical Man-Month, Chapter 11)
    - La modifica di fetchpop soddisfece parzialmente ESR, che continuando a cercare si imbattè in un'ulteriore alternativa, popclient.
    - Invece di adattare le caratteristiche utili di questo al progetto in corso, decise di buttare la prima versione e di usare solo l'esperienza fatta per ripartire da capo sulla nuova base di codice più adatta.

La cattedrale ed il bazaar

4. Con l'attitudine giusta, è facile trovare cose interessanti da fare
    - Nel caso di popclient, l'autore originale aveva perso interesse nel progetto.
    - ESR si trovò ”promosso” a mantainer, e anzichè proporre semplicemente qualche modifica divenne il responsabile dello sviluppo strategico
    - Questo è possibile solo se gli ”ideali” dell'open source sono condivisi:
5. Quando perdi interesse in un progetto, hai il dovere di lasciarlo nelle mani di un successore competente

La cattedrale ed il bazaar

6. Trattare gli utenti come co-sviluppatori è la strada di minima resistenza verso il rapido miglioramento del codice ed il debugging efficace
    - Nel mondo OSS è quasi sempre vero che una parte degli utenti è in grado di assistere fattivamente gli sviluppatori

La cattedrale ed il bazaar

7. Rilascia presto. Rilascia spesso. E ascolta i tuoi utenti.
    - La sensazione comune prima dell'avvento del ”bazaar” era che il rilascio di software instabile avrebbe causato inevitabilmente il suo abbandono
    - Linux provò il contrario, semplicemente correggendo in tempi rapidissimi ogni problema (spesso è stata rilasciata più di una versione di kernel al giorno!)
    - Questo approccio anzichè scoraggiare gli utenti li mantenne stimolati e gratificati
    - ”Stimulated by the prospect of having an ego-satisfying piece of the action, rewarded by the sight of constant (even daily) improvement in their work.” -- ESR; CatB

La cattedrale ed il bazaar

8. *Linus Law*: ”Given enough eyeballs, all the bugs are shallow” -- ESR
   - Data una base sufficientemente ampia di utenti co-sviluppatori, quasi ogni problema sarà caratterizzato rapidamente, e si troverà qualcuno a cui la soluzione apparirà ovvia
   - Questa è probabilmente la differenza chiave tra la cattedrale ed il bazaar
9. Strutture dati ben studiate e codice stupido funzionano molto meglio dell'opposto

La cattedrale ed il bazaar

10. Se tratti i tuoi beta-testers come la tua risorsa più preziosa, reagiranno diventandolo.
    - ”I released early and often (almost never less often than every ten days; during periods of intense development, once a day). I grew my beta list by adding to it everyone who contacted me about fetchmail. I sent chatty announcements to the beta list whenever I released, encouraging people to participate. And I listened to my beta-testers, polling them about design decisions and stroking them whenever they sent in patches and feedback.” --ESR; CatB
    - ”I got bug reports of a quality most developers would kill for, often with good fixes attached. I got thoughtful criticism, I got fan mail, I got intelligent feature suggestions.” --ESR; CatB

La cattedrale ed il bazaar

11. La seconda miglior cosa, dopo l'avere buone idee, è riconoscere le buone idee altrui. A volte è la cosa migliore in assoluto.

    Nel caso del progetto di ESR, l'idea di delegare la consegna della posta ai componenti esterni già pronti allo scopo fu di un utente. Il risultato fu un contemporaneo incremento delle funzionalità e della robustezza:

12. Spesso le soluzioni più innovative vengono dalla realizzazione che il modello iniziale del problema era sbagliato.

La cattedrale ed il bazaar

13. "Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher." (Antoine de Saint-Exupery)

14. Ogni strumento deve essere utile per quel che ci si aspetta, ma uno strumento davvero grande si presta ad usi inaspettati

[i punti da 15 a 18 sono specifici di fetchmail]

La cattedrale ed il bazaar

19. Dato un mezzo di comunicazione come Internet, ed un coordinatore capace di dirigere senza coercizione, molte teste sono meglio di una.

    In ”The Mythical Man - Month”, Fred Brooks enunciò la sua legge: ”il rendimento di un team di programmazione non cresce linearmente con le dimensioni. In particolare, aggiungere sviluppatori ad un progetto in ritardo lo ritarda ulteriormente”

Allora da dove arriva il successo di Linux?

Ego-less programming

La risposta viene da una osservazione esposta da Gerard Weinberg nel suo classico ”The Psychology of Computer Programming”:

Dove gli sviluppatori non sono ”territoriali” e incoraggiano i colleghi ad osservare e commentare il loro lavoro, i miglioramenti sono incredibilmente più rapidi.

In retrospettiva è chiaro come questa sia una correzione della legge di Brooks.

La dote di Linus Torvalds è stata quella di creare un ”mercato dell'ego” in cui la soddisfazione (tendenzialmente egoista) dei programmatori singoli sia legata ad obiettivi che possono essere raggiunti solo con un'ampia e continua cooperazione.

CatB - Conclusioni

I vantaggi tecnici...

> ”Perhaps in the end the open-source culture will triumph not because cooperation is morally right or software "hoarding" is morally wrong (assuming you believe the latter, which neither Linus nor I do), but simply because the closed - source world cannot win an evolutionary arms race with open - source communities that can put orders of magnitude more skilled time into a problem.” --ESR; CatB

...la mancanza di svantaggi economici

> Si stima che il *90%* dei costi di sviluppo sw non sia legato a prodotti commercializzati, che quindi, se resi OS, non porterebbero a perdite di fatturato...

Il software libero nelle imprese

Vediamo ora l'altro aspetto del software libero nelle imprese, non legato allo sviluppo ma all'utilizzo.

Sicuramente Linux è meno usato di quanto i suoi sostenitori si aspetterebbero.

Secondo i ”concorrenti” di Linux, questo è dovuto alla sua inadeguatezza all'uso professionale. Vediamo come questo sia errato.

Linux nelle imprese

Il primo problema è di ”direzione” della comunicazione.

- Linux è certamente nato come sistema per ”appassionati”, ed anche ora viene sostenuto tipicamente *dal basso*, dai tecnici, che tipicamente non sono in grado di farlo apprezzare ai loro superiori
- Le grandi software house, al contrario, fanno marketing *dall'alto*, direttamente ai CEO, CTO e CFO - le persone che prendono le decisioni ... e che tipicamente non hanno le competenze per apprezzare gli aspetti tecnici.

Linux nelle imprese

Un altro problema è di presentazione.

- Nell'ambiente d'impresa i sistemi operativi servono due segmenti ben distinti, server e desktop.
- Nessuno obietta alla solidità di Linux come server, ma è ancora diffusa l'idea che come desktop lasci a desiderare.
- Poichè i desktop si vedono e i server no...

Linux nelle imprese

Quali sono allora i punti di forza da sottolineare per un efficace ”marketing” di Linux?

- Costo
- Produttività
- Supporto
- Sicurezza
- Maturità

Ottenere Linux

Le distribuzioni

- cosa sono
- quale scegliere

Predisposizione del sistema

Pianificazione della convivenza con altri S.O.

Installazione

Distribuzioni

GNU/Linux è un insieme composto di elementi sostanzialmente fissi, reperibili in forma sorgente:

- kernel
- librerie di sistema
- utilità di base
- software applicativo di uso comune

Per portare su di un sistema questi componenti però serve

- o un altro sistema funzionante su cui scaricare e compilare tutti i pezzi
- o una *distribuzione*

Distribuzioni

Una distribuzione

- raccoglie in un supporto fisico pratico tutti i componenti, già compilati per l'architettura su cui si vuole installare GNU/Linux
- fornisce gli strumenti per l'avvio autonomo del sistema da installare
- tramite un installer propone la corretta sequenza dei passi di predisposizione del sistema
- fornisce strumenti per la gestione del software installato/installabile: il grande valore aggiunto è la gestione delle dipendenze

Distribuzioni: storia

La prima distribuzione (SLS) compare nel 1992, ma

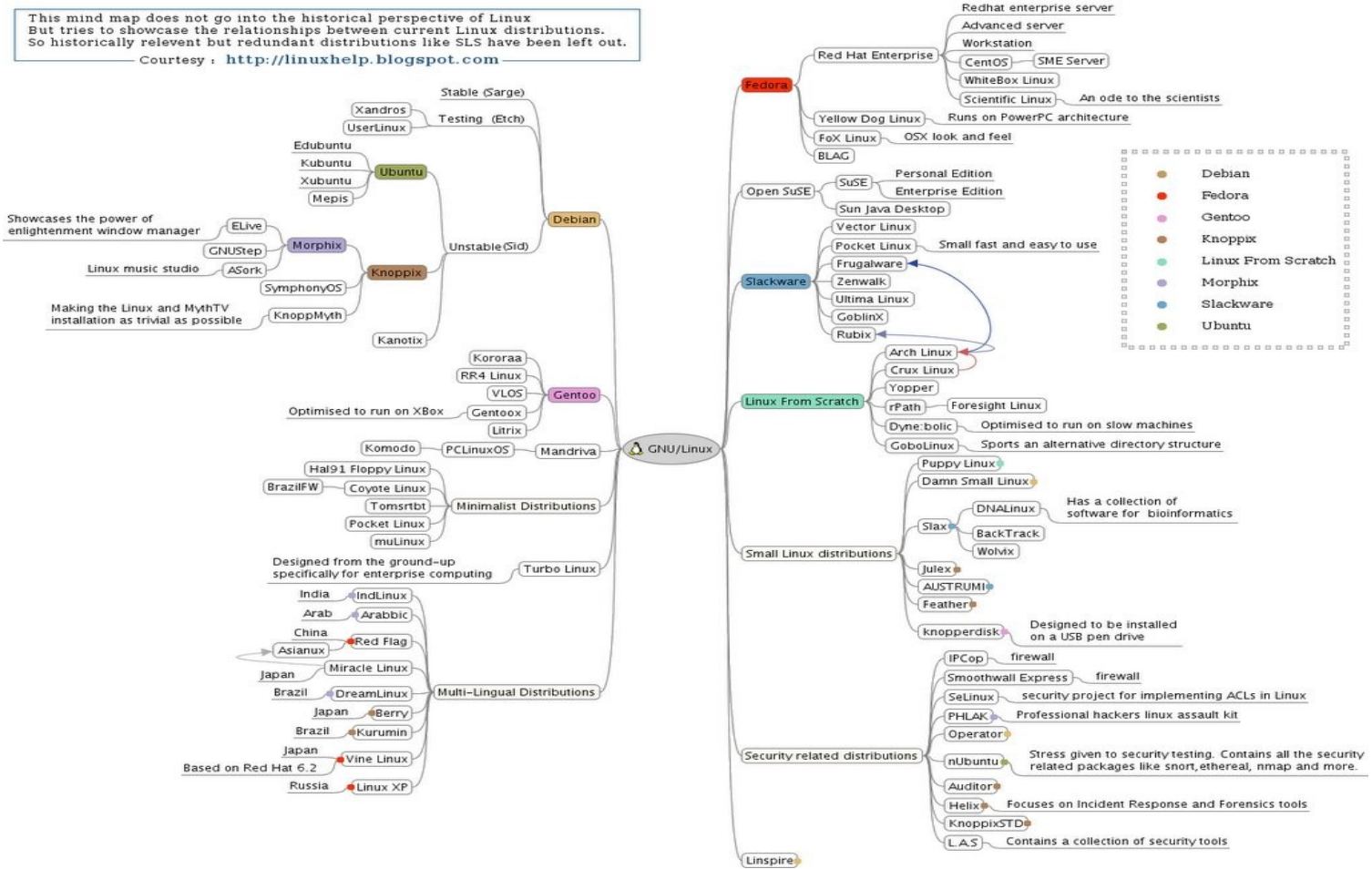
- è poco pratica anche per l'epoca
- contiene un mix di pacchetti con licenze libere e proprietarie che la rende difficile da usare senza grattacapi

Nel 1993 compaiono le grandi capostipiti:

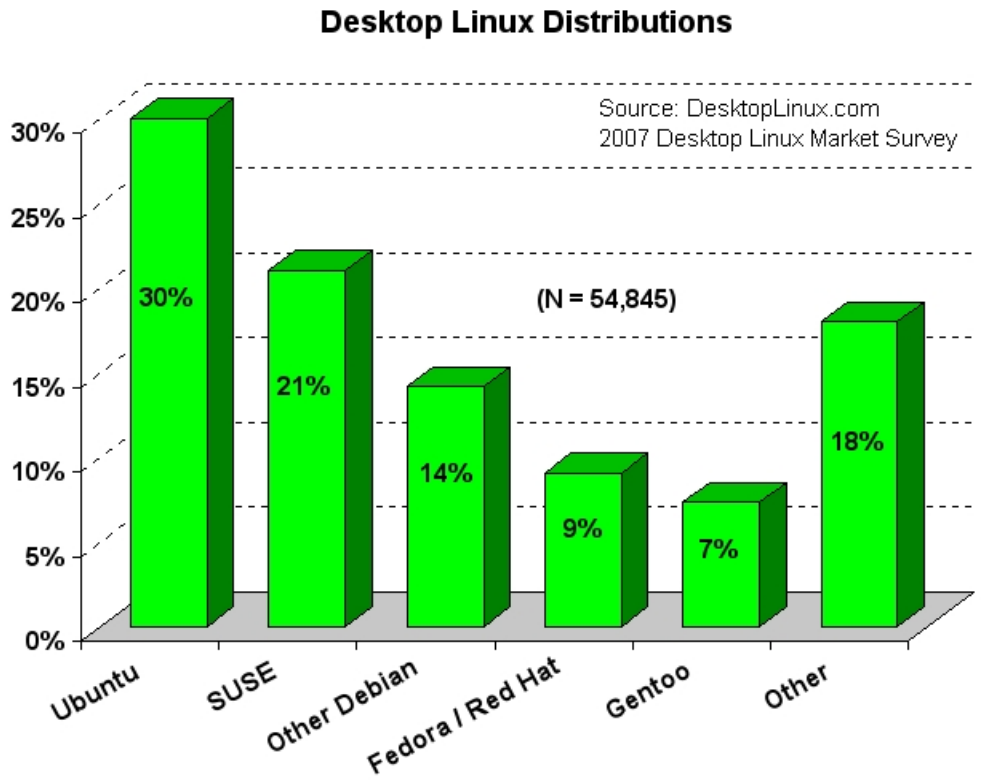
- Slackware, spartana ma efficiente, la più antica ancora in vita
- Debian, con un manifesto simile a quello di GNU

Nel 1994 viene lanciata RedHat, la prima distribuzione con chiari intenti commerciali

Distribuzioni: ad oggi



Sul desktop



Distribuzioni: criteri per la scelta

Architetture supportate

tutte le distribuzioni supportano i processori Intel 32bit, la maggior parte quelli a 64bit, alcune sono disponibili per tutte le varietà di processori su cui è stato portato il kernel

È bene ricordare che i pacchetti di terze parti potrebbero non essere disponibili per tutte le architetture supportate

Stabilità vs. Aggiornamento

il processo di rilascio frequente e continuo del software nel mondo GNU/Linux ha come conseguenza inevitabile che le versioni più aggiornate possano essere meno stabili

vi sono distribuzioni che hanno come filosofia l'inclusione dei pacchetti più recenti (e quindi con funzionalità maggiori) anche a costo di una minor robustezza, ed altre che garantiscono l'inclusione solo di software ben collaudato

Distribuzioni: criteri per la scelta

Supporto e durata

- La disponibilità di supporto garantito è tipica delle distribuzioni commerciali, ma anche con le distribuzioni gratuite più diffuse, in virtù della dimensione della relativa comunità di utenti, è semplice risolvere eventuali problemi
- Per installazioni di tipo server esistono varianti denominate LTS (Long Term Support): per 5/7 anni chi cura la distro garantisce che gli aggiornamenti non modifichino le API (tipicamente viene garantito solo il backporting dei security fix, non quello di tutti i bug fix)

Ampiezza del set di pacchetti

- Si va dai 1500 delle distro minimali ai 26000 di Debian
- Una scelta intelligente mette tutto l'essenziale in 1 CD

Distribuzioni: criteri per la scelta

Procedura di installazione

- Un installer adatto agli utenti inesperti non deve proporre scelte troppo dettagliate e complesse, che possono invece essere desiderabili per un utente con necessità più avanzate
- Alcuni installer possono contenere strumenti ausiliari, ad esempio per ricavare lo spazio per Linux su di un disco già occupato da altri S.O.

Strumenti per la gestione dei pacchetti

- Il sistema di packaging e la qualità dei relativi tool per cercare, installare, aggiornare e rimuovere software possono differire notevolmente
- I migliori tool sono in grado di gestire l'upgrade di versione dell'intera distribuzione

Distribuzioni: criteri per la scelta

Strumenti di configurazione

Sebbene le configurazioni avanzate siano tipicamente fatte manualmente dai sistemisti, la disponibilità di strumenti di ausilio basati su interfaccia testo o grafica può essere un criterio di scelta

Interfaccia grafica (desktop environment)

Il sottosistema grafico è tra i più complessi da configurare, e incide moltissimo sulle prestazioni, per cui poter scegliere il desktop environment adatto all'hardware disponibile può essere molto importante

in ordine approssimativo di ”peso” e funzionalità crescenti citiamo: JWM, FluxBox, Xfce, KDE, Gnome

Sul desktop