Free Software e Open Source: la libertá nel software

Festi Giorgio
festi.giorgio@gmail.com

January 17, 2006

Contents

1 Introduzione **2**

2 Cos'é libero? **3**

3 La nascita del Software **4**
3.1 L'avvento dei primi computer 4
3.2 La filosofia hacker 5
3.2.1 L'accesso ai computer deve essere libero. 5
3.2.2 L'informazione deve essere libera. 6
3.2.3 Dubitare dell'autoritá. Promuovere il decentralismo. . . . 6
3.2.4 Nessuna discriminazione. 6
3.3 Il primo software proprietario 6

4 GNU / Linux **9**
4.1 La Free Software Foundation e il progetto GNU 9
4.2 Linus Torvalds e il kernel Linux 12
4.3 X Window 14
4.4 Gnome e KDE 14
4.4.1 Gimp e il toolkit Gtk 14
4.4.2 Kde e il toolkit Qt 15
4.4.3 Gnome 15
4.5 Le distribuzioni 15
4.5.1 Le prime distribuzioni 16
4.5.2 Debian 16
4.5.3 Slackware 17
4.5.4 Yggdrasil 17
4.5.5 Red Hat 17
4.5.6 Caldera 18
4.5.7 Pacific HiTech 18
4.5.8 SUSE 19

5 Open Source e Free Software **20**

6 Pro e contro **22**
6.1 Economicitá 22
6.2 Sicurezza 22
6.3 Qualitá 24
6.4 Libertá 24

7 La trappola del software proprietario **25**

8 Le licenze **27**
8.1 Pubblico dominio 27
8.2 Shareware 27
8.3 Freeware 28
8.4 General Public License 28
8.5 Lesser General Public Licenze 28
8.6 Berkeley Software Distribution 28

9 Organizzazione **30**
9.1 Lancio 30
9.2 Espansione 30

10 Modelli economici **32**
10.1 Il caso Apache: suddivisione dei costi 32
10.2 Il caso Cisco: suddivisione dei rischi 33
10.3 Articolo civetta/posizionatore sul mercato 33
10.4 Widget frosting 34
10.5 Rivelare la ricetta, aprire un ristorante 34
10.6 Fornire accessori 35
10.7 Liberare il futuro, vendere il presente 35
10.8 Doppia licenza 35

11 Glossario **36**

A Allegati **37**
A.1 Lettera aperta agli Hobbysti, di Bill Gates, 3 febbraio 1976 . . . 37
A.2 Annuncio del progetto GNU, di Richard M. Stallman 27 settembre 1983 38
A.3 Annuncio di Linux, Linus Torwald 25 agosto 1991 40

Chapter 1

Introduzione

Questa tesi vuole analizzare due movimenti legati allo sviluppo e alla distribuzione del software "libero".
Si prenderá in considerazione la nascita del software, il Free Software, l'Open Source e il sistema operativo GNU / Linux (il primo progetto di software libero).
Cercherá di illustrare le licenze piu famose usate in questo settore, alcuni modelli economici alternativi al modello tradizionale.

Ringrazio Maurizio Napolitano per avermi mostrato questo altro volto del software ed avermi indicato la strada per entrare in questa filosofia.

Le opinioni espresse in questa tesi sono personali.

[1]Per leggere una copia della licenza visita il sito web http://creativecommons.org/Licenze/LegalCode/by-nc-sa.

Chapter 2

Cos'é libero?

Free as free spech not free beer! [**Richard Matthew Stallman**]

In inglese il termine free ha un doppio significato, ossia gratuito e libero. Questo termine venne utilizzato perché rimaneva comunque il piu adatto. Altri termini avrebbero sí risolto problema del duplice significato ma ne avrebbero creati altri oppure ridotto il significato originale. Lo scopo principale del Free Software non é legato a nessun fattore economico: nasce infatti dall'idea che l'informazione deve essere libera e dalla cultura del dono. Quindi é sbagliato definire il software libero solo come gratuito, in quanto incorpora una filosofia basata sulla condivisione che per alcune persone é piu importante del denaro.
Stallman concretizzó il concetto di libertá relativo al software in quattro punti:

1. libertá di eseguire il programma per qualunque scopo, senza vincoli sul suo utilizzo.
2. libertá di studiare il funzionamento del programma, e di adattarlo alle proprie esigenze.
3. libertá di redistribuire copie del programma.
4. libertá di migliorare il programma, e di distribuirne i miglioramenti.

Chapter 3

La nascita del Software

3.1 L'avvento dei primi computer

Era il 1958 e il Massachusetts Institute of Technology possedeva un Ibm 704, costatogli diversi milioni di dollari, il quale occupava una intera stanza e richiedeva continua assistenza da parte di personale tecnico.

L' IBM 704 fu un computer disegnato per calcoli ingegneristici e scientifici, trovó il suo utilizzo nel campo della ricerca ma anche per scopi commerciali. Tutti i modelli della serie 700 erano computer basati sull' elaborazione bath ossia risolvevano i problemi attraverso una sequenza di istruzioni precedentemente caricati. Non era possibile interagire con la macchina durante la sua elaborazione. L' IBM 704 era in grado di eseguire 4000 operazioni a secondo e fu il primo computer commercializzato su larga scala a gestire in modo completo i numeri reali.

Solo le universitá piú prestigiose potevano permettersi un computer e il modo per accedervi era abbastanza burocratico, gli studenti passavano il loro pro-

gramma scritto su delle schede perforate[1] ai tecnici e dovevano aspettare il loro turno, secondo una precisa gerarchia, per ottenere un risultato.
Le persone che potevano accedervi erano veramente poche e dovevano comunque dipendere dai tecnici.
Tra i vari club che l'universita offriva c'é n'era uno in particolare, il Tech Model Railroad Club dove i soci potevano costruire i modellini dei treni oppure occuparsi del sistema che li faceva girare. Ció che rendeva particolare questo club era la sua filosofia, ovvero chiunque poteva testare, riparare, perfezionare e modificare il modello.
All'interno del club si era sviluppato un vocabolario alternativo, tra cui la parola hack che significava qualsiasi progetto nato dal solo piacere di costruire.
Nel 1959 John McCarthy insegnava il primo corso di programmazione, e fu lui che chiamó per la prima volta la sua ricerca intelligenza artificiale sviluppando un programma che gli consentiva di sfidare il computer a scacchi. [2] Alcuni membri del T.m.r.c. dopo aver seguito il corso rimasero affascinati dall'arte della programmazione, e cercarono di avvicinarsi il piu possibile al gigantesco Ibm 704, sfruttando le ore notturne e ingraziandosi i tecnici.
Quando il Lincoln Lab regaló un computer chiamato Tx-o, che aveva usato per la costruzione di uno piu grande, il Tx-2, gli stessi membri si precipitarono e nessuno avrebbe potuto allontanarli.
Fortunatamente attorno a questo computer non esisteva la burocrazia che circondava l'ibm 704.
Bastava prenotare la propria sessione di lavoro su di un foglietto e chiunque poteva accedervi.
La atmosfera che circondava il T.m.r.c. si espanse attorno al Tx-o dettando alcune tacite regole, alla base della filosofia hacker.

3.2 La filosofia hacker

3.2.1 L'accesso ai computer deve essere libero.

Un metodo efficace per imparare come funziona qualcosa é smontarlo e rimontarlo. Gli hacker la pensavano esattamente cosi e detestavano qualsiasi ostacolo o barriera che impediva di imparare, aggiustare o migliorare qualcosa.
La maggior parte degli hacker di allora odiava guidare per il semplice fatto che il sistema stradale era, per loro, imperfetto e avrebbero voluto riprogettarlo da zero.
Se si pensa al notevole costo di un computer negli anni 60 si capisce perché il suo non utilizzo fosse uno spreco enorme! Capitava che qualcuno alla sera chiudesse a chiave i laboratori, ma era proprio durante la notte che gli hacker lavoravano.
Nacque cosi il Lock Hacking, ossia l'arte di aprire le porte.
C'era chi con un ferretto cercava di forzare la serratura, chi per certe porte faceva passare uno spago con dello scotch in modo che si attaccasse alla maniglia per poi farla ruotare, chi si intrufolava dal controssoffitto e molte altre tecniche furono

[1] le schede perforate rappresentavano l'unico input di computer come l'ibm 704, la programmazione a quei tempi avveniva senza nemmeno toccare un computer.

[2] Il programma tenne una sfida aperta con un professore dell'universitá e la perse. Ci furono persone che avanzarno la teoria che nessun computer poteva essere piú intelligente di un'essere umano. Gli attuali software di scacchi tengono testa ai campioni mondiali di questo sport e spesso hanno la meglio.

inventate per il solo scopo di raggiungere un computer. Oltre alle porte aprivano anche i lucchetti dell'armadio degli attrezzi e perfino cassette di sicurezza, ma tutte queste tecniche non vennero mai usate per rubare.

3.2.2 L'informazione deve essere libera.

Questa concezione era accettata da tutti gli hacker e ogni programma veniva lasciato alla portata di tutti. Ció evitava di reinventare la ruota tutte le volte e inoltre chiunque avrebbe potuto correggere eventuali errori offrendo un programma migliore. Gli hacker sono persone pigre, che odiano fare lo stesso lavoro piú volte, e se devono svolgere un lavoro ripetitivo, costruiscono un programma che lo svolga per loro. Prima di sviluppare un prodotto nuovo, gli hacker cercano che qualcuno non l'abbia gia fatto o per lo meno iniziato: ecco perche é importante partire da un base software libera. Linus Torvald si é basato su Minix per costruire Linux, cosi come Stallman cercó un compilatore libero,che non trov´, prima di costruirne uno suo.

3.2.3 Dubitare dell'autoritá. Promuovere il decentralismo.

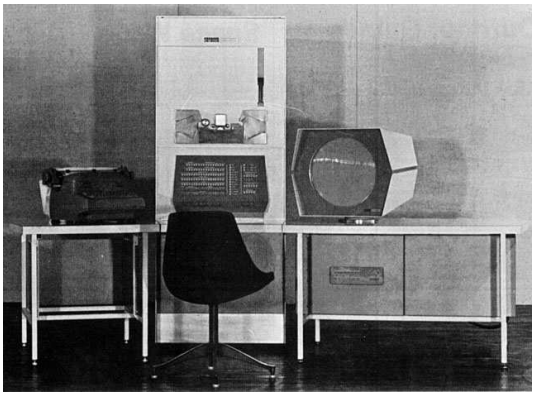
Per garantire il libero flusso delle informazioni il sistema doveva essere aperto, ovvero non dovevano esserci autoritá, si era tutti sullo stesso piano. Questo si é ripercosso sul metodo di sviluppo del software: chiunque puó collaborare ed essere al pari degli altri, anche se é l'ultimo arrivato. La cosa piú odiata era sicuramente la burocrazia, imperfetta e un ostacolo per la ricerca.

3.2.4 Nessuna discriminazione.

Gli hacker non avevano pregiudizi, questo nasceva dal fatto che si curavano mento delle caratteristiche superficiali e prestavano piú attenzione all'abilitá di programmare. Si veniva giudicati in base all'operato e non su falsi criteri come l'eta,la razza o il sesso. Cosi quando un bambino di 12 anni si presentó al laboratorio e mostró le sue capacitá venne subito accettato all'interno della comunitá, anche se faceva un pó rabbia sentire correggere il proprio codice da un ragazzino.

3.3 Il primo software proprietario

Nel 1961 il Massachusetts Institute of Technology ricevette in regalo il pdp-1, che fu subito al centro dell'attenzione dei membri del Tmrc. Fu su questo computer che Sauders e Russell svilupparono Spacewar, un gioco. Essendo il codice a portata di tutti, furono in molti a espanderlo, aggiungendo la gravitá, una migliore traiettoria dei missili e tutte le costellazioni visibili dall'equatore.



Il pdp-1 (Programmed Data Processor-1) fu il primo computer della serie PDP della Digital Equipment ed entró in commercio nel 1960.

Nel 1975 la societá Mits costrui il primo personal computer chiamato Altair basato su un microchip della Intel. Il suo prezzo era basso, ma ognuno doveva assemblarlo.Come computer non era un gran che, ma nonostante ció ebbe un successo enorme.

Mits logo

Nella primavera del 1975 il presidente della societá, Ed Roberts, ricevette una telefonata da due studenti universitari che gli proposero un interprete per il basic, gli rispose che altre persone giá ci stavano pensando, ma lui lo avrebbe comprato dal primo che glielo avrebbe presentato. E cosi Paul Allen e Bill Gates si misero a lavorare su un interprete che potesse stare in meno di 4 k: questa era la memoria dell'Altair. Non fu facile, ma Gates era un maestro nel trovare strade piu corte e nel semplificare il codice.

La differenza tra questo programma e tutti quelli che lo avevano preceduto era lo scopo, cioé la vendita e non il piacere di programmare. Di fatto, il programma era inserito nel catalogo come tutti gli altri componenti hardware aggiuntivi per l'Altair.

Altair logo

Alla prima convention mondiale l'interprete basic si fece vedere e fece infuriare parecchi programmatori che avevano giá spedito centinaia di dollari e non lo avevano ancora ricevuto.

Qualcuno raccolse un nastro contenente il programma di Allen e Gates, non ci volle molto per copiarlo e distribuirlo gratuitamente e non passó molto tempo che quasi tutti erano riusciti a procurarselo. Per i due studenti questo era un proprio e vero furto. Cosi Gates scrisse una lettera intitolata Lettera aperta sulla pirateria[3] rivolta a tutti i programmatori.

La lettera fu di grande effetto, Gates scrisse che le risposte avute dal suo basic erano tutte positive, ma solo il 10% lo aveva comprato e all'ammontare delle royalty ricevute avrebbe lavorato per 2 dollari l'ora. Denunciava che il mercato hobbystico non produrra mai del buon software e che la pirateria ne impedisce lo sviluppo scoraggiando chiunque, poi scriveva: *chi si puó permettere di documentare tutto il lavoro, fissare tutti i bug e poi distribuirlo gratis?*. In fine lasciava il suo indirizzo per chi volesse rispondergli o restituirgli i soldi. Gates ricevette tra le tre-quattrocento lettere, e solo 5 contenevano la somma dovuta. Il fatto era che quando Russel sviluppó spacewar, c'erano solo 50 computer

[3] Allegato 1

in circolazione, ma con una espansione cosí vasta si stava creando un mercato enorme che avrebbe reso ricco chiunque se ne fosse accorto in tempo.
Quando Tom Pittman scrisse un'altra versione del basic chiamata tyni-basic, si pose il problema di come venderlo. Gates si lamentava per i furti subiti e la gente gli rispondeva che se non costasse cosí tanto lo comprerebbero, cosi Tom vendette il suo programma per soli 5 dollari.
Funzionó, inizió a ricevere i soldi e lettere con 5 dollari dicendo che non serviva che gli venisse spedito il programma visto che lo aveva giá copiato da un amico e altre lettere con 10 che dicevano che era quello il giusto prezzo.

Chapter 4

GNU / Linux

Quando uscirono le prime distribuzioni, a Stallman stava molto a cuore che si chiamassero Gnu/Linux e non solo Linux, di fatto gran parte del lavoro era fatto dalla Fsf con il progetto GNU. Questa é la storia di come si é arrivati a costruire un intero sistema operativo alternativo e abbastanza competitivo da far traballare il monopolio detenuto da Microsoft con Windows.

4.1 La Free Software Foundation e il progetto GNU

> *É successo raramente nella storia, ma é successo, che in un momento in cui sembrava che le regole del gioco fossero immutabili, che i vincitori fossero imbattibili e i perdenti senza alcuna possibilitá di riscatto, un uomo solo, assolutamente privo di potere, ricchezze, fama, bellezza, amicizie, un uomo qualunque della specie innumerevole dei perdenti, riuscisse a sovvertire le regole del gioco e a far saltare il banco. [...] un uomo eccezionale che nasce perdente e diventa vincente, che non é bello ma affascinante, che non é simpatico ma é adorato come un dio; non ha amicizie vere ma conta migliaia di ammiratori; mette i lunghi capelli in bocca o nel piatto ove sta mangiando ma é conteso ospite alla tavola dei ricchi e dei potenti; non sorride mai ma si traveste da santo mettendosi in testa, a mo di aureola, la superficie attiva di un hard disk della prima generazione; non ha i soldi per pagarsi una cena al ristorante ma ha sconvolto un mercato da migliaia di miliardi di dollari.*
> **[tratto da Codice Libero]**

In questo passo si sta parlando di Richard Stallman, il fondatore della filosofia Free Software e colui che ha mantenuto l'arte della condivisione! Come abbiamo visto all'avvento dei primi computer, la condivisione avveniva in modo naturale, come lo scambiarsi formule di matematica o fisica. Fu con la nascitá dell'idea di proprieta intelletuale che nacquero i primi software proprietari! Stallman se ne rese conto quanto cominció a lavorare presso il Massachusset Institute of Tecnology, nel reparto di intelligenza artificiale, l'allora ,come lo definisce lui, *paradiso*

degli hacker e *casa sua*, il nono piano del Tech square! Trovó un'ambiente totalmente diverso da quello che c'era ad Harward, non esisteva una gerarchia da seguire per l'utilizzo di un computer, chi prima arrivava meglio alloggiava! Impara la cultura hacker, la necessitá di condividere il codice, di poterlo modificare e ridistribuire.
Quando la stampante del laboratiorio si inceppó per l'ennesima volta, gli bastó aggiungere qualche riga di codice che informava colui che mandava la stampa che c'era un errore! Soluzione semplice, ma geniale! Ma quando arrivó la nuova stampante, non poté piu operare in questo modo, la societá che aveva regalato la stampante al laboratiorio non aveva consegnato il codice sorgente! Quando Richard chiese il codice a un ex-dipendente questo rispose che aveva firmato un contratto che gli impediva di divulgare queste informazioni! Iniziava cosi a formasi dentro Stallman l'idea di software proprietario e del suo danno per la societá: non riusciva a capire perché una societá che ne avrebbe tratto solo profitti non vole mai rilasciare il codice sorgente!
Tra i lavori svolti al laboratiorio, di particolar importanza é TECO, un editor di testi! Il suo problema era che richiedeva di inserire una lunga sequenza di istruzioni seguita da un "end of command" per poter cambiare una sola frase, e come definirá Stallman era come giocare una partita a scacchi a occhi bendati. Un altro editor permetteva di aggiornare il testo dopo la battitura di ogni tasto, un'idea che Stallman decise di copiare, ma con qualche aggiunta! L'utente poteva tramite una combinazione di tasti attivare delle macro. Fu una rivoluzione. Vennero costruite tantissime macro, cosí tante che ormai il programma era diventato quasi un linguaggio di programmazione. Ci fu la necessita di riunire le principali macro, per evitare un'effetto Torre di Babele: grazie cioé alla grande personalizzazione del programma risultava molto complicato lavorare su quello di un'altro. Questa nuova versione venne chiamata EMACS. Gli hacker del laboratiorio avevano sviluppato un macchina e decisero che prima o poi avrebbero potuto fare una societá e iniziare a venderla. Grennblatt e Noftsker erano di questa idea, ma il modello su cui basare l'azienda fu la causa che fece nascere due societá, la Lmi e la Symbolics. Greenblatt aveva in mente un'azienda in pieno stile hacker e voleva il potere per manternerla tale, ma agli altri hacker non andava: avevano sempre lavorato nell'anarchia dove tutti erano alla pari. Inizió cosi una lotta tra le due societá, e la Symbolics assunse molti dei restanti hacker del laboratiorio. Stallman rimase al laboratorio, ma decise di schierarsi dalla parte della Lmi, quando la Symbolics coprí con il copyright il sistema operativo Lisp, che le due aziende avevano in comune. Stallman esaminava ogni nuova aggiunta fatta dalla Symbolics, scopriva come funzionava e la sviluppava per la Lmi. Da solo era riuscito a tenere testa a quello che facevano i migliori hacker che c'erano nel laboratorio.
Questo vicenda aumentó la sua giá grande fama di hacker, ma lo portó a una scelta cruciale, lasciare da parte le sue idee e programmare per un'azienda, o cambiare lavoro. Scelse di sviluppare un sistema operativo completamente libero e per fare ció si dimette dal MIT (1984) per evitare che potesse esercitare qualche diritto sul suo operato!
Il 27 settembre del 1983 Stallman pubblicó un'articolo [1] con oggetto "Nuova implementazione di UNIX", nel quale spiegava che avrebbe iniziato un progetto chiamato GNU,(acronimo ricorsivo che significa Gnu is Not Unix), che aveva

[1] allegato 2

come scopo la creazione di un sistema operativo libero simile a Unix.

Lo gnu é il logo principale del progetto omonimo. La pronuncia corretta é peró diversa, la G deve essere dura per evitare fraintendimenti con new. Ricorda Stallman che non c'é niente di nuovo in questo progetto da quello che succedeva negli anni 60 tra gli hacker del MIT.

Nell'articolo chiedeva aiuto economico ma anche collaboratori e programmi!
Il suo primo progetto fu il compilatore GNU e , come ogni bravo hacker, invece che scrivere da zero tutto il codice cercó prima se qualch'uno non lo avesse giá fatto. Trovó VUCK, prodotto da un universitá, ma il programma non era libero e quindi inutilizzabile per Stallman. Continuó la sua ricerca per arrivare a Pastel che, pur essendo libero, presentava un grosso problema di progettazione. Arrivó alla conclusione che se voleva un compilatore doveva svilupparlo da zero.
Nel 1984 lasció da parte il suo primo progetto, e inizió a sviluppare la versione GNU di Emacs che finirá nel 1985, anno nel quale pubblicherá un manifesto e fonderá la Free Software Foundation (FSF), associazione no-profit. La versione di Emacs venne distribuita con la licenza GNU General Public License (si veda la sectione relativa alle licenze), che verrá utilizzata per la maggior parte del software GNU. Nel 1986, modifica la licenza in modo da togliere ogni riferimento a un programma specifico e renderla il piu generale possibile. Sempre lo stesso anno pubblica la versione 1.0 di Emacs (utilizzando le versioni intere per i programmi stabili) e il GNU debugger. Nel 1987 finisce di sviluppare il primo progetto: Il GNU Compiler C, un compilatore per il linguaggio C, che qualcuno colpito dalla semplicita del codice lo studio e lo adatto per il C++. Mancava ancora peró un componente fondamentale, il Kernel GNU, ovvero il cuore di ogni sistema operativo, la parte che gestisce le periferiche, la memoria e le principali chiamate al microprocessore. Nel 1987 cominciarono a svilupparne uno, ma si puó dire che fu nel 1990 la vera data di inizio del kernel HURD!

HURD

4.2 Linus Torvalds e il kernel Linux

Linus Benedict Torvalds nacque il 28 dicembre del 1969 a Helsinki, in Finlandia. Il nonno era docente di statistica all'universitá di Helsinki, e fu lui che fece conoscere al nipote il suo primo computer, un Commodore Vic-20. Linus studió quel computer per cinque anni, quando decise di comprarne uno nuovo sia perché il suo era diventato vecchio sia perché ormai lo conosceva troppo bene.
Nel 1990 in uno dei viaggi per divulgare il progetto GNU, Stallman si recó al Politecnico di Helsinki in Finlandia. Tra il pubblico di quella assemblea, c'era Linus Torvalds, allora studente del'universitá vicina a Helsinki. Nel 1991 il grande progetto GNU aveva ancora bisogno di un kernel: infatti tutto il software GNU era costretto a girare su un kernel Unix, mentre la Free Software Foundation stava lavorando a Hurd.
Non avendo i soldi per comperare un kernel Unix, Linus decise che ne avrebbe sviluppado uno suo. Non si rendeva conto di quanto tempo gli sarebbe costato, ma questo fu un bene, in quanto se lo avesse saputo non si sarebbe nemmeno cimentato in un progetto simile. Inizió a studiare dal libro di Andrew Tannenbaum intitolato "Operating System: Design and Implementation", nel quale era allegato un software dimostrativo chiamato Minix, sviluppato dallo stesso autore.
Sul newsgroups dedicato a Minix, Tannenbaum riceveva ogni giorno richeste e modifiche, la maggior parte delle quali rifiutava. Lo scopo di minix era didattico e per questo non avrebbe dovuto espandersi piu di tanto: doveva rimanere semplice per consentire agli studenti di capirne il funzionamento.
Il 5 gennaio del 1991 Linus compró un PC, un 386, e mentre aspettava che gli arrivassero i dischi contenenti Minix, "studió" il computer giocando a Prince of Persia sul Dos.
Poiché in quel periodo stava studiando il Task Switching, sviluppó un kernel con due processi, uno scriveva A e l'altro B, il risultato era AAAABBBB. Adattó questi due processi alle sue esigenze di collegarsi con l'universitá e scaricare

la posta. Cosi un processo leggeva dalla tastiera e inviava al modem e l'altro leggeva dal modem e inviava allo schermo. Amplió tutto quando aggiunse un metodo per salvare file sul disco. Tutto questo era molto macchinoso, in quanto Linus doveva riavviare ogni volta la macchina per leggere le mail, per poi riavviarla per usare minix. Decise di ricreare tutte le caratteristiche di minix sul suo kernel. E cosi usó tutta l'estate del 91 per portare avanti il suo progetto. Quando la sua voce e il suo progetto inizió a farsi sentire sulla newsgroups di minix, Ari Lemmke gli regaló un'area dove poter rilasciare Linux. All'inizio il nome originale pensato da Torvalds era Buggix, che cambio poi in freax (free+freak+x), ma a Lemmke non piacquero e decise di usare Linux, sia perché suonava decisamente meglio degli altri, sia per fare omaggio al suo autore.
La licenza utilizzata prevedeva la distribuzione gratuita, la disponibilitá del codice sorgente, il mantenimento del copyright e il divieto di distribuirlo sotto pagamento, nemmeno per i costi tecnici. Questa licenza venne sostituita dalla GPL nella versione 0.12, una mossa decisiva che consentí la grande sua grande diffusione.
La sua pubblicazione[2] fu annunciata su comp.os.minix (il newsgroup riservato alle discussioni sul sistema operativo didattico creato da Tanenbaum) dove chiese un parere sul suo lavoro.
Il suo annuncio riscosse un grande successo tanto da far intervenire il professore in persona, che se pur contento di Linux, lo definiva obsoleto, in quanto la sua struttura monolitica non garantiva una oppurtuna portabilitá, mentre il kernel che la FSF andava sviluppando era l'ideale. Linus rispose dicendo che il suo sistema era decisamente meglio di quello del professore, e che se il kernel GNU fosse stato giá pronto, non avrebbe neanche iniziato a programmare Linux.
Uno dei punti di forza di Linus era di accettare i suggerimenti degli utenti e dare loro una risposta (o "feedback") il piú presto possibile, trattandoli tutti come se fossero esperti.
Fu cosí che quando qualcuno gli suggerí di implementare una memoria virtuale [3], lui lo fece, rilasciando nel Natale del 1992 la versione di 0.11+VM. Altro punto di forza era il fatto che piuttosto che modificare i programmi per farli girare su Linux, preferiva modificare il kernel, rendendolo sempre piu compatibile.
Entro 3 anni gli sviluppatori del kernel avevano integrato una buona parte del lavoro fatto dalla FSF, come il GCC e il GDB e altre utility BSD. Questo portó Torvalds a rilasciare il suo lavoro con una licenza GPL, per esprimere la propria lealtá al movimento del software libero.
Il primo filesystem nativo di Linux si chiamava Xiafs, nato da Minixfs. Poi subentro exct, ma al primo crash del sistema tutti i dati andavano persi, fu cosí che nacque ext2, il quale non ricreava ogni volta il filesystem, ma lo aggiornava.
Un altro problema era dovuto proprio al famosissimo TCP/IP, in quanto la gestione dei protocolli presentava molti bachi. Fu Alan Cox a riparare a questa falla, non avendo i fondi a disposizone per comperare un coprocessore matematico per far girare Unix, scaricó Linux, il quale emulava quella componente. E cosí decise di dotarlo di un Network Layer decente.

[2] allegato 3
[3] Virtual Memory, ossia gestire una parte di hard disk come se fosse Ram

4.3 X Window

Microsoft Windows e Apple Machintos avevano sviluppato un nuovo approccio all'uso del computer attraverso un interfaccia grafica, chiamata anche GUI (grafical user interface). Usciva la versione 3.1 di Windows e Linux offriva come unica interfaccia con l'utente una shell, ossia una riga di comando dove digitare i comandi che il kernel poteva interpretare.
Il sistema operativo Unix offriva un base chiamata X, che consentiva di aprire piú finestre e controllare cosí piú programmi simultaneamente.
Nel 1992 nessuno aveva pensato di sviluppare una simile interfaccia, nemmeno Linus il quale era troppo impegnato a sviluppare il kernel, fu cosí che nacque un gruppo chiamato Xfree86 (gioco di parole derivato da X386, la versione di X per i chip intel 80386). Questo gruppo distribuiva gratuitamente le correzione di X per le versioni Unix, ma quando entró in contatto con il progetto Linux decise di seguirlo nella sua filosofia. Tra i due gruppi nasceva una sinergia, Xfree86 portó nuovi utenti a Linux, e viceversa, Linux diede una piattaforma libera dove sviluppare il server X.

4.4 Gnome e KDE

4.4.1 Gimp e il toolkit Gtk

Anche se con un server X si potevano avere piú finestre, mancavano ancora le librerie che offrivano i vari widget, ossia tutti quelli elementi come bottoni, elenchi a tendina, caselle di testo, ecc...
Questa grave mancanza era dovuta al fatto che Linux fu costruito da hackers e non era di certo rivolto a utenti inesperti.
Inoltre la gran parte dei programmi erano rivolti a sviluppatori e amministratori di rete, mancavano quindi i programmi per il resto degli utenti.
Spencer Kimball e Peter Mattis si erano interessati alle evoluzioni del World Wide Web con particolare attenzione alla grafica. Il programma piú utilizzato dai Web designer era ed é Photoshop di Adobe (per l'elaborazione di immagini).
Vollero ricreare un programma simile anche per Linux, il nome del progetto si chiamó GIMP (General Image Manipulation Program oppure GNU Image Manipulation Programm).
Decisero di sviluppare la loro applicazione con Motif, il toolkit grafico principale di Unix. Questa libreria non era né open source, né gratuita e i due studenti faticarono molto per arrivare a pubblicare Gimp 0.54 nel febbraio del 1996.
I due studenti si divisero i compiti, Spencer continuó a lavorare sulla manipolazione di immagini mentre Mattis inizió a sviluppare una libreria chiamata Gtk (Gimp Tool kit).
Mentre con Motif molte cose non si riuscivano a fare, o perché non esistevano o semplicemente perché non si sapeva come fare, con la nuova libreria avrebbero eliminato questi due problemi, se non esisteva lo si poteva costruire e se Spences non sapeva come fare, gli bastava chiedere al compagno.
Quando il programma venne rilasciato assieme alle nuove librerie Open Source ebbero l'appoggio della comunitá, che inizió a inviare correzioni e feedback.

4.4.2 Kde e il toolkit Qt

Ettrich sviluppó LyX, che era un formattatore di testi basato sul linguaggio di composizione di Tex [4] ideato da Donald Knuth. LyX consentiva di scrivere documenti senza preoccuparsi di impaginazione, margini, posizione delle immagini e tutti gli altri aspetti della formattazione. LyX utilizzava dapprima le librerie Motif, per poi passare a una libreria gratuita, XForm, ma non libera secondo la definizione della Free Software Foundation. Ettrich si accorse che la grafica proposta dal suo programma era pessima, quando la sua ragazza lo utilizzó per redigere la sua tesi. Decise a questo punto di cercare una nuova libreria e creare un ambiente simile a Windows. Il Toolkit che utilizzó si chiamava Qt di un'azienda norvegiese chiamata TrollTech. Anche Qt non era free software, ma era distribuito gratuitamente a scopi non commerciali, non era possibile peró vedere il sorgente. Ettrich annunció il suo progetto KDE (K Desktop Environment) nell'ottobre del 1996 assieme ad un'analisi di come sviluppare le intefacce grafiche. L'assenza di una base free spaccó in due gli utenti Linux, dove i piú fedeli alla filosofia di Stallman rinunciarono a contribuire e utilizzare Kde. L'azienda norvegiese si rifiutó di utilizzare una licenza come la GPL per due motivi: in primo luogo molti programmatori vedevano a rischio il loro posto di lavoro e minacciarono di licenziarsi, in secondo luogo vi era la paura di un possibile forking, cioé che il progetto potesse venir sviluppato piú velocemente dalla comunitá anche in direzioni contrarie a quelle fissate dall'azienda e causare una biforcazione del progetto. Cercó comunque di venire incontro ai suoi utenti, i quali temevano che la licenza di Qt potesse cambiare e diventare del tutto proprietaria da un giorno all'altro. Trolltech istitui la kde Free Qt Foundation che assicurava una versione free di Qt anche in caso di cessione o vendita della societá.

4.4.3 Gnome

Miguel de Icaza Amozurrita, non essendo riuscito a convicere Trolltech ad utilizzare una licenza Free, decise di dar vita ad un progetto concorrente GNOME (GNU Network Object Model Environment) basato su il toolkit di Mattis, Gtk. Gnome era concepito per essere un insieme di tanti piccoli programmi, piú facili da gestire, piú adatti ad una metodologia di sviluppo open source e migliori per il debug. L'annuncio del progetto avvenne nel 1997 e molti si schierarono a favore di questo progetto. Perfino Red Hat, una societá tra le piú prestigiose nelle distribuzioni Linux, era interessata a questo progetto, in quanto la sua filosofia consisteva nel distribuire solo software Open source ed aveva iniziato a perdere importanza quando altre distribuzioni adottarano kde.

4.5 Le distribuzioni

Una distribuzione si preoccupa di fornire un sistema operativo completo della maggior parte di programmi necessari, di selezionarne solo quelli con una certa

[4] Il nome Tex deriva dal greco tekhne che significa arte. Il suo autore Donald Knuth sosteneva l'idea che programmare fosse un'arte, al pari di comporre una canzone o una poesia, e che la riconpensa fossa la soddisfazione avuta dalla sua creazione. Knuth, nato a Milwaukee il 1938, é famoso anche per la sua opera in volumi "The Art of Computer programming", il primo uscí nel 1969

qualitá, inoltre offre una facile installazione anche per i meno esperti, molte distribuzioni offrono anche l'aggiornamento on-line. Quello che rende diverso questo settore é il fatto che chiunque puó rilasciare una sua distribuzione modificandone un'altra, garantendo cosí che nessuno possa detenere il monopolio.

4.5.1 Le prime distribuzioni

All'inizio Linux venne distribuito in due dischetti, uno chiamato boot, contenente il kernel e uno chiamato root, contente il filesystem. Quando si accendeva il computer si inseriva il primo dischetto, che dopo aver caricato in memoria il kernel chiedeva il secondo dischetto, a questo punto si poteva comiciare ad usare il pc.
Torvald rilasció il codice di questi due dischetti sul server di Helsinky e altri siti nel mondo li ridistribuirono a loro volta. Il sito del Manchester Computer Center, era uno di questi, ma nel febbraio del 1992 rilasció una sua variante che cercava di semplificare l'installazione di Linux. Dopo la distribuzione MCC ne nacquero altre come la TAMU e la SLS di Peter MacDonald che quando venne rilasciata includeva il kernel Linux, delle utility di base, X windows e l'implementazione del protocollo TCP/IP. Quest'ultima divenne la piú diffusa, ma man mano che le applicazioni disponibili aumentavano e il kernel diventava sempre piú complesso e la stabilitá della distribuzione veniva compromessa.

4.5.2 Debian

Il Progetto Debian[5] fu fondato ufficialmente il 16 agosto del 1993 da Ian Murdock, allora uno studente universitario alla Purdue University.
Il suo annuncio su un forum riscosse tantissime risposte, dando un'impulso notevole al progetto.
Ian si lamentava del fatto che la SLS fosse cosí poco stabile e volle creare una distribuzione basata sul modello di sviluppo utilizzato da Torvalds ossia sviluppata apertamente, libera, nello spirito di Linux e di GNU.
L'organizzazione era meritocratica e aveva Ian come capo, e dei luogotenenti che erano resposabili di determinate aree. Ogni decisione accendeva un dibattito e sulla base di questo dibattito Ian avrebbe espresso il suo parere, proprio come avveniva con Linux.
Perfino Richard Stallmand replicó a quell'annuncio, dicendo che la Free Software Foundation si stava interessando sempre di piú al progetto Linux, visto che il kerlen GNU richiedeva piú tempo del previsto, e che avrebbero finanziato Debian. Il finanziamento, che duró un anno a partire dal novembre 1994, poneva tra le questioni idealiste di Stallman e gli utenti, Ian Murdock che doveva decidere da che parte stare. Stallman voleva sempre di piú avere il controllo, mentre gli utenti rifiutavano l'idea di avere un capo , voleva cambiare il nome e utilizzare GNU/Linux o LiGNUx. Per molti questa proposta non era giusta. Sebbene molti dei programmi erano del progetto GNU altri appartenevano per esempio al gruppo Xfree86. Linus disse che non volle intromettersi in questa faccenda, ma che, secondo lui, era necessario un nome accattivante come Linux, e poi distribuire equamente i meriti tra i vari sostenitori. Nel marzo del 1996 Ian si dimesse sia per la nuova disputa nata sia per altri motivi quali la famiglia

[5]il nome Debian é dato dalle iniziali della moglie Debora e Ian.

e la decisione di conseguire una laurea. Il suo successore fu Bruce Perence che si schieró dalla parte degli utenti, dopo aver fatto due conti, ringrazió Stallman del finanziamento e disse che avrebbero potuto autofinanziarsi grazie alle donazioni.

4.5.3 Slackware

Tutto cominció nel 1992, quando Patrick Volkerding usó Linux perché gli serviva un interprete lisp[6] per un suo progetto. La distribuzione che usava si chiamava SLS Linux della Soft Landing System.

Tuttavia presentava molti errori, cosí Patrick inizió a fissarli e in seguito decise di rilasciare una distribuzione per sé e i suoi amici, che guadagnó rapidamente popolaritá. Patrick la rese disponibile al publico sotto il nome di Slackware.
Venne in seguito distribuita da Walnut Creek su CD-Rom, cosa che diede accesso a Linux a molte persone che prima non avrebbero potuto.

4.5.4 Yggdrasil

Il nome di questa distribuzione é tratto dalla mitologia dell'antica Scandinavia e significa "albero del mondo". Nel 1992 veniva rilasciata la prima release alpha, denominata LGX, acronomi dei tre componenti principali: Linux, GNU e X.
Il 18 febbraio del 1993, il fondatore Adam Richter, annunciava la prima versione di Yggdrasil su CD-Rom.
Questa versione comprendeva strumenti per il multimedia, X-Windows, un'applicazione per l'installazione e la possibilitá di essere lanciata direttamente dal CD. Una cosa rese particolare questa distribuzione, il fatto che includesse Motif e che per ogni sua vendita 5 dollari avrebbero finanziato lo sviluppo di un suo clone free. Certo, per molti mescolare Free Software con prodotti proprietari rappresentava come un tradimento da parte di Yggdrasil, ma d'altro canto questo permise di rendere piú accattivante Linux.

4.5.5 Red Hat

Fu fondata da Marc Ewing nel 1993, a quel tempo utilizzava la SLS per sviluppare un suo progetto, ma si accorse che perdeva piú tempo a mantenere il sistema che non il suo programma. Decise che "ció che il mondo aveva veramente necessitá era una distribuzione Linux che fosse davvero buona".

Ai tempi del college Marc indossava un cappello bianco e rosso, appartenuto a suo nonno, che durante l'ultimo anno smarrí. Utilizzó quel cappello per il nome della sua distribuzione chiamandola Cappello rosso, concedendosi la libertá di togliere il bianco, visto che non suonava bene. Ewing pensó di dotare la sua distribuzione di un sistema per l'aggiornamento: nessuno prima di allora ci aveva mai pensato. Questa distribuzione voleva orientarsi anche verso gli utenti meno esperti, visto che Linux stava diventando popolare, e cercó di facilitare

[6] Il Lisp (List Processor) é un linguaggio di programmazione interpretato, spesso usato nei progetti di intelligenza artificiale. É stato ideato nel 1958 da John McCarthy. La Symbolics Technology Inc. ha realizzato negli anni 80 delle workstation con sistema operativo interamente programmato in LISP. Le prime LISPM (LISP Machine) erano state implementate al MIT.

l'installazione, la configurazione e il package management[7]. La filosofia perseguita credeva fermamente nel movimento Free Software, includendo cosi solo prodotti con licenze GPL. Red Hat si sente fortemente obbligata verso la comunitá e ricambia finanziando la Free Software Foundation, il Linux Documentation Project, il gruppo Xfree86 e molti altri. Red Hat fú anche la prima che trovó un nuovo mercato nato attorno a Linux, ossia l'assistenza. Bob Young nel 1993 gestiva una piccola societá di distribuzione software Unix chiamata ACC Bookstore. Scoprí che le vendite di prodotti Open Source crescevano sempre di piú rispetto a prodotti proprietari. Gli servivano nuovi prodotti da mettere sul suo catalogo e alcuni clienti gli consigliarono Red Hat. Si creó un intesa perfetta e dopo un paio di mesi le due societá si fusero in Red Hat Software. L'11 agosto del 1999 Red Hat venne quotata in borsa, apri a 14 dollari e chiuse a 52, ottenendo cosi un valore di mercato di 35 miliardi di dollari.

4.5.6 Caldera

Caldera [8] era un progetto nato all'interno di Novell, allora azienda leader nel campo del networking e proprietaria di Unix. A capo del progetto, prima chiamato Exposé poi Corsair, c'era Bryan Sparks. Novell cambió presidente, il quale mise il progetto da parte. Sparks chiese al vecchio presidente di finanziare il progetto che avrebbe portato avanti. Nell'ottobre del 1994 Caldera venne rilasciata come versione beta, per arrivare alla versione definitiva nel 1996. Caldera raccoglieva parecchie applicazioni rivolte alla gestione aziendale su un Cd a parte tra cui anche un porting di un database famoso, Adabas D della AG Software.

4.5.7 Pacific HiTech

Cliff Miller era nato a San Francisco, ma dopo aver conseguito una laurea in linguistica, si trasferi in Giappone dove insegnó inglese. Poi si trasferi in Cina dove lavoró in un Universitá e dove conobbe Iris, sua moglie, la quale parlava bene sia inglese che giapponese. Cliff tornó negli Stati Uniti e lavoró presso la Xenon nel gruppo dedicato al multilingue. Dopo varie richieste, ottenne dall'universitá dello Utah una borsa di studio per specializzarsi in informatica. La moglie venne licenziata in seguito all'acquisizione della societa presso la quale lavorava da parte della Novell. Miller e Iris fondarono una societá che commercializzava software occidentale e americano sul mercato giapponese. Si imbatterono cosi in Linux, e decisero di vendere una loro distribuzione per il mercato giapponese. La loro distribuzione conteneva peró software proprietario per garantire una certa qualitá nella traduzione, ma ne rilasciarono anche una completamente free. Ottennero un grande successo, e secondo Miller la loro distribuzione copriva il 50 % della popolazione. Cambiarono il nome della distribuzione che divenne TurboLinux e decisero inoltre, nel 1999, di concentrare gli sforzi ed entrare anche nel mercato americano. Ci riuscirono sviluppando una linea di prodotti server e la tecnologia cluster[9]

[7] I sistemi di package management si occupano di installare i programmi su un sistema operativo e di solito tengono traccia automaticamente delle dipendenze tra il programma e le cosidette librerie dinamiche (DLL). Le librerie dinamiche sono caricate dal sistema operativo e offrono funzionalitá di base.

[8] il nome deriva da un cratere di un vulcano sull'orlo dell'inattivita

[9] Questa tecnologia permette di sfruttare tutta la potenza di calcolo di piú macchine collegate, gestendo anche la possibilita che una macchina possa cedere. Consente di avere le stesse

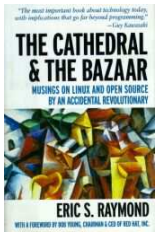
4.5.8 SUSE

Software und System Entwicklung, ossia sviluppo di software e di sistema.
Roland Dyroff, assieme a degli amici, volevano offrire assistenza in ambito Unix, ma Linux bussó alle loro porte prima che il progetto si avviasse. Offrirono cosi la SLS nel 1992, e poi passarono a Slackware per poi uscire con una loro distribuzione nel 1996. In Europa esistevano parecchi tecnici Linux, ma le versioni Slackware spesso uscivano con ritardo e bisognava applicare delle path per renderlo utilizzabile ad un utenza tedesca. Grazie a questi fattori conquistarono il mercato europeo, e tuttora rimane la piú diffusa in Europa.

prestazione che avrebbe un supercomputer, semplicemente collegando piu computer tra loro, abbassando cosi notevolmente i costi.

Chapter 5

Open Source e Free Software



Il libro "la cattedrale e il bazar"

Il termine Open source (codice aperto) venne usato da Eric S. Raymond nel 1997 in un suo documento intitolato "The Cahedral and the Bazaar" dove illustrava due modelli diversi di sviluppo del software, quello tradizionale e quello basato sulla libera circolazione del codice.
Allora si cercava un termine per invogliare le aziende a prendere in considerazione il free software, che per molti rimaneva sinonimo di gratuito.
Anche se la Free software Foundation aveva giá ribadito piú volte che free significava libero, con motti come *"free as free speech, not free beer"* (Libero come la libertá di parola non come una birra gratis). Fu nel 1998 che Bruce Perens e Eric S. Raymond fondarono l'Open Source Initiative (Osi), con lo scopo principale di definire cosa fosse l'Open Source.
L'Open Source Definition é arrivata oggi alla versione 1.9 e prevede 9 comandamenti:

Il logo del movimento Open Source

1. Ridistribuzione libera e gratuita cioé é vietato imporre limitazioni sulla vendita, inoltre non si possono prevedere royalties o altri pagamenti, evitando cosi la tentazione di guadagni a breve termine e di abbandono del progetto.
2. Il codice sorgente deve essere sempre reperibile gratuitamente, e in una forma che non sia inutilizzabile, aiutando cosi nella modifica del prodotto e la sua evoluzione
3. La licenza deve permettere le modifiche e lo sviluppo di prodotti derivati, le quali devono mantenere la stessa licenza
4. Ci deve essere integritá del codice sorgente dell'autore, il quale puó richiedere che le versioni future abbiano un nome diverso, questo per garantire l'autenticitá di un prodotto.
5. Non devono esserci discriminazioni contro gruppi o persone
6. Non devono esserci discriminazioni per il suo utilizzo
7. I diritti allegati a un programma vengono distribuiti a tutti coloro ai quali il programma é a sua volta redistribuito

8. Il prodotto deve conservare la sua licenza, sia che esso venga incorporato in un altro prodotto, sia che esso ne venga estrapolato.

9. Non deve porre vincoli su altri software che siano su uno stesso supporto.

Mentre il movimento Free software ha come scopo il libero scambio di informazioni, considerando la libertá piu importante del denaro, l'Open Source mira a facilitare la collaborazione allo sviluppo per ottenere software migliore. Sebbene gli scopi siano diversi le strade che i due movimenti percorrono sono quasi uguali.
Se per Stallman quindi il software proprietario rappresenta una minaccia alla nostra libertá, per Raymond é solo un modello economico.
Stallman pubblicó un manifesto intitolato "software libero vs open source", dove esprime il suo timore che la libertá venga messa in secondo piano rispetto al vantaggio economico. Seppur il termine open source possa invogliare piú utenti verso il software libero, nulla toglie a loro di preferire un software proprietario, perché nessuno gli avrá insegnato la libertá.
Alcune distribuzioni GNU/Linux offrono software non libero, credendo che sia un vantaggio per gli utenti invece che un passo indietro per la libertá.

> Il movimento del Software Libero e quello Open Source sono come due partiti politici all'interno della nostra comunitá.
> Siamo in disaccordo sui principi di base, ma siamo d'accordo sulla maggior parte degli aspetti pratici.
> **Richard Stallman**

Chapter 6

Pro e contro

6.1 Economicitá

Sicuramente il vantaggio economico é un fattore che ha permesso una diffusione vasta del software libero. Tuttavia con la diffusione di programmi Peer to Peer, ottenere software proprietario gratuitamente non é piú un grosso problema e comunque sulla rete sono presenti molti programmi freeware.
La maggior parte dei virus sono stati creati nei paesi dell'est, anche se questo puó suonare strano é stato grazie all'ostacolo di programmi proprietari che hanno indotto molti a studiare a fondo l'informatica per raggirare le protezioni sui software. Sebbene sia illegale installare software pirata, per la maggior parte degli utenti questo non rappresenta un problema. Per le aziende il problema c'é, anche se in Trentino un terzo delle imprese non é in regola, e questo ha permesso un maggior interessamento a soluzioni libere.

6.2 Sicurezza

Microsoft afferma che non esiste nessuno che risponde di eventuali danni provacati da Linux, anche perché non esiste un suo proprietario. Questo é vero, ma anche la stessa Microsoft non si assume la responsabilitá di assicurare il corretto funzionamento del software. Al punto sette del contratto di licenza usato per i prodotti Microsofs si puó leggere:

> ESCLUSIONE DI TUTTI I DANNI.
> NEL LIMITE MASSIMO CONSENTITO DALLA LEGGE IN VIGORE, IN NESSUN CASO MICROSOFT O I SUOI FORNITORI SARANNO RESPONSABILI PER DANNI CONSEQUENZIALI, INCIDENTALI, DIRETTI, INDIRETTI, SPECIALI, MORALI O DI ALTRA NATURA (INCLUSI, IN VIA ESEMPLIFICATIVA, I DANNEGGIAMENTI ALLE PERSONE E ALLE PROPRIETÁ I DANNI PER PERDITA DI PROFITTI, PER INTERRUZIONE DI ATTIVITÁ PER PERDITA DI INFORMAZIONI, PER PERDITA DI PRIVACY, PER OMISSIONE DI RISPETTO DI OBBLIGHI INCLUSO QUELLO DI COMPORTARSI IN BUONA FEDE O CON RAGIONEVOLE DILIGENZA, PER NEGLIGENZA E PER

> QUALSIVOGLIA ALTRA PERDITA ECONOMICA O, PIÚ IN GENERALE, PER OGNI PERDITA) DERIVANTI DA O IN QUALSIASI MODO CONNESSI ALL' UTILIZZO O ALL' IMPOSSIBILITÁ DI UTILIZZARE IL PRODOTTO SOFTWARE, SIANO ESSI RISULTANTI DALL'ADEMPIMENTO DEL CONTRATTO, DA TORTO, NEGLIGENZA, RESPONSABILITÁ ASSOLUTA O ALTRE AZIONI LESIVE, ANCHE NEL CASO IN CUI MICROSOFT O I SUOI FORNITORI SIANO STATI INFORMATI DELLA POSSIBILITÁ DEL VERIFICARSI DI TALI DANNI. LA SUDDETTA ESCLUSIONE DI TUTTI I DANNI SARÁ EFFICACE ANCHE QUALORA VENGA MENO LO SCOPO ESSENZIALE DI QUALSIVOGLIA RIMEDIO.

Dunque bisogna fidarsi del software e di chi lo sviluppa. Ci sono due motivi per cui un'azienda potrebbe decidere di vendere software contenente errori. Il primo é che non lo sa, e questo é frutto dell'errore umano, il secondo sta nel fatto che all'uscita di una nuova versione potrá affermare di aver fissato i bug di quella precedente. In questo modo gli utenti comprerebbero la nuova versione, uguale alla precedente senza i vecchi errori, ma molto probabilmente con alcuni nuovi. Bisogna dire che questa tecnica se la possono permettere solo aziende che godono giá di una buona immagine e che il loro unico problema sta nel vendere una versione successiva. In una comunitá di sviluppo aperto si puó verificare solo il primo caso, in quanto, non avendo un diretto riscontro economico, andrebbe soltanto a sfavore del progetto. La possibilitá di generare errori umani non aumenta con l'aumentare degli sviluppatori che partecipano al progetto, ma cresce la possibilita di scovarli e ripararli aumenta se piú utenti hanno accesso al sorgente. Oltre che della sicurezza del programma, bisogna parlare anche della sicurezza dei dati. La possibilitá del codice sorgente rappresenta un'arma a doppio taglio: da un lato offre garanzia visto che eventuali debolezze del programma vengono segnalate e risolte, dall'altra queste debolezze potrebbere essere utilizzate per scopi negativi. Se si pensa a un programma di crittografia[1], l'autore potrebbe inserire delle cosi dette back door in modo da poter leggere i file crittografati. Se il software vennisse rilasciato completo di sorgente non sarebbe molto difficile scovare questi retroscena. La disponibilitá del sorgente ha inoltre permesso di testare l'affidabilitá. Bisogna dire che nessun software di crittografia é sicuro al cento per cento, ma se milioni di persone ne hanno studiato il codice e finora nessuno ha ancora trovato un'errore, o una "back door", si puó suppore che sia piú sicuro di uno con codice chiuso. D'altro canto quando Red Hat rilasció Piranha, un software rivolto alla gestione dei server, chi studió il sorgente del software scopri che gli sviluppatori avevano lasciato un'username ("piranha") e un password ("q") i quali permettevano l'accesso ai server e il loro controllo totale.
Alcune Pubbliche Amministrazioni stanno passando ad un modello Open Source, questo per alcuni motivi fondamentali. Il cittadino ha diritto alla trasparenza, ossia a come i suoi dati vengono trattati. Solo con un software libero si puó garantire la riservatezza dei dati e il loro utilizzo. Non deve essere obbligatorio usare software specifico e quindi occorre utilizzare formati liberi. Inoltre le nuove tecnologie, come il voto elettronico, necessitano di una sicurezza maggiore

[1]La crittografia comprende tutte le tecniche per rendere un messaggio "offuscato" in modo da non essere comprensibile a persone non autorizzate.

che l'Open Source puó fornire.

6.3 Qualitá

Un approccio Open Source permette uno sviluppo piú veloce, raggiungendo qualitá elevate in tempo piú breve di un approcio tipico. Questo é in parte vero, ma la qualitá di software dipende fortemente dalla abilitá del singolo sviluppatore. Esistono tuttavia prodotti commerciali che godono di molta esperienza, e che si sono definiti degli standard nel loro campo, come Photoshop per il fotoritocco e il web design. Nelle applicazioni server dominano comunque le soluzioni Open Source, ossia le combinazioni di Linux, Apache [2], MySql [3] e Php[4].

6.4 Libertá

Questo per molti rappresenta il vantaggio fondamentale per preferire software libero a quello proprietario.
Per l'utente normale non ha molta importanza aver accesso al codice sorgente, anche perché non capirebbe niente. Ma per un programmatore poter guardare il sorgente é un pó simile a quando da ragazzi si smonta e si rimonta ogni cosa. Per capire meglio l'importanza del software libero, basti pensare alla libertá di stampa. Anche se pochi di noi scriveranno un'articolo su un giornale é comunque importante garantire che chiunque possa esprimere il suo pensiero. L'arte della programmazione viene considerata come ogni altra arte artistica. Per arrivare a scrivere un buon libro bisogna leggere molti libri e fare molta pratica: per scrivere un buon programma bisogna leggere molti programmi, se il codice non fosse disponibile l'unico metodo che resterebbe sarebbe fare molto pratica.
Stallman ricorda che da bambini se si andava a scuola con delle caramelle si insegnava a condividerle con gl'altri: ora dovrebbero insegnare che é reato.

[2] Apache é un server web, ossia quell'applicazione che offre le risorse indicate del client
[3] MySql é un noto database
[4] Php é un linguaggio lato-server

Chapter 7

La trappola del software proprietario

Il software proprietario mette in trappola i suoi utenti, la tecnica per farlo é molto semplice! Si prenda un programma (ad esempio un word-processor) che esporti un file che é leggibile solo da se stesso, chiunque volesse vedere il tuo risultato deve acquistarlo e, cosa piú importante, il tuo file sará leggibile solo con versioni future. In questo modo la software-house ti obbliga a comprare una versione nuova ogni qual volta ne esca una!
Il software proprietario mette nel sacco la concorrenza, ovvero non riporta nessuna documentazione e anzi mina il suo formato in modo che risulti piú difficile capirne il funzionamento, utilizzando standard chiusi, e si preoccuperá di cambiare il formato della versione futura in modo tale che eventuali programmi di concorrenza non funzioni piu!
La legge tutela il software proprietario, se un programmatore volesse scrivere un suo programma sostitutivo a quello che avete pensato e riuscisse a capire il formato del file, sará accusato di Reverse-engineering, ostacolando cosi la libera concorrenza.
Su questo punto diverse aziende stanno cambiando approccio, condordando degli standard comuni e competendo piuttosto sulle soluzioni.
Lo standard permette alla concorrenza di sviluppare prodotti diversi sullo stesso tipo di dato. Un esempio sotto gli occhi di tutti é l'HTML: oggi esistono diversi browser che elaborano lo stesso tipo di dato! Un altro esempio é il TCP/IP, senza il quale saremmo ancora nel mezzo di una guerra tra formati che avrebbero impedito lo sviluppo di internet che abbiamo.
Questi due standard sono liberi, ossia non hanno proprietario, ma potrebbero esistere dei consorzi [1] che ne salvaguardano l'evoluzione, definendone le regole.
Un altro aspetto molto importante del software proprietario é che l'utente non sa quello che il programma fa, questo puó risultare piu o meno importante a seconda dei casi! Se desiderate scrivere un documento non vi importerá molto come funziona il vostro word processor, piuttosto il suo risultato, ma se volete effettuare una transazione tra la vostra banca e quella del vosto fornitore sarebbe meglio informarsi a priori se é affidabile! Non é detto che se milioni

[1] Per esempio il W3C definisce le regole per lo standart HTML, se si rispettano si é sicuri che il proprio codice sará accettato da tutti i browsers che le implementano.

di persone lo usano sia sicuro, potrebbe togliere una cifra quasi invisibile per ogni transazione senza che gli utenti se ne accorgano! Altri casi possono essere anche la raccolta di informazioni riservate all'utente, in questo caso sarebbe una violazione della privacy, ma nessuno se ne accorgerebbe!
Questo accadeva in un noto programma di fotoritocco, il quale comunicava segretamente alla polizia chiunque scannerizzasse banconote. Che questo sia un atto lecito o illecito per combattere la criminalitá non si sa, ma nessuno vuole comparire sugli elenchi della polizia solo per aver scannerizzato una banconota.

Chapter 8

Le licenze

> Una licenza é un insieme di leggi sul diritto d'autore che determina i comportamenti ammissibili a riguardo di un prodotto.

I prodotti Open source e Free software sono anch'essi accompagnati da una licenza a difesa non del valore commerciale, come lo posso essere licenze per prodotti proprietari, ma del valore morale, ossia il riconoscimento dei crediti morali per gli autori del prodotto. Esse infatti concedono diverse libertá, come la copia e la distribuzione, che in una licenza proprietaria vengono vietate.
Esistono tuttavia svariati tipi di licenze non proprietarie, differenti per alcuni criteri come la possibilitá di combinarsi con altri prodotti open source o con prodotti proprietari e il grado di protezione dei diritti di libertá.
L'Open Source Initiative (OSI) mantiene una lista aggiornata delle licenze ritenute conformi alla definizione ufficiale di Open Source.
Il termine copyleft venne coniato da Richard Stallman, ed é il contrario di copyright, mentre il secondo significa diritto d'autore, il primo é il permesso d'autore, che concede la libertá di modificare, redistribuire il prodotto con la stessa licenza originale. Il copyleft evita cosi che un prodotto open source possa diventare proprietario. Una limitazione della libertá o solo la sua salvaguardia?

8.1 Pubblico dominio

Il prodotto viene rilasciato senza alcuna licenza e l'autore rinuncia a tutti i diritti di proprieta intellettuale. La copia, la modifica e la distribuzione sono consentite, ma non si garantisce la preservazione della libertá. Non rientra dunque tra le licenze Open Source.

8.2 Shareware

Un prodotto shareware di solito é distribuito gratuitamente e ne consente l'utilizzo e la distribuzione, ma dopo un periodo di prova viene richiesta la registrazione e il suo pagamento. Non é obbligato a fornire il codice sorgente e l'utilizzo puó essere discriminato a secondo che venga usato per uso commerciale o meno

8.3 Freeware

Consente di utilizzare e distribuire gratuitamente il prodotto, ma non é disponibile il codice sorgente, inoltre dev'essere utilizzato per scopi privati e non commerciali

8.4 General Public License

Fu scritta dalla Free Software Foundation e usata nella gran parte dei progetti GNU. Assicura la libertá di utilizzo senza alcuna discriminazione, la modifica, la distribuzione e la disponibilitá del codice sorgente. Inoltre é una licenza copyleft per quanto conserva queste libertá anche nelle versioni successive. Non permette peró di essere distribuita assieme a ad altre licenze, oppure le altre parti cadranno sotto questa licenza. Questo tipo di licenza é detto virale, nel senso che puó espandere verso altri prodotti che cadano nella sua distribuzione. Rappresenta il maggior esponente delle licenze copyleft.

8.5 Lesser General Public Licenze

Mantiene le stesse caratteristiche della Gpl, ma evita il comportamento virale, in quanto alcune imprese accusavano che fosse inapplicabile in quanto antieconomica. La Lgpl consente infatti che librerie coperte da questa licenze possano venir utilizzate da prodotti con licenze differenti.

8.6 Berkeley Software Distribution

La redistribuzione, la modifica e l'uso in forma sorgente o binaria sono consentite, purché si conservi questa licenza. La sua principale caratteristica consiste nel fatto che permette di coprire con un' altra licenza le modifiche apportate.

Licenze	Non Open Source			Open Source		
	Public Domain	Freeware	Shareware	GPL	LGPL	BSD
Gratuitá del prodotto	tempo limitato	si	si	si	si	si
Distribuzione libera	si	si	si	si	si	si
Utilizzo libero	si	limitato	limitato	si	si	si
Disponibilitá del sorgente	non obbligatorio	no	no	si	si	si
Modifica libera	si	no	no	si	si	si
Mixabilitá con software proprietario	si	no	no	no	si	si
Protezione della liberta o copyleft	no	limitata	limitata	si	si	si
Privilegi o discriminazioni	no	si	si	no	no	no

Licenze	Progetti	Percentuale
GNU General Public License (GPL)	26,157	68.02
GNU Lesser General Public License (LGPL)	2,258	5.87
BSD License (original)	1,372	3.57
Freeware	976	2.54
Freely Distributable	962	2.50
BSD License (revised)	735	1.91
Artistic License	681	1.77
Other/Proprietary License with Free Trial	574	1.49
Free for non-commercial use	565	1.47
Free To Use But Restricted	495	1.29
MIT/X Consortium License	480	1.25
Public Domain	476	1.24
Other/Proprietary License	402	1.05
OSI Approved	389	1.01
Shareware	310	0.81
Other/Proprietary License with Source	293	0.76
The Apache License	278	0.72
Mozilla Public License (MPL)	223	0.58
Perl License	150	0.39
The Apache License 2.0	92	0.24
Common Public License	78	0.20
GNU Free Documentation License (FDL)	56	0.15
Python License	55	0.14
Open Software License	53	0.14
The PHP License	46	0.12
zlib/libpng License	44	0.11
Q Public License (QPL)	41	0.11
Free For Educational Use	32	0.08
Academic Free License (AFL)	22	0.06
Eiffel Forum License (EFL)	21	0.05
GNAT Modified GPL (GMGPL)	20	0.05
Free For Home Use	19	0.05
DFSG approved	15	0.04
IBM Public License	12	0.03
Zope Public License (ZPL)	11	0.03
Affero General Public License	10	0.03
SUN Binary Code License	9	0.02
The Clarified Artistic License	9	0.02
SUN Public License	8	0.02
W3C License	7	0.02
Aladdin Free Public License (AFPL)	5	0.01
SUN Community Source License	5	0.01
WTFPL	2	0.01
The Latex Project Public License (LPPL)	2	0.01
Voxel Public License (VPL)	2	0.01
Eclipse Public License	1	0.00
Guile license	1	0.00
Netscape Public License (NPL)	1	0.00

fonte: www.freshmeat.net

Chapter 9

Organizzazione

Organizzare un progetto Open Source non é compito facile, bisogna innanzitutto offrire giá una base di partenza a cui gli utenti possano interessarsi e dare dei feedback. Occorre mantenere la fiducia degli utenti e sopratutto avere un buon carisma. Erick S. Raymond nella sua analisi "La cattedrale i il bazar" spiegava il successo del suo primo programma Open Source e ricavava dalla sua esperienza alcune regole.

9.1 Lancio

> Ogni buon lavoro software inizia dalla frenesia personale di uno sviluppatore [**Raymond**]

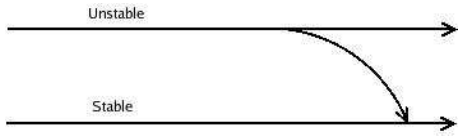
Lo sviluppo Open Source pone le sua fondamenta sulla passione di sviluppare: un prodotto riuscirá meglio se l'autore ci mettera passione. Questo fattore é molto importante perché distingue notevolmente lo sviluppo tradizionale, il cui interesse maggiore é quello di percepire la propria paga. Torvald, come Stallman o Knuth e molti altri, offrí giá una buona base di partenza per il loro progetto, il quale attiró sempre piú utenti costruendo cosí una comunitá. Il carisma dell'autore é parte fondamentale del lancio e dello sviluppo di un prodotto. La maggior parte degli annunci erano fatti su forum, dove ci si identificava,si indicava il progetto e sostanzialmente si chiedeva aiuto. Ora sono nati dei siti come SourceForge.net o FreshMeat.net, che raccolgono e gestiscono questi progetti, mettono a disposizione i loro server, offrono forum, chat, newsletter con il solo scopo di migliorare la comunicazione e ampliare la comunitá.

9.2 Espansione

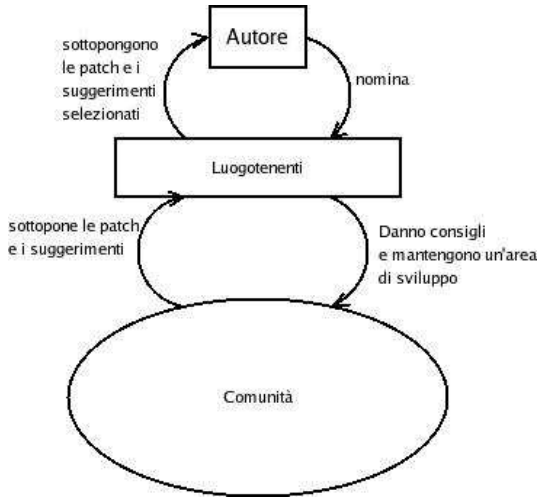
> Distribuisci presto. Distribuisci speso. E presta ascolto agli utenti. [**Raymond**]

Gli utenti si possono rivelare degli utili sviluppatori, dapprima riportando errori, poi correggendoli e inviando patch. Infatti il punto di forza dell'Open Source sono proprio gli utenti: al loro aumentare, aumenterá la velocita di sviluppo. La dimensione del kernel Linux ha assunto un'andamento esponenziale, nel 1991

occupava 63,36 byte, un anno dopo 273, nel 1993 ne occupava 711 e questo é dovuto all'aumento della sua popolaritá. Le release avvenivano anche 2 volte la settimana e questo portó confusione tra gli utenti i quali non sapevano quale versione funzionava meglio.
Si assunse cosi la convenzione che le release pari fossero stabili, mentre quelle dispari fossero in fase di sviluppo. In modo tale che gli audaci avrebbero potuto lavorare sulle unstable e gli utenti che necessitavano di sicurezza sulle release stable.
Lo sviluppo proseguiva cosi su due linee differenti.



Puó succedere che l'autore non riesca a continuare il suo progetto e nello stesso tempo soddisfare le richieste degli utenti. Potrebbe quindi ignorare alcuni suggerimenti causando cosí la possibilita di un code-forking, ossia la biforcazione del progetto. Questo evento ha un effetto negativo e uno positivo. Un utente potrebbe decidere di dare vita a una biforcazione del progetto in seguito al rifiuto da parte dell'autore di una sua idea, spaccando cosí in due la comunitá e dimezzando le risorse. L'aspetto positivo é che quell'idea potrebbe rivelarsi vincente.
Per un progetto di grandi dimensioni é opportuno dividere il lavoro e nominare quindi dei luogotenenti che siano responsabili di determinate aree. La nomina da parte dell'autore si basa sopratutto sulla meritocrazia, ossia in base al contributo e all'esperienza. I luogotenenti costituiscono un filtro tra la comunitá e l'autore, oppure possono gestire un'area del progetto indipendentemente.



Chapter 10

Modelli economici

Il metodo tradizionale prevede di vendere un permesso esclusivo ad un'utente di utilizzare un prodotto. Questo sistema basato sulla vendita della licenza non é compatibile con software open source. Questo metodo non rappresenta l'unica soluzione economica, anche se é la piú intuitiva e consente ricavi nel breve periodo, ma é noto che in certi casi questa soluzione possa portare al crollo del vendite. Il software é un bene molto diverso dagli altri presenti sul mercato. Ci sono alcune considerazioni che bisogna tenere presente. Il processo produttivo del software prevede lunghi periodi per fabbricare il primo esemplare e tempi quasi nulli per i successivi. E' un bene che non si usura e che non necessita di manutenzione, ma di assistenza.

10.1 Il caso Apache: suddivisione dei costi

[Eric S. Raymond]
Questa soluzione si addice a prodotti che non sono destinati alla vendita, bensí all' utilizzo, comportando quindi solo dei costi per il suo sviluppo. Fu adottata dal gruppo Apache per sviluppare un web-server. Anche se il software non avrebbe portato ricavi direttamente, ha invece abbassato notevolmente i costi del suo sviluppo. Un web server é un'applicazione che gestisce le richieste di pagine web tramite il protocollo HTTP. Agli albori di Internet esistevano gia 2 grandi web-server, uno era della Microsoft e l'altro della Netscape. Rob McCool, che era uno studente presso l'Universitá dell'Illinois, sviluppó presso L'NCSA (Nationale Center of Supercomputing Application) un server web, e lo distribuí gratuitamente. Era un web server molto valido tanto che molti lo preferivano ai suoi concorrenti, ma a causa della diminuzione dei programmatori del centro, gli utenti che inviavano estensioni o correzioni di errori non ricevevano piu risposte. Behlendorf si offrí di creare una comunitá che portasse avanti il progetto, una comunitá basata sulla meritocrazia, dove chi si impegnava di piú e mostrava doti maggiori di altri, acquistava ruoli piú importanti. Questa nuova comunita si chiamó Apache e rilasciava il web server Apache 0.6.2 nel 1995, con una licenza Open Source. La versione 1.0, rilasciata nel dicembre del 95 era diventata la piu usata su Internet, e nel 2000 superó le rivali Netscape e Microsoft. Il server Apache ha contribuito all'integritá della rete: infatti, se non avesse avuto questo successo, i rivali avrebbero sviluppando due architetture diverse e incompatibili

costringendo gli utenti a scegliere o per l'uno o per l'altro, e il World Wide Web si sarebbe spaccato in due.

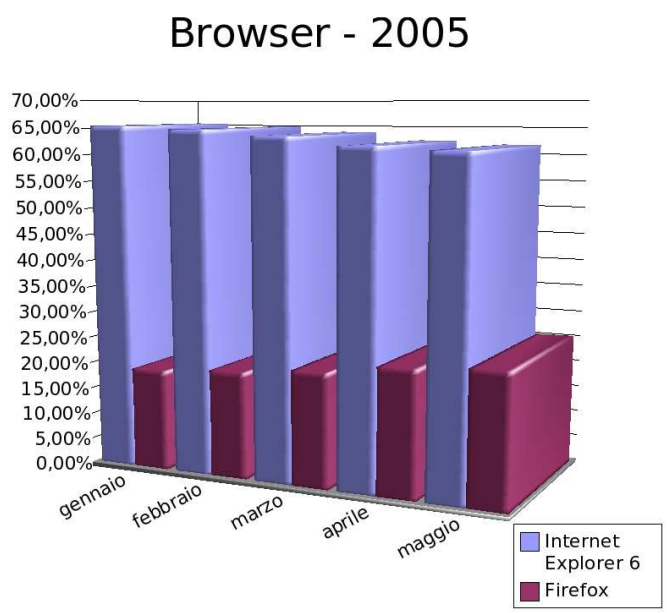
10.2 Il caso Cisco: suddivisione dei rischi

[Eric S. Raymond]
Puó capitare che un'azienda sviluppi del software non destinato alla vendita, ma al suo utilizzo interno. Quando gli sviluppatori di quel prodotto lasceranno l'azienda, non ci sará piú nessuno a mantenerlo, sará cosi destinato a perdere sincronia con il mondo reale. La soluzione di rilasciare il software garantisce la collaborazione della comunitá e il suo continuo sviluppo.

10.3 Articolo civetta/posizionatore sul mercato

[1] **[Eric S. Raymond]**
Questa strategia ha l'obiettivo di acquistare o mantenere quote di mercato di un prodotto rendendolo libero per favorire un suo complementare. E' il caso di Netscape, che adotto una soluzione open source per mantenersi sul mercato. Nel 1997 stava perdendo rapidamente quote di mercato in confronto a Internet Explorer di Microsoft, fu cosi che decise nel gennaio del 1998 di tentare un'altra strada,l'open source. Annunció che avrebbe rese pubblico il codice sorgente del proprio browser.
Alcuni componenti del browser non appartenevano alla Netscape, quindi le librerie delle aziende che non vollero pubblicare il proprio codice sorgente non vennero rilasciate.
Il nome del progetto si chiamó Mozilla e riscosse un notevole successo nella comunitá. La licenza con cui venne rilasciato fu la Mozilla Public License e venne accettata dalla comunitá visto che rispettava la definizione di open source.
Il suo rilascio consentí uno sviluppo e un debugging piu veloce, impedendo a Microsoft di istituire un monopolio sui browsers, per appropriarsi dello standard html e eliminare ogni possibile concorenza anche dal lato server. Non solo Netscape si salvó, ma continuó a guadagnare fette di mercato e tutt'ora é una piú che valida alternativa al browser Microsoft.
Il grafico sotto riportato mostra la crescita del browser Firefox, nato dal progetto Mozilla, e la graduale discesa di Internet Explorer.
I dati sono forniti dal sito www.w3school.com.

[1] Descritta anche come strategia ibrida di produttori di software nel libro "Open Source" di Moreno Muffatto e Matteo Faldini



Oracle, il famose database, rilasció i suoi sorgenti in seguito al successo ottenuto da Mysql,diretto rivale, e dovette cosí rivedere il proprio modello business in un modello compatibile con una soluzione open source.

10.4 Widget frosting

[Eric S. Raymond]
I produttori di hardware si sono ritrovati a sviluppare software per rendere utilizzabili i propri prodotti, ad esempio i device driver. Il software in questione non rappresenta fonte diretta di ricavi, bensí un costo che potrá essere abbassato passando a un modello open source. I produttori di hardware sono restii a pubblicare i propri codici sorgente, perché temono che la concorrenza possa copiarli.
Secondo Eric S. Raymond il plagio é una trappola, in quanto la concorrenza impiegherebbero una buona parte del ciclo produttivo per ottenere un prodotto che é gia presente sul mercato. Vista la velocitá con cui vengono proposti nuovi hardware sempre piú potenti sarebbe solo un spreco di risorse posizionare in ritardo un prodotto giá esistente.

10.5 Rivelare la ricetta, aprire un ristorante

[Eric S. Raymond] [2]

E' quello che succede con le distribuzioni Linux, societá come redhat, suse, etc.. vendono non il software bensí l'assemblaggio di un sistema operativo e altri servizi aggiuntivi come l'assistenza. Inoltre il rilasciare un prodotto comporta

[2]Descritta come strategia pure play nel libro "Open Source" di Moreno Muffatto e Matteo Faldini)

una notevole pubblicitá e i clienti si rivolgeranno piú ai suoi sviluppatori che alla concorrenza.
Nel paragone fatto da Raymond si considera che sebbene chiunque possa ottenere le ricette di cucina si preferisce il ristorante per i suoi servizi.

10.6 Fornire accessori

[Eric S. Raymond]
Si possono vendere accessori come magliette oppure manuali. La vendita di manuali permette ad un'azienda di pagara dei programmatori che sviluppino software per venderne la documentazione. La O'reilly Associates impiega molti sviluppatori noti nella comunitá open source per pubblicare ottimi libri e costruirsi una reputazione.

10.7 Liberare il futuro, vendere il presente

[Eric S. Raymond]
Si rilascia il prodotto in formato binario e codice sorgente con una licenza che permetta la libera circolazione ma vieti l'uso commerciale senza il pagamento di contributi. Si annuncia che dopo un certo periodo, per esempio sei mesi o un anno (vista la velocita di rinnovamento del software), il prodotto sará rilasciato sotto una licenza open source. Ció consente di ottenere dei vantaggi dalla comunitá open source e una buona pubblicitá.

10.8 Doppia licenza

In questo sistema si rilascia il software sotto 2 licenze. Una licenza copyleft per ottenere i vantaggi offerti dalla comunitá e un'altra che permette dietro pagamento di usare questo software in applicativi proprietari. Questo sistema viene utilizzado da aziende come Mysql per quanto riguarda l'omonimo database, e Trolltech per le librerie grafiche Qt.

Chapter 11

Glossario

Acronimo: é una parola formata con le iniziali delle parole di una frase o di una definizione.
Back door: Si potrebbero definire come dei passaggi segreti, se pensate che i vostri dati siano al sicuro
dentro il castello, dovete anche preoccuparti che non ci siano cunicoli sotteranei che minaccino li minaccino.
Basic: l'acronimo di Beginner's All purpose Symbolic Instruction Code e cio Codice di istruzioni simboliche ad uso generale per principianti. Questo linguaggio di programmazione era stato concepito negli anni 60 per essere di facile apprendimento agli studenti.
Codice eseguibile o compilato: é l'unico linguaggio che un calcolatore possa eseguire, é composto da sequenze di codice binario. E' incomprensibile per qualsiasi programmatore
Codice sorgente: é un insieme di istruzioni e dati, scritto in un linguaggio di programmazione. Rappresenta la versione originale del programma prima delle operazioni di traduzione verso linguaggi interpretabili dal calcolatore.
Compilatore: trasforma il codice sorgente in codice binario;
Debugger: programma che permette di trovare gli errori in un programma.
Freeware: un software che viene distribuito in modo gratuito.
Hacker: persona che si impegna nell'affrontare sfide intellettuali per aggirare o superare creativamente le limitazioni che le vengono imposte, in primo luogo nei suoi ambienti di interesse, che solitamente comprendono l'informatica o l'ingegneria elettronica.
Interprete: il lavoro che svolge un'interprete consiste nel tradurre ed eseguire riga per riga in codice macchina il sorgente passatogli.
Linguaggio di programmazione: rappresenta una via di mezzo tra il linguaggio umano e il linguaggio macchina! E'un insieme di regole che permette di scrivere correttamente una sequenza di istruzioni. I linguaggi di programmazione possono suddividersi in 3 livelli: il piu basso il linguaggio macchina é composto da sequenze di codice binario, poi viene il linguaggio assemblativo composto da brevi sequenze alfanumeriche che facilitano a il programmatore la comprensione,(esiste una corrispondenza biunivoca tra questi due linguaggi). Il 3 livello, quello dei linguaggi di alto livello, non hanno una corrispondenza 1 a 1 con il livello sottostante e permettono una maggiore comprensione del codice da parte del programmatore.

Appendix A

Allegati

A.1 Lettera aperta agli Hobbysti, di Bill Gates, 3 febbraio 1976

AN OPEN LETTER TO HOBBYISTS

By William Henry Gates III

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however, 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent on Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you dont́ do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write to me at 1180 Alvarado SE, # 114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates
General Partner, Micro-Soft

A.2 Annuncio del progetto GNU, di Richard M. Stallman 27 settembre 1983

From CSvax:pur-ee:inuxc!ixn5c!ihnp4!houxm!mhuxi!eagle!mit-vax!mit-eddie!RMS@MIT-OZ
From: RMS%MIT-OZ@mit-eddie
Newsgroups: net.unix-wizards,net.usoft
Subject: new UNIX implementation
Date: Tue, 27-Sep-83 12:35:59 EST
Organization: MIT AI Lab, Cambridge, MA

Free Unix!

Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU (for Gnu's Not Unix), and give it away free(1) to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed.

To begin with, GNU will be a kernel plus all the utilities needed to write and run C programs: editor, shell, C compiler, linker, assembler, and a few other things. After this we will add a text formatter, a YACC, an Empire game, a spreadsheet, and hundreds of other things. We hope to supply, eventually,

everything useful that normally comes with a Unix system, and anything else useful, including on-line and hardcopy documentation.

GNU will be able to run Unix programs, but will not be identical to Unix. We will make all improvements that are convenient, based on our experience with other operating systems. In particular, we plan to have longer filenames, file version numbers, a crashproof file system, filename completion perhaps, terminal-independent display support, and eventually a Lisp-based window system through which several Lisp programs and ordinary Unix programs can share a screen. Both C and Lisp will be available as system programming languages. We will have network software based on MIT's chaosnet protocol, far superior to UUCP. We may also have something compatible with UUCP.

Who Am I?

I am Richard Stallman, inventor of the original much-imitated EMACS editor, now at the Artificial Intelligence Lab at MIT. I have worked extensively on compilers, editors, debuggers, command interpreters, the Incompatible Timesharing System and the Lisp Machine operating system. I pioneered terminal-independent display support in ITS. In addition I have implemented one crashproof file system and two window systems for Lisp machines.

Why I Must Write GNU

I consider that the golden rule requires that if I like a program I must share it with other people who like it. I cannot in good conscience sign a nondisclosure agreement or a software license agreement.

So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.

How You Can Contribute

I am asking computer manufacturers for donations of machines and money. I'm asking individuals for donations of programs and work.

One computer manufacturer has already offered to provide a machine. But we could use more. One consequence you can expect if you donate machines is that GNU will run on them at an early date. The machine had better be able to operate in a residential area, and not require sophisticated cooling or power.

Individual programmers can contribute by writing a compatible duplicate of some Unix utility and giving it to me. For most projects, such part-time distributed work would be very hard to coordinate; the independently-written parts would not work together. But for the particular task of replacing Unix, this problem is absent. Most interface specifications are fixed by Unix compat-

ibility. If each contribution works with the rest of Unix, it will probably work with the rest of GNU.

If I get donations of money, I may be able to hire a few people full or part time. The salary won't be high, but I'm looking for people for whom knowing they are helping humanity is as important as money. I view this as a way of enabling dedicated people to devote their full energies to working on GNU by sparing them the need to make a living in another way.

For more information, contact me.
Arpanet mail:
RMS@MIT-MC.ARPA

Usenet:
...!mit-eddie!RMS@OZ
...!mit-vax!RMS@OZ

US Snail:
Richard Stallman
166 Prospect St
Cambridge, MA 02139

A.3 Annuncio di Linux, Linus Torwald 25 agosto 1991

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: ¡1991Aug25.205708.9541@klaava.Helsinki.FI¿
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

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 protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Bibliography

[1997] Titolo:La cattedrale e il bazar
Autore:Erik S. Raymond
Data:1997

[1999] Titolo:Il Calderone Magico
Autore:Erik S.Raymond
Data:1999

[1998] Titolo: Colonizzare la noosfera
Autore: Erik S.Raymond
Data: aprile 1998

[1998] Titolo: Trappola nel Cyberspazio
Autore: Roberto di Cosmo
Data: 20 marzo 1998

[2002] Titolo: Codice Libero Richard Stallman e la crociata per il software libero
Autore: Sam Williams
Data: 2002, traduzione italiana 2003

[1998] Titolo: Kriptonite
Autori: Joe Lametta
Data:Torino 1998
Casa Editrice: Nautilus
Home page: www.ecn.org/kriptonite

[1996] Titolo: Hackers gli eroi della rivoluzione informatica
Titolo originale: Hackers Heroes of informatic revolution
Autore: Steven Levy
Data: 1996
Casa Editrice:Shake Edizioni Underground

[] Titolo:Open Source strategia, organizzazione, prospettive
Autori:Moreno Muffato e Matteo Faldani

[2001] Titolo:Codice Ribelle La vera storia di Linux e della Rivoluzione Open Source
Titolo originale: Rebel Code Linux and the Open Source Revolution
Autore:Glyn Moody
Data:2001
Casa Editrice: Hops Libri
ISBN: 88-8378-033-7

[2004] Titolo: Il manuale del giovane Hacker II edizione- Tutto quello che avreste voluto sapere su Internet e nessuno vi ha mai detto
Titolo originale: Steal this computer book 3 - What they won't tell you about the internet
Autore: Wallace Wang
Casa editrice: Hops libri
Data: 2004

[] http://www.it.wikipedia.org

[] http://www.Linuxtrent.it

[] http://www.google.com

[] http://www.gnu.org

[] http://www.opensource.org

[] http://www.apogeonline.com/openpress

[] http://gnuwin.epfl.ch/articles/it/trapello/trappola.pdf

[] http://www.prosa.it/philosophy

[] http://www.softwarelibero.it/GNU/

[] http://softwarelibero.kuht.it

[] http://www.ecn.org/kriptonite

[] http://www.debian.org/

[] http://fedora.redhat.com/

[] http://www.gentoo.org/

[] http://www.slackware.com/

[] http://openskills.info/