

Linux Facile*

di Daniele Medri

30 giugno 2001

*© 2000, 2001 Daniele Medri - "Linux Facile" versione 5.0 - Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with one Front-Cover Texts: "Linux Facile - di Daniele Medri", and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License". (pag. 223)

A Monica, i miei genitori, i miei amici

Introduzione

Il successo del software libero viene decretato ogni anno che passa e la sua più famosa interpretazione è GNU/Linux, un sistema operativo molto performante, scalabile e sicuro, diffuso in tutto il mondo grazie alle proprie caratteristiche di qualità. Software *vivo*, liberamente usufruibile e distribuibile senza nessuna limitazione, un movimento che cresce e si diffonde tra le persone e il mondo professionale con una grande rapidità grazie alla rete Internet, la quale è stata ambiente incubatorio per lo stesso.

L'idea di questo manuale è sorta durante un corso di laboratorio universitario sulle tecnologie telematiche dove la contrapposizione tra le persone interessate e la dispersione delle informazioni era notevole e non certo positiva. Da questa situazione rilevata ho cercato di costruire un punto di partenza, un manuale *entry-level* sul sistema operativo e tutto il movimento storico e filosofico che lo ha preceduto. Questa è stata la direzione personalmente seguita nella stesura di questo testo, direzione sempre anteposta ad ogni scelta per agevolare utenti inesperti come quelli esperti, per insegnare novità o divenire semplice base informativa, *appunti di quotidiana amministrazione* di sistema.

Organizzazione degli argomenti

Il concetto di *percorso formativo* ha caratterizzato ogni versione del manuale e ha trovato la sua massima espressione applicativa in quella attuale. Gli argomenti presentati sono stati divisi per parti:

- La prima parte analizza i fattori *ambientali* che hanno spinto la crescita di GNU/Linux. L'obiettivo era quello di offrire una chiave di lettura integrativa alle solite nozioni tecniche.
- La seconda parte introduce GNU/Linux sul piano storico e su quello tecnico illustrando le componenti del sistema analizzando applicazioni, soluzioni per l'installazione e la gestione.
- La terza parte analizza gli ambienti grafici. Un'ampia introduzione sul Server *xFree86* e la descrizione degli ambienti desktop largamente diffusi e le applicazioni utili per essere produttivi e collegarsi alla rete.
- La quarta parte è stata pensata per il *networking*, ovvero l'utilizzo di un sistema GNU/Linux per soluzioni di rete o semplicemente per navigare su Internet tramite un modem.

Convenzioni

Esistono numerose distribuzioni GNU/Linux e in un panorama così ampio era necessario compiere delle scelte. Nel mio quotidiano operare utilizzo con gioia Debian GNU/Linux ma sono conscio del fatto che a livello commerciale sono maggiormente diffuse distribuzioni come SuSE, Red Hat o Mandrake. Sulla base di queste considerazioni ho impostato il manuale basandomi su Debian senza pur tralasciare le altre alternative esistenti.

Il prompt dei comandi Per la rappresentazione del prompt dei comandi si è genericamente scelto di utilizzare il simbolo:

#

per comandi da digitare come utente "root", mentre il semplice simbolo del dollaro per i comandi utilizzabili come utenti di sistema:

\$

Per la rappresentazione dei tasti si è scelto di borderarli come segue:

INVIO, CTRL, Alt

Note dell'autore

Questo manuale è stato sviluppato utilizzando LyX / L^AT_EX ed è liberamente distribuibile ed utilizzabile secondo le norme della licenza GNU Free Documentation License che tutela i diritti dell'opera e dell'autore. In caso di eventuali trasgressioni della licenza saranno presi i dovuti e necessari provvedimenti legali.

Ogni suggerimento o consiglio per migliorare l'opera in questione è ben accetto e può essere inviato per posta elettronica all'indirizzo:

Daniele Medri <madrid@linux.it>
homepage: <http://www.linux.it/~madrid/>

Sito web ufficiale del manuale:

<http://www.linuxfacile.org/>

Esistono siti mirror tra i quali segnalo:

- > <http://linux.interpunctonet.it/mirror/linuxfacile/>
- > <http://ada2.unipv.it/linuxfacile/>
- > <http://www.ziobudda.net/mirror/LinuxFacile/>
- > <http://mirror.sgala.com/LinuxFacile/>
- > <http://www.valtellinux.it/>
- > <http://www.linuxzine.it/linuxfacile/>
- > <http://parma.linux.it/linuxfacile/>

La lista aggiornata dei mirror è presente sul sito web del manuale.

Versioni precedenti

- 1.0 - 1 gennaio 2000.
 - Stampa a cura di Sgi Italia e autorizzata dall'autore.
- 1.2 - 27 marzo 2000.
 - Stampa a cura di Lumina S.A.S Milano (13.500 copie).
- 2.0 - 30 luglio 2000.
- 2.1 - 5 agosto 2000.
 - Stampa a cura di Lumina S.A.S Milano (13.500 copie).
- 3.0 - 12 gennaio 2001.
 - Stampa a cura dell'editore Systems e allegata alla rivista Inter.Net - Marzo 2001 (66.000 copie)
- 4.0 - 9 aprile 2001

Chiunque desideri stampare il manuale su larga scala (e non per il semplice uso personale) è tenuto a comunicare all'autore la propria intenzione sia come forma di gentilezza che per tutela personale a livello economico onde evitare l'uscita in coincidenza di altre edizioni.

Ringraziamenti

Una doverosa menzione di tutte le persone che ogni giorno contribuiscono a diffondere GNU/Linux nel mondo e in Italia particolarmente, persone che volontariamente spendono il proprio tempo libero per un impegno che è anche loro passione. Ringrazio particolarmente: Andrea Arcangeli, Luca *andrew* Andreucci, Maurizio *napo* Napolitano, Michel *ziobudda* Morelli, Gianluca Rubini, Dario BURNELLI, Alessandro *Alf* Forghieri, Giorgio Zarrelli, Maurizio *Tannoiser* Lemmo.

Hanno contribuito: Camillo Bongiovanni, Roberto A. Foglietta, Matteo *MatOfPing* Nastasi, Marco Cova, Federico Dalpane, Gilberto *velenux* Ficara, Gaetano Santoro. Un ringraziamento particolare a tutti i lettori che con le loro domande e i loro suggerimenti mi hanno aiutato a far crescere questo manuale.

Parte I.

Storia

1. Hackers

L'esponenziale crescita della rete ha cambiato l'identità della società evoluta o per lo meno quella "informatizzata". Un incredibile mole di dati circola sulle autostrade dell'informazione, dati privati, transazioni economiche e altro ancora. La vita delle persone si impossessa del mezzo e porta con sé ogni umana caratteristica e fobia, reale e non. Se nel mito popolare esisteva l'uomo nero oggi esiste un nuovo termine per *descrivere qualcuno che attenta ai diritti altrui*: hacker.

Avevo già opinioni in merito quando Filippo Bianchi, un giornalista di Repubblica, mi intervistò chiedendomi chiarimenti sulla terminologia a lui poco chiara e molto blasonata dai media che ruotava attorno al termine e in tale ottica riporto le mie considerazioni a titolo informativo.

1.1. Jargon: la nascita di un gergo

L'inizio della cultura hacker si può datare all'anno 1961, lo stesso anno in cui il MIT acquistò il primo calcolatore PDP-1. In questo contesto universitario il termine fu subito adottato dagli studenti del *Tech Model Railroad Club*, un gruppo di persone che si divertiva a costruire automatismi per gestire il traffico ferroviario per modellini di treni. Lo stesso club divenne in seguito il principale nucleo del laboratorio di Intelligenza Artificiale del MIT, gruppo di sviluppo delle principali tecnologie moderne informatiche.

Con lo sviluppo della rete di comunicazione ARPAnet, *il gergo hacker* (jargon) si diffuse nelle principali Università collegate. La terminologia utilizzata fu raccolta in un file da Raphael Finkel presso l'università di Stanford nel 1975. Un nuovo polo di sviluppo della cultura hacker fu il centro di ricerca di Palo Alto dove dalla fine degli anni '70 alla metà degli anni '80 vennero sviluppate un numero altissimo di nuove tecnologie come le interfacce visuali ad icone e finestre, le stampanti laser e tecnologie per le reti locali di computer. Il legame tra *hacker* e nuove scoperte tecnologiche è quindi strettissimo. Caratteristica comune per quelle persone che definiamo hacker è il raggiungimento di un obiettivo nel minor tempo possibile, una rivisitazione in chiave moderna di "il principio giustifica il mezzo" di Machiavelli.

Come viene riportato nel Jargon File:

“:hacker: n. [originally, someone who makes furniture with an axe] 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. 2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming. 3. A person capable of appreciating {hack value}. 4. A person who is good at programming quickly. 5. An expert at a particular program, or one who frequently does work using it or on it; as in 'a Unix hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.) 6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example. 7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations. 8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence 'password hacker', 'network hacker'. The correct term for this sense is {cracker}.”

1. Hackers

Una persona che si diverte esplorando i dettagli nella programmazione di sistemi e cerca la modalità per ottenere il massimo delle prestazioni. Qualcuno che programma con entusiasmo (spesso ossessivamente) o si diverte programmando ancor prima che teorizzare sulla programmazione. Una persona capace di apprezzare. Una persona capace di programmare rapidamente. Una persona esperta nell'utilizzo di un dato programma. Una persona esperta o entusiasta per ogni ambito, anche l'astronomia.

Esiste anche una connotazione negativa del termine legata a fattori di criminalità informatica e viene espressamente indicata con il termine *cracker*. I media spesso confondono le due definizioni e utilizzano *hacker* con carattere negativo. Anche la *new economy* gioca con il fattore terrore e non è un caso che siano proprio i produttori di soluzioni di sicurezza a migliorare le proprie quotazioni in borsa durante attacchi informatici, un *effetto* che spesso autoalimenta la propria *causa* scaturante.

1.2. Cracker e Phreaker

Il termine è stato coniato all'incirca nel 1985 da hackers che cercavano di difendere e distaccarsi dalle connotazioni negative che i media usavano a dismisura. Fondamentalmente entrambi i termini delineano persone che hanno compiuto attività di cracking e le tecniche spesso sono le medesime ma spesso la differenza è nel fine. Mentre un hacker scardina delle misure di sicurezza per ottenere uno scopo benefico e aggirare dei limiti tecnici, un cracker si appropria della conoscenza per proprio esclusivo interesse. Ulteriore neologismo che si integra con i termini citati è phreaker, colui che compie cracking sulla rete telefonica per effettuare ad esempio chiamate a lunga distanza senza spendere nulla. Fino alla prima metà degli anni '80 la tecnologia che controllava le reti telefoniche era piuttosto antiquata e facilmente scardinabile per propri scopi. Il cambiamento avvenne con l'aggiornamento tecnologico successivo che tagliò radicalmente molte delle vecchie tecniche di phreaking. Anche l'arresto di gruppi di persone che compievano questa tipologia di azioni divenne un freno ed è diventato difficile sentir parlare di *blue box*¹ al giorno d'oggi.

1.3. Linux e l'arte di fare "hacking"

Linus Torvalds, il creatore di Linux, ha sempre definito il proprio sistema come "*un sistema operativo per hackers scritto da un hacker*". La chiave di lettura è piuttosto semplice da decifrare: lo sviluppo di soluzioni software prevede la necessità di automatizzare procedure umane rendendole più semplici, utili e usabili. Il networking ha trovato una rapida convergenza con il software negli ultimi anni aggiungendo potenzialità come rischi dal punto di vista della sicurezza. Compiere *azioni di hacking* verso questi software vuol dire testarne le capacità di sicurezza e stabilità onde evitare problemi o disservizi degli stessi. Da punto di vista più generale anche il supporto informativo degli utenti finali offrono un corposo feedback agli sviluppatori ove sia possibile farlo. Nel mondo del software libero è molto facile contattare i singoli sviluppatori del software creando un rapporto di cooperazione rapido ed efficace, cosa molto difficile e spesso eccessivamente filtrata nel mondo del software proprietario.

Linux e il software libero in generale non sarebbero quelli che sono se non avessero avuto l'apporto e la creatività degli hackers della Free Software Foundation di Stallman e dei cosiddetti *battitori liberi*, studenti e sviluppatori che portano la propria esperienza e il proprio tempo verso il *free software* circoscrivendo con grande rapidità il concetto di qualità totale o evoluzione nell'accezione scientifica.

¹Le blue box erano appositi strumenti che venivano utilizzati dai phreaker per simulare le chiamate telefoniche.

1.4. L'importanza della sicurezza informatica

Nel testo sono numerosi i richiami sulla sicurezza personale e la sicurezza lato server come strumento di gestione delle informazioni e delle comunicazioni condiviso. Di seguito viene accennato il comportamento giuridico italiano sul tema introdotto.

1.4.1. Limiti minimi di sicurezza per le aziende

Si è fatto cenno al termine *cracker* come "pirati informatici" e di conseguenza mi è sembrato ovvio riportare anche come la legge italiana si muove in merito. Dalla legge 31 dicembre 1996, n. 675 viene tratto quanto segue:

Art. 15 - Sicurezza dei dati

1. I dati personali oggetto di trattamento devono essere custoditi e controllati, anche in relazione alle conoscenze acquisite in base al progresso tecnico, alla natura dei dati e alle specifiche caratteristiche del trattamento, in modo da ridurre al minimo, mediante l'adozione di idonee e preventive misure di sicurezza, i rischi di distruzione o perdita, anche accidentale, dei dati stessi, di accesso non autorizzato o di trattamento non consentito o non conforme alle finalità della raccolta.
2. Le misure minime di sicurezza da adottare in via preventiva sono individuate con regolamento emanato con decreto del Presidente della Repubblica, ai sensi dell'articolo 17, comma 1, lettera a), della legge 23 agosto 1988, n. 400, entro centottanta giorni dalla data di entrata in vigore della presente legge, su proposta del Ministro di grazia e giustizia, sentiti l'Autorità per l'informatica nella pubblica amministrazione e il Garante.
3. Le misure di sicurezza di cui al comma 2 sono adeguate, entro due anni dalla data di entrata in vigore della presente legge e successivamente con cadenza almeno biennale, con successivi regolamenti emanati con le modalità di cui al medesimo comma 2, in relazione all'evoluzione tecnica del settore e all'esperienza maturata.
4. Le misure di sicurezza relative ai dati trattati dagli organismi di cui all'articolo 4, comma 1, lettera b), sono stabilite con decreto del Presidente del Consiglio dei ministri con l'osservanza delle norme che regolano la materia.

Art. 36 - Omessa adozione di misure necessarie alla sicurezza dei dati

1. Chiunque, essendovi tenuto, omette di adottare le misure necessarie a garantire la sicurezza dei dati personali, in violazione delle disposizioni dei regolamenti di cui ai commi 2 e 3 dell'articolo 15, è punito con la reclusione sino ad un anno. Se dal fatto deriva nocumento, la pena è della reclusione da due mesi a due anni.
2. Se il fatto di cui al comma 1 è commesso per colpa si applica la reclusione fino a un anno.

Il complesso di accorgimenti e cautele di tipo organizzativo e tecnologico che le imprese devono adottare al fine di evitare la dispersione, la perdita o l'uso illecito dei dati contenuti nei loro data base, rappresenta uno degli adempimenti più importanti previsti dalla legge n.675 del 1996, la cui individuazione è stata affidata al regolamento n.318 del luglio 1999. Il regolamento stabilisce che tali misure debbano essere adottate entro il 29 marzo 2000 ed è viva, pertanto, l'attenzione del

1. Hackers

mondo delle imprese ai problemi di adeguamento. L'incontro tecnico ha avuto, quindi, lo scopo di dissipare, attraverso il confronto e il dialogo con il Garante, dubbi interpretativi e difficoltà di applicazione delle norme.

La legge sulla privacy, oltre a garantire la salvaguardia dei dati personali di ogni persona, offre, dal punto di vista aziendale, una serie di vincoli e opportunità volta a definire una metodologia aziendale capace non solo di *promuovere la cultura della protezione dei dati*, ma anche di *salvaguardare il patrimonio informativo aziendale*. Va aggiunto che il regolamento non vuole descrivere a livello tecnico le soluzioni più idonee ma punire con sanzioni chi non adempie neppure le misure minime di sicurezza fissate dalla legge sulla privacy. Nello sviluppo di una cultura della protezione dei dati un ruolo fondamentale avrà la formazione del personale. La consapevolezza delle misure per la sicurezza e la protezione dei dati non devono essere viste come un vincolo ma come un'occasione di crescita, che consente una maggiore trasparenza dei flussi di informazioni, la revisione delle attività svolte e delle procedure di gestione e il riordino degli archivi².

1.5. Riflessioni

Questo primo capitolo ha l'intenzione di attivare *riflessioni*. Non si desidera tutelare il crimine informatico, che come tale deve essere punito, ma fare luce su aspetti che spesso vengono trascurati o trattati in maniera errata e poco documentata da molti giornalisti, spesso per ignoranza o *favori* a chi detiene interessi avversi alla filosofia del software libero. Essere "hacker" non vuol dire essere "criminali", ma rapportarsi in una determinata maniera di fronte alla tecnologia, un particolare approccio che spinge a testare una soluzione, materiale o immateriale come il software, per scoprirne i limiti e apportare propria genialità per migliorarla e renderla qualitativamente "migliore". Niente di diverso dalla naturale spinta evolutiva. *L'hacking* deve essere un fattore culturale presente nella vita reale di una comunità informatizzata dove l'individuo è rappresentato sempre più da una collezione di informazioni e la tutela di queste ultime decreta la privacy dello stesso.

²Per maggiori dettagli e sviluppi delle argomentazioni illustrate in questa sezione si consiglia di visitare il sito <http://www.interlex.it>.

2. Internet e Telematica

Internet ha decretato la nascita di una nuova civiltà fondata sulla comunicazione telematica. Sono cadute le barriere legate alla distanza metrica ed è possibile, grazie alla rete, comunicare in tempo reale con una persona nella parte opposta del globo. Internet ha reso il mondo a misura d'uomo.

2.1. Le origini

Alla fine degli anni '60 il Dipartimento della Difesa degli Stati Uniti sviluppò soluzioni per affrontare eventuali attacchi bellici. Erano gli anni della guerra fredda e questa condizione spingeva la ricerca verso sistemi di sicurezza e comunicazione. L'iniziale progetto venne affidato dal Dipartimento della Difesa ad ARPA¹.

Tra il 1962 e il 1964 Paul Baran, un ingegnere alle dipendenze della RAND Corporation² e figura in stretto rapporto con ARPA, elaborò 11 rapporti dedicati al problema di come costruire una rete di telecomunicazioni in grado di "sopravvivere" in caso di guerra nucleare e di attacco al territorio americano da parte di forze nemiche ed è da questi rapporti che è sorto per la prima volta il termine di "rete distribuita", un concetto basilare nel successivo sviluppo delle reti telematiche e di Internet.

Una rete distribuita è caratterizzata da una topologia a ragnatela e non esiste una gerarchia preordinata di nodi. Venne creata una prima rete di 4 nodi e la struttura era basata sul precedente concetto e sulla pariteticità dei nodi (peer to peer).

La trasmissione dei dati all'epoca era particolarmente vincolata alle infrastrutture hardware e i sistemi operativi utilizzati; ARPAnet era così vincolata a particolari infrastrutture di calcolo e comunicative. I limiti furono presto superati grazie allo sviluppo di un protocollo ideato per la trasmissione dei dati, il tcp/ip (Transmission Control Protocol/Internet Protocol), lo stesso protocollo che tuttora viene utilizzato per Internet. L'Internet Protocol Suite, completata negli anni '70, era in sintesi un mezzo flessibile per la comunicazione tra "mezzi di trasmissione" eterogenei. Dopo questa particolare innovazione ARPAnet divenne predominio dei ricercatori universitari che lavoravano per il Pentagono, la comunità scientifica che si accorse per prima della grande utilità nell'utilizzo remoto di supercalcolatori per scopi di ricerca e comunicazione tra le varie università americane sparse sul territorio nazionale. La mole di informazione crebbe a tal punto che nel 1984 ARPAnet venne scissa in due reti ben distinte: Milnet per scopi militari ed Arpanet per scopi scientifici. Il Dipartimento della Difesa Americana smise di finanziare il traffico interuniversitario ed il compito di gestione venne assunto dal National Science Foundation, che diede vita a NSFnet. Nel 1985 la rete scientifica cambiò nome in Internet e le stime mostravano 326 computer collegati dei quali 16 fuori dagli Stati Uniti. Nel 1990 erano collegati in rete circa 1000 calcolatori e tre anni dopo, nel 1993, la cifra divenne 20 volte superiore. Tutto questo fu possibile grazie alla *liberalizzazione* di Internet in ambiti accademici e scientifici e alla diffusione del sistema operativo Unix nella versione Berkeley, strumento storicamente in relazione con la rete stessa sin dalle sue prime fasi di sviluppo.

¹Advanced Research Project Agency.

²La RAND Corporation era un istituto civile di ricerche strategiche con sede a Santa Monica (California), descritto nel 1951 su Fortune come "l'azienda dell'aeronautica militare per l'acquisto di cervelloni", in cui brillanti accademici meditavano sulla guerra nucleare e sulla teoria dei giochi.

2.1.1. Autostrade dell' Informazione

All'inizio degli anni '90, come abbiamo rilevato precedentemente, il numero di calcolatori collegati ad Internet ebbe una crescita esponenziale tanto da interessare molti enti commerciali. I vincoli iniziali vietavano l'utilizzo della rete per scopi differenti da quelli scientifici e molte compagnie interessate furono spinte ad entrare in stretta correlazione con le università nell'ambito della ricerca per superare questi limiti. Molte compagnie commerciali crearono loro stesse delle dorsali di comunicazione da integrare ad Internet per i propri scopi e generarono un processo evolutivo senza limitazioni verso la copertura capillare del pianeta. Dopo una situazione simile le barriere vennero abbattute e si diede libero campo alle iniziative private e all'uso della rete per scopi variegati. Le autostrade dell'informazione (Information Highways), così come venne definita Internet, erano un obiettivo fondamentale nella campagna politica di Al Gore, un sogno che divenne realtà con il suo incarico a vice-presidente degli Stati Uniti durante il governo Clinton. Gli Stati Uniti erano consci del loro vantaggio tecnologico e dell'opportunità che detenevano nello sviluppo di queste strutture globali. La posta in gioco era composta da ingenti capitali economici ma l'interesse di modellare a loro immagine e somiglianza l'intera cultura moderna era l'elemento maggiormente perseguibile. Vennero fatte diverse scelte strategiche: finanziamenti statali per lo sviluppo di strutture scolastiche / sanitarie e la liberalizzazione delle comunicazioni, i primi erano volti allo sviluppo di servizi basilari e la seconda scelta diretta a fermare eventuali comportamenti di oligopolio.

2.1.2. Il Vecchio Continente

Pur con un approccio di base differente la Comunità Economica Europea avviò il suo cammino in modo differente con la speranza di seguire un comportamento equilibrato nelle sue scelte decisionali. Esisteva un progetto del 1985 denominato Cosine (Cooperation for Open Systems Interconnection in Europe) per la costituzione di una rete europea integrando realtà nazionali già esistenti. Il processo evolutivo di globalizzazione e sviluppo delle comunicazioni venne riassunto in un rapporto nel 1992, un documento importante che ha dettato le linee di comportamento per la liberalizzazione delle comunicazioni a partire dal gennaio 1998. Nell'ambito Italiano le direttive comunitarie sono state rispettate e si è visto il passaggio in fasi differenti da un unico ente di gestione delle telecomunicazioni ad una situazione aperta gestita da un numero di compagnie private e supervisionata dall'autorità governativa.

2.1.3. La situazione italiana

La storia di Internet nei periodi fino ad ora considerati è stata molto periferica e il mondo telematico si è mosso al di fuori dei protocolli TCP/IP e della rete delle reti. Questa situazione era comune sia nel contesto accademico che nella telematica amatoriale e commerciale. Nonostante questo isolamento tecnologico vi era un panorama variegato diviso tra Videotel, Fidonet, la rete dell'Istituto Nazionale di Fisica Nucleare (INFNet) e successivamente la rete accademica del GARR, la rete I2U degli utenti UNIX (l'origine di Iunet) e molte BBS cittadine.

L'inizio della telematica nazionale può determinarsi con l'anno 1986, anno in cui la SIP³ iniziò a commercializzare Videotel, un servizio telematico a struttura centralizzata al quale si accedeva tramite le normali reti telefoniche. A questo primo approccio susseguì la rete Fidonet, una struttura decentralizzata vicina al modello di Baran e fundamentalmente basato su un sistema di messaggistica. Fidonet era un'aggregazione di BBS⁴ con supporto per la tecnologia FTN, sistemi facilmente

³SIP, servizio telefonico nazionale poi diventato Telecom Italia.

⁴Bulletin Board System, sistemi telematici centralizzati nei quali uno o più macchine server fungono da centrali per lo smistamento di comunicazioni tra gli utenti. Le BBS sono genericamente sistemi telematici nei quali un gruppo di utenti può accedere a un ristretto insieme di file.

implementabili su semplici e comuni personal computer che tramite chiamate telefoniche notturne e automatizzate permettevano lo scambio dei messaggi sui nodi di riferimento. La relativa facilità con la quale era possibile implementare un nodo di questa rete si diffuse rapidamente alimentando la primordiale cultura underground nazionale. Al di fuori di Fidonet furono di particolare rilievo le esperienze di BBS come Agorà, legata al Partito Radicale, e MCLink, servizio complementare alla rivista MCmicrocomputer, entrambe localizzate a Roma. INFNet (Istituto Nazionale di Fisica Nucleare) fu la prima istituzione scientifica italiana a disporre di strutture telematiche già dal 1978; nel 1987 la rete era cresciuta e contava 118 nodi collegati e fu anche il primo centro di ricerca a disporre di una connessione permanente a Internet. Nel 1988 la situazione accademica italiana venne riconsiderata totalmente e venne costituito il GARR (Gruppo Armonizzazione Reti per la Ricerca), organizzazione coordinata dal Ministero della Ricerca scientifica alla quale era assegnato il compito di amministrare e sviluppare l'interconnessione tra le reti delle diverse università italiane. Gli iniziali poli della dorsale, o backbone, collegati ad alta velocità furono Milano, Bologna, Pisa, Roma e Bari ma presto la rete fu ampliata ad altre 50 centri universitari. Il primo Internet provider italiano fu Iunet nato grazie all'associazione I2U degli utenti Unix Italiani, raccordo italiano con la rete EUNET che legava esperti di Unix, matematici e informatici. Iunet è stata la prima rete Internet italiana non legata al GARR nata inizialmente senza finalità commerciali. Nel 1994 nacque a Milano Iunet SPA acquisita da Olivetti Telemedia e successivamente da Infostrada.

A conclusione di questa anteprima nazionale sul mondo telematico risalta che la propria popolazione era molto variegata comprendendo semplici smanettoni domestici o ricercatori universitari, soluzioni semplici come Videotel o reti ad alta velocità con servizi molto specializzati anche se i tassi di diffusione di questi servizi era tra i più bassi in Europa.

2.2. Le comunità virtuali

La telematica ha cambiato la società modificando i comportamenti di quest'ultima a un livello superiore rispetto a qualsiasi altra tecnologia. Come si è visto in questo capitolo, Internet è un progetto che ha radici molto lontane e in questo arco di tempo la rete si è evoluta per rispondere alle esigenze delle persone e delle aziende che investono in questa infrastruttura con un elemento in comune: la comunicazione.

Anche il software libero e Linux, in particolar modo, ha creato una propria comunità di utenti e successivamente tratteremo questo argomento nel dettaglio (71).

Netiquette

Come ogni comunità sociale che si forma, anche le comunità virtuali hanno una propria regolamentazione dei comportamenti. La Netiquette, come viene chiamata, è una forma di "educazione" che porta a rispettare e proteggere le interazioni tra le persone in rete. A titolo di esempio, possiamo citare come un caso estraneo alla Netiquette l'uso improprio di uno strumento come la posta elettronica per fare pubblicità commerciale. Questo fenomeno chiamato "spam" è stato largamente combattuto negli U.S.A. e l'Europa inizia a risentirne il contraccolpo in seguito alla crescita dei servizi di vendita online. Un altro caso di Netiquette può essere l'interazione che avviene tra un gruppo di persone in chat (IRC), dove comportamenti scorretti di alcuni soggetti possono limitare o interrompere i rapporti con le altre persone.

Per chi desiderasse approfondire gli argomenti trattati, il documento di riferimento è RFC1855 "Netiquette Guidelines", ed anche RFC2635 "A Set of Guidelines for Mass Unsolicited Mailings and Postings" disponibili sulla rete presso:

`ftp://ftp.nic.it/rfc/rfc1855.txt`

2. Internet e Telematica

<ftp://ftp.nic.it/rfc/rfc2635.txt>

2.3. Servizi e strumenti di comunicazione

La nascita e lo sviluppo della rete ha creato rapporti di comunicazione e servizi relativi. Diversi sono gli strumenti sviluppati per lo svolgimento delle attività quotidiane e di seguito vengono presentati alcuni tra i più importanti.

2.3.1. Posta elettronica

La posta elettronica (e-mail) è lo strumento più diffuso ed importante della rete che permette alle persone di comunicare tra loro tramite semplici messaggi testuali o messaggi ricchi di elementi multimediali come è possibile al giorno d'oggi con gli attuali strumenti software esistenti. Le persone abituate a lavorare con sistemi operativi di tipo Unix sono vicine alla messaggeria digitale da sempre presente in sistemi multiutente. Non è un caso che la forma degli indirizzi e-mail sia la stessa che rappresenta un utente collegato @ (at, presso) un sistema. Anche se sono andate perse tutte le caratteristiche tattili delle *vecchie lettere cartacee*, utilizzare la posta elettronica vuol dire utilizzare un mezzo veloce con costi ridotti e con la possibilità di trasferire file come allegati.

2.3.2. Newsgroup

I newsgroup sono gruppi di discussione divisi per argomento dove è possibile spedire un proprio messaggio in una bacheca virtuale pubblica dove altre persone potranno leggerlo ed eventualmente rispondere. E' una comunicazione *uno-a-molti* che tra le sue origini dal lontano 1979 quando alcuni studenti informatici delle università della Duke University e della University of North Carolina ne implementarono la struttura per consentire lo scambio di messaggi tra le due comunità informatiche al margine di ARPANET. La caratteristica che rivoluzionò a suo tempo questo servizio fu la possibilità di organizzare molteplici discussioni pubbliche su argomenti specifici non localizzate o controllate in siti centrali ma diffuse e sincronizzate tra i vari sistemi che supportavano il servizio.

2.3.3. IRC (Internet Relay Chat)

Probabilmente è il servizio telematico che affascina maggiormente le persone perchè permette di comunicare in tempo reale con altre persone sia in modo diretto che in forma pubblica ad un gruppo di persone presenti in determinati canali, aggregazioni che si possono liberamente creare con estrema facilità.

IRC originariamente nacque come protocollo di conferenza testuale per BBS nell'agosto del 1988 e vide la sua definizione ufficiale con il documento RFC 1459 del 1993. L'originale sviluppatore del sistema di comunicazione IRC fu il finlandese Jarkko Oikarinen ma l'interesse per questa soluzione software attrasse moltissimi sviluppatori e crebbe rapidamente implementando numerose caratteristiche e servizi. Il primo server disponibile su Internet fu `tolsun.oulu.fi` il quale funziona ancora nel momento in cui scrivo questa breve parentesi storica.

Al giorno d'oggi esistono numerose famiglie di server IRC sparsi in tutto il mondo e tra queste la più diffusa sul nostro territorio nazionale è IRCNET.

2.3.4. Il Web e gli ipertesti

E' il servizio Internet più recente ma anche quello che ha avuto maggior sviluppo negli ultimi anni. Utilizzando un navigatore o browser è possibile visualizzare pagine ipertestuali di informazione con il supporto di immagini, suoni, animazioni. Per definire cosa sia un ipertesto prendiamo in

considerazione le seguenti definizioni:

“Testo che non costituisce una singola sequenza e che può essere letto in diversi ordini; specialmente testo e grafica [...] che sono interconnessi in modo che il lettore del materiale (un sistema informatico) può interrompere la lettura del documento in determinati punti per consultare altro materiale correlato.” (Oxford Dictionary)

e anche la seguente:

“Termine coniato da Ted Nelson attorno al 1965 per designare un insieme di documenti contenenti riferimenti incrociati i quali permettono al lettore di muoversi facilmente da un documento all'altro” (Free On Line Dictionary of Computing)

In sintesi gli ipertesti sono insiemi di documenti consultabili in modo non sequenziale attraverso un software (es. browser⁵) che permette al lettore di muoversi agevolmente da un documento ad un altro documento correlato in modo discontinuo.

Sebbene il concetto di ipertesto risalga agli anni '60, l'applicazione comune più diffusa ha preso piede con la nascita del Web, la ragnatela, progettata da Tim Berners-Lee al CERN di Ginevra tra il 1989 e il 1990. Successivamente questo sistema si diffuse rapidamente passando da interfacce semplici e a caratteri a soluzioni software molto raffinate con supporto per elementi grafici e multimediali.

2.4. DNS e i Nomi Dominio

Il DNS (Domain Name System) è lo strumento utilizzato per gestire una parte molto importante di Internet: la risoluzione dei nomi di dominio.

2.4.1. Internet e le internet

E' doveroso analizzare, in questa sezione, due termini che spesso possono esser visti come identici. Le parole in questione sono “Internet” con la lettera iniziale maiuscola e “internet”. Sebbene possa sembrare una finezza, la distinzione c'è ed è importante. Internet, con la lettera maiuscola, fa riferimento alla rete che iniziò con il progetto ARPAnet e continua oggi come la confederazione di tutte le reti TCP/IP direttamente o indirettamente collegate al backbone⁶ degli Stati Uniti. La parola “internet” con tutte le lettere minuscole rappresenta tutte le reti, di piccole o grandi dimensioni, che utilizzano lo stesso protocollo di comunicazione. Ogni internet (lettere minuscole) non necessariamente dev'essere connessa a Internet (lettera maiuscola iniziale) e non necessariamente deve utilizzare tcp/ip come suo protocollo di base. Le internet aziendali sono un esempio, come quelle di tipo Xerox XNS o DECnet. Negli ultimi anni sono nati ulteriori nuovi termini più che altro spinti da politiche di marketing. Il termine “intranet” rappresenta una rete aziendale che utilizza gli strumenti di Internet in un ambito ristretto. Parallelamente il termine “extranet” rappresenta la rete che unisce diverse compagnie tra loro, oppure una compagnia con i propri rivenditori.

⁵Il browser è un programma che permette di accedere e consultare documenti tramite il protocollo HTTP. Il primo browser grafico per il Web è stato Mosaic ed è stato sviluppato da Marc Andressen nel 1993. Successivamente Andressen ha fondato la società Netscape.

⁶Il backbone è l'infrastruttura portante delle telecomunicazioni.

2.4.2. Le origini del Domain Name System

In questo stesso capitolo si è già accennato ai passaggi storici da ARPAnet a Internet. Nella primordiale rete esistente ogni computer collegato era caratterizzato da informazioni che lo identificavano in maniera univoca in rete. Il file che conteneva tutte queste informazioni era `HOSTS.TXT`⁷ ed era mantenuto dal Network Information Center dello SRI⁸. Gli amministratori dei sistemi connessi in rete ARPAnet inviavano i propri cambiamenti al SRI-NIC e scaricavano successivamente il file `HOSTS.TXT` con le variazioni avvenute. Era chiaramente un meccanismo macchinoso e lo divenne ancor più quando ARPAnet iniziò ad utilizzare il protocollo TCP/IP e la popolazione della rete crebbe enormemente. A questo punto il metodo basato sul file `HOSTS.TXT` riscontrava diversi problemi:

- **Collisione dei nomi** Era facile avere due nomi di computer identici in rete e questo si scontrava con il presupposto di univocità preposto.
- **Consistenza** Mantenere aggiornato il file `HOSTS.TXT` divenne sempre più difficile visti i continui aggiornamenti che venivano richiesti.
- **Traffico e carico** Gli strumenti presenti presso SRI-NIC divennero inefficienti a causa del traffico di rete incredibile e dell'alto carico che dovevano subire i processori dei Server presenti.

Le persone che governavano la rete ARPAnet cercarono una tecnologia valida per sostituire il metodo fino ad allora utilizzato, capace di scalare meglio le esigenze, ridurre il traffico in rete, facile da gestire e garante dell'integrità dei nomi di dominio esistenti. Nel 1984, Paul Mockapetris creò la basilare struttura del nuovo sistema e rilasciò in rete i documenti RFC⁹ 882 e 883 dove veniva descritto il Domain Name System, o semplicemente DNS. La prima implementazione di DNS era chiamata JEEVES ma venne rapidamente susseguita da BIND, un software sviluppato per la versione di Unix BSD 4.3¹⁰ da Kevin Dunlap.

2.4.3. Nomi di Dominio

Un nome di dominio è una *etichetta*, un nome che viene associato ad un indirizzo IP in rete ed è rappresentato da un nome e un top-level domain, ad esempio:

```
istruzione.it
```

Dove "istruzione" è in questo caso un nome fittizio e ".it" rappresenta la nazionalità del dominio in questione.

Top-Level Domain Lo spazio dei nomi di dominio era diviso originariamente in 7 Top-Level:

.com Organizzazioni di carattere commerciale, come ad esempio Silicon Graphics (`sgi.com`), Sun Microsystems (`sun.com`).

.edu Istituti di educazione americani, come ad esempio U.C. Berkeley (`berkeley.edu`), Purdue University (`purdue.edu`).

⁷In ambienti Unix rappresentato dal file `/etc/hosts`.

⁸SRI è l'istituto della ricerca di Stanford a Menlo Park, California.

⁹RFC o Request for Comments. Sono documenti che introducono una nuova tecnologia sulla rete Internet. Gli RFC sono distribuiti gratuitamente in rete e sono destinati agli sviluppatori che vogliono supportare o migliorare determinate tecnologie.

¹⁰UNIX BSD 4.3 è stato sviluppato dall'Università della California a Berkeley.

- .gov** Organizzazioni governative, come ad esempio la NASA (`nasa.gov`), National Science Foundation (`nsf.gov`).
- .mil** Organizzazioni militari, come ad esempio l'esercito degli Stati Uniti, U.S. Army (`army.mil`) e la Marina, U.S. Navy (`navy.mil`).
- .net** Organizzazioni orientate alle tecnologie di networking, come ad esempio NSFNET (`nsf.net`).
- .org** Organizzazioni non commerciali, come ad esempio Electronic Frontier Foundation (`eff.org`).
- .int** Organizzazioni internazionali, come ad esempio la NATO (`nato.int`)

Non è un caso che l'orientamento dei top-level domain sia modellato principalmente attorno alle esigenze degli Stati Uniti; si ricorda che proprio quest'ultimi finanziarono il progetto ARPANet, successivamente diventato Internet.

Lo sviluppo della rete ha mostrato le limitazioni degli originali top-level domain e a questi sono stati aggiunti i Domini Nazionali, ovvero una codifica di 2 lettere che identifica ogni singola nazione nel mondo secondo lo standard ISO 3166¹¹. L'Italia è rappresentata dal top-level domain `.it`.

Registrare un Nome di Dominio

La gestione dei nomi di dominio è gestita in modalità differenti. Nel caso di nomi di dominio che utilizzano gli originari 7 top-level domain la gestione era riservata fino a poco tempo fa esclusivamente all'InterNIC, un organo nazionale statunitense, mentre dopo una politica di deregulation ora è possibile usufruire di alternativi servizi da parte di imprese che si occupano appositamente di questi servizi telematici. Per quel che riguarda i nomi di dominio ISO 3166 la gestione viene affidata a livello nazionale. In Italia la gestione del top-level domain `it` è gestita dal GARR, autorità al quale è necessario rivolgersi per la registrazione.

2.4.4. DNS in Breve

Il Domain Name System è una banca dati distribuita che permette il controllo locale dei nomi di dominio su segmenti della rete. Tramite un sistema di replica, le modifiche fatte su un determinato segmento di rete da parte degli amministratori incaricati vengono "copiate" e distribuite pubblicamente nella totalità dei computer collegati. I programmi che gestiscono tuttora questi segmenti di rete sono denominati Name Server mentre sono Resolver i programmi che interrogano quest'ultimi per la risoluzione di un nome di dominio in rete. Per chiarire maggiormente il sistema prendiamo un esempio base. Un amministratore di DNS deve gestire un particolare dominio chiamato `linux.it`. In questo dominio è presente un computer in rete che vogliamo chiamare `erlug.linux.it`. L'amministratore per fare questo deve agire sul Name Server presente sul dominio `linux.it`, aggiungere l'indirizzo ip e il nome `erlug.linux.it`. Sarà compito del Name Server stesso, tramite il sistema che lo contraddistingue a replicare questi dati sui DNS di livello superiore finché l'informazione non è accessibile ad ogni computer in rete. Come si può capire questo metodo scompone il traffico di rete e il carico dei processori a livelli.

F.A.Q.

Introduciamo, sin dal primo capitolo, la sezione della F.A.Q. ovvero Frequently Asked Questions, le domande richieste maggiormente. L'autore valuta eventuali domande da parte dei lettori per ampliare la lista ed eliminare ulteriori dubbi sorti dagli argomenti in esame.

¹¹La Gran Bretagna in accordo con lo standard ISO 3166 dovrebbe utilizzare il top-level domain di tipo `gb`, ma molte organizzazioni preferiscono utilizzare l'estensione `uk`, probabilmente per motivazioni di carattere politico.

2. Internet e Telematica

Domanda: “Devo utilizzare obbligatoriamente il DNS per collegarmi ad Internet?” Per la connessione ad Internet non è necessario questo parametro ma lo diventa se volete utilizzare la pagine web o accedere ai servizi offerti dalla rete. Senza DNS dovrete conoscere gli indirizzi IP di ogni server in rete, mentre nel caso contrario basterebbe conoscere il nome di dominio e in automatico avviene la risoluzione. E' più semplice ricordarsi che per andare sul sito di ErLug (Emilia Romagna Linux User Group) basta aprire un browser e digitare l'indirizzo `http://erlug.linux.it` invece del numero IP che identifica il server in rete.

Domanda: “Cosa mi serve per poter gestire la posta elettronica?” Se si desidera gestire una corrispondenza di posta elettronica è chiaramente necessario disporre di una propria casella postale per le risposte. Esistono numerosi servizi che offrono gratuitamente caselle di posta elettronica. Nel caso italiano esistono molti provider Internet che insieme alla connessione includono questo servizio. Una volta ottenuta una casella postale servono pochi ulteriori parametri: il server SMTP, ovvero il servizio che gestisce la “spedizione” delle vostre e-mail. Esempio:

```
mail.provider.it
```

il server POP3 (o IMAP a seconda dei casi) per gestire la posta che arriva nella vostra casella postale. E' frequente trovare come server di questo tipo lo stesso utilizzato per la spedizione:

```
mail.provider.it
```

Il server di “spedizione” (SMTP o Simple Mail Transfer Protocol) non richiede parametri aggiuntivi per lo svolgimento delle proprie operazioni al contrario del server di “ricezione” (POP3) dove risiedono i vostri personali messaggi. Per accedere a quest'ultimo con un programma di gestione della posta elettronica è necessario aggiungere il parametro che identifica l'utente (genericamente la parte iniziale del vostro indirizzo di posta) e una parola d'ordine (password) che solo voi conoscete.

Domanda: “Utilizzando posta elettronica e newsgroup è possibile scaricare dei virus nel proprio computer?” Con la posta o le news è possibile trovare allegati dei file. Guardatevi da questi se non conoscete chi ve li spedisce! I virus sono particolari programmi che si insediano nel vostro computer alterando le operazioni basilari. Vengono trasmessi attraverso file eseguibili (ad esempio i file .exe nel mondo Dos/Windows) o attraverso documenti particolari che possono includere linguaggi macro al proprio interno (ad esempio i file .doc o .xls). Gli ultimi virus esaminati sono chiamati “Macro Virus”.

3. Unix

Negli anni '60, agli albori della Scienza dei Computer (*Computer Science*) i calcolatori non *parlavano* tra loro. Spesso i calcolatori di uno stesso produttore avevano bisogno di interpreti per *condividere* informazioni ed è inutile prendere in considerazione computer di produttori differenti. Sono passati molti anni e in estrema sintesi verranno illustrati nella pagine che seguono.

3.1. Da Multics a Unix

Nel 1965, nei laboratori della Bell, una divisione AT&T, veniva aperto un progetto chiamato Multics (Multiplexed Information and Computing Service) che vedeva la collaborazione di General Electric e del MIT¹ per trovare una soluzione capace di gestire centraline telefoniche, sistemi di calcolo multiutente con costi ridotti. Nonostante le buone intenzioni, problemi di budget decretarono la fine del progetto ma questo non fermò Ken Thompson e Dennis Ritchie che in seguito all'esperienze fatte precedentemente continuarono a lavorare su un progetto loro con i relativi ed evidenti problemi correlati come la necessità di trovare un calcolatore per lavorare. Venne preparato un piano di sponsorizzazioni ma questo fu rifiutato e le voci di corridoio dicono che Bill Baker, il vice-presidente dei Bell Labs, esclamò:

"Bell Labs just doesn't do business this way!"

Era un progetto molto vago sotto l'aspetto economico e non fu abbastanza convincente per ricevere i fondi necessari, là dove era già stata decretata la fine di Multics in precedenza. Ritchie e Thompson non si persero d'animo, prepararono un prospetto del sistema operativo e lo distribuirono tramite l'ufficio stampa dei Bell Labs ad un ampio numero di ricercatori. Questo scatto ebbe i suoi frutti e venne trovato un *piccolo e usato* PDP-7 da utilizzare per la sperimentazione e lo sviluppo del progetto. L'estate del '69 fu interamente dedicata a realizzare una bozza significativa per iniziare i lavori del progetto.

¹Massachusetts Institute of Technology



Figura 3.1.: Ritchie e Thompson in foto recenti

3. Unix



Figura 3.2.: Ritchie e Thompson di fronte al PDP

Nel 1970 fu battezzato *Unix* da Brian Kernighan come gioco di parole con *Multics* e fu sottoposto un prospetto a due responsabili dei dipartimenti di ricerca per l'acquisto di un PDP-11; Doug McIlroy e Lee McMahon furono di fondamentale importanza e una volta trovati i fondi necessari fu possibile acquistare il calcolatore per il progetto. Il porting di Unix dal PDP-7 al PDP-11 venne fatto tramite due terminali Teletype 33 (immagine ??) e rapidamente attrasse a sè credibilità nei confronti degli altri dipartimenti di ricerca. Sulla spinta del successo fu ulteriormente proposto di acquistare un nuovo sistema PDP-11/45 e quest'ultima proposta non ebbe problemi a concretizzarsi.

In origine tutto il sistema operativo era scritto in linguaggio assembly. Era necessario trovare una soluzione per rendere questo iniziale sistema operativo portabile su altri calcolatori in maniera semplice. Ken Thompson inventò invece un linguaggio chiamato B, che fu influenzato da un altro linguaggio chiamato BCPL². Nel 1971 cominciò il lavoro al linguaggio che sarebbe diventato il C, un passo evolutivo che svincolava il linguaggio dal precedente calcolatore utilizzato per lo sviluppo.

Nel 1973 il kernel di Unix fu riscritto in C e poteva funzionare su dieci macchine. Nell'ottobre del 1973, Thompson e Ritchie presentarono un paper al Symposium for Operating System Principles e l'interesse per il sistema esplose. All'epoca, la AT&T era indagata per comportamenti monopolistici e le fu ordinato di rimanere fuori dall'industria dei computer; i suoi legali decisero che sarebbe stato giusto permettere alle università di disporre del sistema operativo alle condizioni dettate dalla stessa compagnia che in buona parte si sintetizzavano nella totale assenza di assistenza tecnica. Ciò costrinse gli utenti a riunirsi per potersi prestare assistenza reciprocamente, rinforzando i valori che all'inizio avevano portato alla creazione del sistema. Ma una tradizione di condivisione non avrebbe impedito che nella comunità degli utenti si formassero profonde divisioni - e due delle prime, a quel tempo stavano proprio per nascere. Nel 1975 Ken Thompson ritornò all'Università della California a Berkeley, e portò Unix con sé. Là due dottorandi, Chuck Haley e Bill Joy, fecero il porting del sistema Pascal, al quale Thompson aveva lavorato, ad uno stato in cui poteva essere utilizzato, e crearono l'editor di testi vi. Nacque così la *Berkeley System Distribution* di Unix o più semplicemente BSD Unix, una soluzione che veniva distribuita su nastro su richiesta. Le stesse intenzioni erano ben lontane dagli scopi commerciali; la diffusione di BSD Unix, come affermò Joy, era un modo per ampliare la base degli utenti e avere maggior feedback e così fu.

Nel corso degli anni BSD Unix venne continuamente migliorato e ampliato. La "Second Berkeley Software Distribution" (2BSD) era pronta nel 1978 e a questa susseguì una nuova versione nel dicembre 1979, la 3BSD. Quest'ultima venne particolarmente apprezzata all'interno degli ambienti scientifici e universitari. Grazie a Bob Fabry viene ricevuto un ingente finanziamento dal progetto DARPA (Defense Advanced Research Project Agency) volto a migliorare e sviluppare le funzionali-

²Basic Combined Programming Language, ideato da Martin Richards presso l'università di Cambridge.

tà di rete del sistema operativo. Nel 1980 vengono rilasciate le distribuzioni 4BSD e 4.1BSD, il primo sostanzioso passo nel ramo networking su TCP/IP.

Nella metà degli Anni Settanta Unix stava crescendo rapidamente, ma veniva spesso sottovalutato in rapporto a soluzioni presenti. Nel 1978 la DEC mise in commercio VAX, che era il successore del PDP-11, la culla di Unix. Questi calcolatori si diffusero rapidamente e il loro sistema operativo proprietario, il VMS, fu largamente sostituito da Unix, specialmente dopo l’uscita del BSD 4.2 nel 1982. AT&T nel 1984 fondò la propria divisione computer e vennero delineati alcuni prodotti da offrire sul mercato, tra questi Unix System V che girava su calcolatori 3B della stessa compagnia. Nello stesso anno Bill Joy lasciò l’università e fondò la Sun Microsystems, compagnia che produceva calcolatori con incluso BSD Unix e il supporto per TCP/IP. Fu una scelta strategica quella di allegare il software al proprio hardware senza costi aggiuntivi. Il mercato aveva alta profittabilità e rapidamente accorsero nuovi concorrenti come Convex, Multiflow, Masscomp, Silicon Graphics, Pyramid, Stardent e NeXT. Era l’epoca dei *killer micros*, come furono chiamati, soluzioni hardware più veloci e meno costose dei minicomputer e corredate da sistemi Unix e Unix-like. I produttori di workstation distrussero il mercato dei minicomputer e poi cominciarono a farsi guerra tra loro. Unix si era spostato al centro dell’attenzione seppure ogni produttore offrì la propria versione del sistema operativo. Alcune di queste erano basate sullo stile che si era sviluppato intorno al BSD, mentre altre si richiamavano all’USG System V Release 4 (SVR4) della AT&T.

3.2. La “Networking Release 1”

Le varie generazioni di Unix si susseguirono fino al 1989, anno di nascita della “Network Release 1”, una versione di Unix caratterizzata dalla totale libertà di utilizzo del codice sorgente, cosa che precedentemente era vincolata da una particolare licenza AT&T. Avvenne un cambiamento rilevante e le spinte motrici furono motivazioni di carattere economico (costi aggiuntivi) e la libera fruizione di codice sorgente. La somma di denaro da versare ad AT&T per ottenere la licenza di accesso al codice sorgente non era particolarmente rilevante ma coincise con l’accattivante soluzione proposta dall’Università di Berkeley.

In pochissimi mesi, un esercito di programmatori sostituì ogni applicazione software che potesse essere ricondotta ad AT&T; anche il kernel non fu da meno ma a causa di motivazioni tecniche e legali non fu possibile eliminare determinate porzioni di codice. Ogni vincolo venne superato con la “Network Release 2” del 1991. Nell’anno successivo, Bill Jolitz annuncia la nascita di 386/BSD, versione di BSD Unix scritta per microprocessori Intel 80386, il trampolino di lancio per il NetBSD Group, una task force intenta a migliorare le soluzioni per l’installazione/diffusione lato “utente”. Il 1992 segna inoltre il passaggio di mano degli Unix System Laboratories da AT&T a Novell, la quale compie a sua volta un passaggio nel 1995 in favore di SCO (Santa Cruz Operation).

Non si può negare la vita travagliata del sistema operativo ma nonostante questo ha affascinato e continua ad affascinare una larghissima schiera di utenti e molto spesso stupire per caratteristiche che il futuro decreterà essenziali nel settore informatico contemporaneo.

3.3. Free Software Foundation

Nel 1983 Richard M. Stallman fondò il progetto GNU e la Free Software Foundation, il cui scopo era produrre un sistema Unix-compatibile completamente gratuito e liberamente distribuibile. Tuttavia, il metodo di questo gruppo era cominciare con le utilities - editor di testo, compilatori, e tutto il resto - lavorando in direzione di un kernel completamente nuovo. Malgrado non si sia ancora

3. *Unix*

realizzato il progetto a livello del kernel³ il software prodotto da GNU era di una tale qualità che gli amministratori di sistema spesso sostituirono i programmi originali delle loro distribuzioni di Unix con le versioni GNU. Malgrado questo le compagnie preferirono curare i loro interessi giocando probabilmente contro questa filosofia di libero scambio del codice senza limitazioni.

F.A.Q.

Domanda: “C’è differenza tra Unix e UNIX, scritto con lettere maiuscole?” Con *Unix* indichiamo l’insieme di tutti i sistemi di questo tipo. *UNIX* è un marchio registrato che indica la realizzazione di Unix creata da USL (Unix System Laboratories) della Novell.

Domanda: “Che importanza ha avuto il sistema operativo Unix nello sviluppo della rete Internet?” Esiste un rapporto sinergico tra Unix e Internet. Se da un lato il sistema operativo ha trovato ampiamente sviluppo in rete, Internet e le tecnologie relative hanno trovato la propria culla nel sistema operativo in questione. L’esempio del Domain Name System è rappresentativo. L’organizzazione delle informazioni è gerarchicamente molto simile alla struttura ad albero del file system Unix.

Domanda: “Unix è ancora un privilegio per pochi? Serve necessariamente un computer molto potente per utilizzare questo sistema operativo?” Esistono versioni commerciali e “free”, nel concetto Open Sources. Le versioni commerciali genericamente sono sviluppate per particolari piattaforme hardware e ottimizzate per sfruttare le caratteristiche di quest’ultime. Le versioni “free” grazie alla loro natura “aperta” si evolvono rapidamente offrendo il supporto per un ampio insieme di periferiche e molte caratteristiche tecniche tipiche delle versioni commerciali. Tra i sistemi “free” possiamo ricordare GNU/Hurd e GNU/Linux: entrambi hanno un ampio supporto di periferiche e offrono prestazioni molto buone sui computer comunemente in vendita sul mercato.

³Il progetto HURD rappresenta il sogno di GNU ma questo kernel è ancora in fase ampiamente sperimentale ai giorni nostri. Per maggiori informazioni in merito è possibile leggere informazioni online su <http://www.gnu.ai.mit.edu/software/hurd/>.

Parte II.
Filosofia

4. Open Source

Dopo aver presentato la storia di UNIX e accennato alla nascita del movimento GNU è sembrato doveroso dare maggiori dettagli sul movimento che ha coinvolto una comunità di sviluppatori molto ampia e ha dato vita tra i numerosi progetti al sistema operativo GNU/Linux.

Proprio per aumentare l'informazione in merito sono stati scelti due documenti autorevoli scritti da altrettanto autorevoli guru. Il primo documento è di Richard Stallman, l'uomo che ha fondato la Free Software Foundation, considerato per molti un visionario per molti altri un genio. Il testo vuole essere una base da analizzare per creare una propria opinione in merito. Il secondo documento è di Bruce Perens, il fondatore della distribuzione Debian, ed è una dettagliata analisi delle varie licenze software presenti con relative caratteristiche. Il testo è diretto con particolare cura agli sviluppatori ma può essere un ottimo riferimento per gli utenti finali che utilizzeranno soluzioni software per le proprie esigenze.

Puntualizzazioni dell'autore Mentre il primo articolo è indiscutibile perchè trattasi di una esposizione storica personale sul secondo articolo vanno fatte delle premesse. Perens introduce problematiche sulle licenze dei due principali ambienti desktop per GNU/Linux, ovvero GNOME e KDE, preferendo il primo progetto per motivazioni etiche. L'articolo coglie indiscutibilmente nel segno ma non è aggiornato al cambiamento di licenza in GPL delle librerie Qt di Troll Tech, lo strato basilare del desktop KDE. In seguito a questo cambiamento sembra diventato superficiale mantenere due progetti affini con uguali motivazioni di carattere etico. Dal punto di vista dell'usabilità KDE è anche molto avanti rispetto al progetto antagonista.

4.1. Il progetto GNU¹

di Richard Stallman

La prima comunità di condivisione del software

“Quando cominciai a lavorare nel laboratorio di Intelligenza Artificiale del MIT nel 1971, entrai a far parte di una comunità in cui ci si scambiavano i programmi, che esisteva già da molti anni. La condivisione del software non si limitava alla nostra comunità; è un cosa vecchia quanto i computer, proprio come condividere le ricette è antico come il cucinare. Ma noi lo facevamo più di quasi chiunque altro.

Il laboratorio di Intelligenza Artificiale (AI) usava un sistema operativo a partizione di tempo (timesharing) chiamato ITS (Incompatible Timesharing System) che il gruppo di hacker² del laboratorio aveva progettato e scritto in linguaggio assembler per il Digital PDP-10, uno dei grossi elabo-

¹La traduzione di questo saggio è stata revisionata e curata, con la supervisione dell'autore, da Lorenzo Bettini, Antonio Cisternino, Francesco Potorti e Alessandro Rubini. Al documento sono state apportate correzioni e migliorie della traduzione.

²Nota di Stallman: L'uso del termine *hacker* nel senso di *pirata* è una confusione di termini creata dai mezzi di informazione. Noi hacker ci rifiutiamo di riconoscere questo significato, e continuiamo a utilizzare la parola nel senso di uno che ami programmare, e a cui piaccia essere bravo a farlo.

4. Open Source



Figura 4.1.: Richard Stallman

ratori di quel periodo. Come membro di questa comunità, hacker di sistema nel gruppo laboratorio, il mio compito era migliorare questo sistema.

Non chiamavamo il nostro software *software libero*, poiché questa espressione ancora non esisteva, ma si trattava proprio di questo. Quando persone di altre università o di qualche società volevano convertire il nostro programma per il proprio sistema e utilizzarlo, erano le benvenute. Se si vedeva qualcuno usare un programma sconosciuto e interessante, si poteva sempre chiedere di vederne il codice sorgente, in modo da poterlo leggere, modificare, o prenderne, cannibalizzarne alcune parti per creare un nuovo programma.”

La comunità si dissolve

“La situazione cambiò drasticamente all’inizio degli anni ’80 quando la Digital smise di produrre la serie PDP-10. La sua architettura, elegante e potente negli anni ’60, non poteva essere estesa in modo naturale ai più grandi spazi di indirizzamento che si stavano rendendo possibili negli anni ’80. Questo significò che quasi tutti i programmi che formavano ITS divennero obsoleti.

La comunità di hacker del laboratorio di Intelligenza Artificiale si era già dissolta non molto tempo prima. Nel 1981 la Symbolics, nata da una costola del laboratorio stesso, gli aveva sottratto quasi tutti gli hacker; l’ormai esiguo gruppo rimasto fu dunque incapace di sostenersi (il libro *Hackers* di Steve Levy narra questi eventi, oltre a fornire una fedele ricostruzione di questa comunità ai suoi inizi). Quando il laboratorio di Intelligenza Artificiale nel 1982 acquistò un nuovo PDP-10, i sistemisti decisero di utilizzare il sistema timesharing non libero della Digital anziché ITS. I moderni elaboratori di quell’epoca, come il VAX o il 68020, avevano il proprio sistema operativo, ma nessuno di questi era libero: si doveva firmare un accordo di non-diffusione persino per ottenerne una copia eseguibile.

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

L’idea che la concezione sociale di software proprietario - cioè il sistema che impone che il software non possa essere condiviso o modificato - sia antisociale, contraria all’etica, semplicemente sbagliata, può apparire sorprendente a qualche lettore. Ma che altro possiamo dire di un sistema che si basa sul dividere utenti e lasciarli senza aiuto? Quei lettori che trovano sorprendente l’idea possono aver data per scontata la concezione sociale di software proprietario, o averla giudicata uti-

lizzando lo stesso metro suggerito dal mercato del software proprietario. I produttori di software hanno lavorato a lungo e attivamente per diffondere la convinzione che c'è un solo modo di vedere la cosa.

Quando i produttori di software parlano di "difendere" i propri "diritti" o di "fermare la pirateria", quello che dicono è in realtà secondario. Il vero messaggio in quelle affermazioni sta nelle assunzioni inesprese, che essi danno per scontate; vogliono che siano accettate acriticamente. Esaminiamole, dunque.

Una prima assunzione è che le aziende produttrici di software abbiano il diritto naturale indiscutibile di proprietà sul software, e di conseguenza, abbiano controllo su tutti i suoi utenti. Se questo fosse un diritto naturale, non potremmo sollevare obiezioni, indipendentemente dal danno che possa recare ad altri. È interessante notare che, negli Stati Uniti, sia la costituzione che la giurisprudenza rifiutano questa posizione: il diritto d'autore non è un diritto naturale, ma un monopolio imposto dal governo che limita il diritto naturale degli utenti a effettuare delle copie.

Un'altra assunzione inespressa è che la sola cosa importante del software sia il lavoro che consente di fare - vale a dire che noi utenti non dobbiamo preoccuparci del tipo di società in cui ci è permesso vivere. Una terza assunzione è che non avremmo software utilizzabile (o meglio, che non potremmo mai avere un programma per fare questo o quell'altro particolare lavoro) se non riconosciamo ai produttori il controllo sugli utenti di quel programmi. Questa assunzione avrebbe potuto sembrare plausibile, prima che il movimento del software libero dimostrasse che possiamo scrivere quantità di programmi utili senza bisogno di metterci dei catenacci.

Se rifiutiamo di accettare queste assunzioni, giudicando queste questioni con comuni criteri di moralità e di buon senso dopo aver messo al primo posto gli interessi degli utenti, tenendo conto che gli utenti vengono prima di tutto, arriviamo a conclusioni del tutto differenti. Chi usa un calcolatore dovrebbe essere libero di modificare i programmi per adattarli alle proprie necessità, ed essere libero di condividere il software, poiché aiutare gli altri è alla base della società.

Non c'è modo in questa sede di trattare approfonditamente i ragionamenti che portano a questa conclusione; il lettore interessato può cercare le informazioni in rete a questo indirizzo:

<http://www.gnu.org/philosophy/why-free.html>

Una difficile scelta morale

"Una volta che il mio gruppo si fu sciolto, continuare come prima fu impossibile. Mi trovai di fronte a una difficile scelta morale. La scelta facile sarebbe stata quella di unirsi al mondo del software proprietario, firmando accordi di non-diffusione e promettendo di non aiutare i miei compagni hacker. Con ogni probabilità avrei anche sviluppato software che sarebbe stato distribuito secondo accordi di non-diffusione, contribuendo così alla pressione su altri perché a loro volta tradissero i propri compagni. In questo modo avrei potuto guadagnare, e forse mi sarei divertito a programmare. Ma sapevo che al termine della mia carriera mi sarei voltato a guardare indietro, avrei visto anni spesi a costruire muri per dividere le persone, e avrei compreso di aver contribuito a rendere il mondo peggiore.

Avevo già sperimentato cosa significasse un accordo di non diffusione per chi lo firmava, quando qualcuno rifiutò a me e al laboratorio AI del MIT il codice sorgente del programma di controllo della nostra stampante; l'assenza di alcune funzionalità nel programma rendeva oltremodo frustrante l'uso della stampante. Per cui non mi potevo dire che gli accordi di non-diffusione fossero innocenti. Ero molto arrabbiato quando quella persona si rifiutò di condividere il programma con noi; non potevo far finta di niente e fare lo stesso con tutti gli altri.

Un'altra possibile scelta, semplice ma spiacevole, sarebbe stata quella di abbandonare l'informatica. In tal modo le mie capacità non sarebbero state mal utilizzate, tuttavia sarebbero state sprecate. Non sarei mai stato colpevole di dividere o imporre restrizioni agli utenti di calcolatori, ma queste cose sarebbero comunque successe.

4. Open Source

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

La risposta era semplice: innanzitutto serviva un sistema operativo. Questo è difatti il software fondamentale per iniziare a usare un computer. Con un sistema operativo si possono fare molte cose; senza, non è proprio possibile far funzionare il computer. Con un sistema operativo libero, avremmo potuto avere nuovamente una comunità in cui hacker possono cooperare, e invitare chiunque a unirsi al gruppo. E chiunque sarebbe stato in grado di usare un calcolatore, senza dover cospirare fin dall'inizio per sottrarre qualcosa ai propri amici.

Essendo un programmatore di sistemi, possedevo le competenze adeguate per questo lavoro. Così, anche se non davo il successo per scontato, mi resi conto di essere la persona giusta per farlo. Scelsi di rendere il sistema compatibile con Unix, in modo che fosse portabile, e che gli utenti Unix potessero passare facilmente a esso. Il nome GNU fu scelto secondo una tradizione hacker, come acronimo ricorsivo che significa "GNU's Not Unix" (GNU non è Unix).

Un sistema operativo non si limita solo al suo nucleo, che è proprio il minimo per eseguire altri programmi. Negli anni '70, qualsiasi sistema operativo degno di questo nome includeva interpreti di comandi, assembleri, compilatori, interpreti di linguaggi, debugger, editor di testo, programmi per la posta e molto altro. ITS li aveva, Multics li aveva, VMS li aveva e Unix li aveva. Anche il sistema operativo GNU li avrebbe avuti.

Tempo dopo venni a conoscenza di questa massima, attribuita a Hillel³:

"Se non sono per me stesso, chi sarà per me?
E se sono solo per me stesso, che cosa sono?
E se non ora, quando?"

La decisione di iniziare il progetto GNU si basò su uno spirito simile."

"Free" come libero

"Il termine *free software*" (N.d.T. il termine *free* in inglese significa sia gratuito che libero) a volte è mal interpretato: non ha niente a che vedere col prezzo del software; si tratta di libertà. Ecco, dunque, la definizione di software libero: un programma è software libero per un dato utente se:

- > utente ha la libertà di eseguire il programma per qualsiasi scopo;
- > l'utente ha la libertà di modificare il programma secondo i propri bisogni (perché questa libertà abbia qualche effetto in pratica, è necessario avere accesso al codice sorgente del programma, poiché apportare modifiche a un programma senza disporre del codice sorgente è estremamente difficile);
- > l'utente ha la libertà di distribuire copie del programma, gratuitamente o dietro compenso;
- > l'utente ha la libertà di distribuire versioni modificate del programma, così che la comunità possa fruire dei miglioramenti apportati.

Poiché "free" si riferisce alla libertà e non al prezzo, vendere copie di un programma non contraddice il concetto di software libero. In effetti, la libertà di vendere copie di programmi è essenziale: raccolte di software libero vendute su CD-ROM sono importanti per la comunità, e la loro vendita

³Nota di Stallman: Essendo ateo, non seguo alcuna guida religiosa, ma a volte mi trovo ad ammirare qualcosa che qualcuno di loro ha detto.

è un modo di raccogliere fondi importante per lo sviluppo del software libero. Di conseguenza, un programma che non può essere liberamente incluso in tali raccolte non è software libero.

A causa dell'ambiguità del termine "free", si è cercata a lungo un'alternativa, ma nessuno ne ha trovata una valida. La lingua inglese ha, più termini e sfumature di ogni altra, ma non ha una parola semplice e non ambigua che significhi libero; "unfettered" è la parola più vicina come significato (N.d.T. unfettered è una parola di tono aulico o arcaico che significa libero da ceppi, vincoli o inibizioni). Alternative come "liberated", "freedom" e "open" hanno altri significati o non sono adatte per altri motivi (N.d.T. rispettivamente, liberato, libertà, aperto)."

Software GNU e il sistema GNU

"Sviluppare un intero sistema è un progetto considerevole. Per raggiungere l'obiettivo decisi di adattare e usare parti di software libero tutte le volte che fosse possibile. Per esempio, decisi fin dall'inizio di usare T_EX come il principale programma di formattazione di testo; qualche anno più tardi, decisi di usare l'X Window System piuttosto che scrivere un altro sistema a finestre per GNU.

A causa di questa decisione, il sistema GNU e la raccolta di tutto il software GNU non sono la stessa cosa. Il sistema GNU comprende programmi che non sono GNU, sviluppati da altre persone o gruppi di progetto per i propri scopi, ma che possiamo usare in quanto software libero."

L'inizio del progetto

"Nel gennaio 1984 lasciai il mio posto al MIT e cominciai a scrivere software GNU. Dovetti lasciare il MIT, per evitare che potesse interferire con la distribuzione di GNU come software libero. Se fossi rimasto, il MIT avrebbe potuto rivendicare la proprietà del lavoro, e avrebbe potuto imporre i propri termini di distribuzione, o anche farne un pacchetto proprietario. Non avevo alcuna intenzione di fare tanto lavoro solo per vederlo reso inutilizzabile per il suo scopo originario: creare una nuova comunità di condivisione di software.

A ogni buon conto, il professor Winston - allora responsabile del laboratorio AI del MIT - mi propose gentilmente di continuare a utilizzare le attrezzature del laboratorio stesso."

I primi passi

"Poco dopo aver iniziato il progetto GNU, venni a sapere del Free University Compiler Kit, noto anche come VUCK (la parola olandese che sta per "free" inizia con la V). Era un compilatore progettato per trattare più linguaggi, fra cui C e Pascal, e per generare codice binario per diverse architetture. Scrisi al suo autore chiedendo se GNU avesse potuto usarlo. Rispose in modo canzonatorio, dicendo che l'università era sì libera, ma non il compilatore. Decisi allora che il mio primo programma per il progetto GNU sarebbe stato un compilatore multilinguaggio e multiplatforma. Sperando di evitare di dover scrivere da me l'intero compilatore, ottenni il codice sorgente del Pastel, un compilatore multiplatforma sviluppato ai Laboratori Lawrence Livermore. Il linguaggio supportato da Pastel, in cui il Pastel stesso era scritto, era una versione estesa del Pascal, pensata come linguaggio di programmazione di sistemi. Io vi aggiunsi un frontend per il C, e cominciai il porting per il processore Motorola 68000, ma fui costretto a rinunciare quando scoprii che il compilatore richiedeva diversi megabyte di memoria sullo stack, mentre il sistema Unix disponibile per il processore 68000 ne permetteva solo 64K.

Mi resi conto allora che il compilatore Pastel interpretava tutto il file di ingresso creandone un albero sintattico, convertiva questo in una catena di "istruzioni", e quindi generava l'intero file di uscita senza mai liberare memoria. A questo punto, conclusi che avrei dovuto scrivere un nuovo compilatore da zero. Quel nuovo compilatore è ora noto come Gcc; non utilizza niente del compi-

4. Open Source

latore Pastel, ma riuscii ad adattare e riutilizzare il frontend per il C che avevo scritto. Questo però avvenne qualche anno dopo; prima, lavorai su GNU Emacs.”

GNU Emacs

“Cominciasti a lavorare su GNU Emacs nel settembre 1984, e all’inizio del 1985 cominciava a essere utilizzabile. Così potei iniziare a usare sistemi Unix per scrivere; fino ad allora, avevo scritto sempre su altri tipi di macchine, non avendo nessun interesse a imparare vi ne’ ed.

A questo punto alcuni cominciarono a voler usare GNU Emacs, il che pose il problema di come distribuirlo. Naturalmente lo misi sul server ftp anonimo del computer che usavo al MIT (questo computer, prep.ai.mit.edu, divenne così il sito ftp primario di distribuzione di GNU; quando alcuni anni dopo andò fuori servizio, trasferimmo il nome sul nostro nuovo ftp server). Ma allora molte delle persone interessate non erano su Internet e non potevano ottenere una copia via ftp, così mi si pose il problema di cosa dir loro.

Avrei potuto dire: “trova un amico che è in rete disposto a farti una copia”. Oppure avrei potuto fare quel che feci con l’originario Emacs su PDP-10, e cioè dir loro: “spediscimi una busta affrancata e un nastro, e io te lo rispedisco con sopra Emacs”. Ma ero senza lavoro, e cercavo un modo di far soldi con il software libero. E così feci sapere che avrei spedito un nastro a chi lo voleva per 150 dollari. In questo modo, creai un’impresa di distribuzione di software libero, che anticipava le compagnie che oggi distribuiscono interi sistemi GNU basati su Linux.”

Un programma è libero per tutti?

“Se un programma è software libero quando esce dalle mani del suo autore, non significa necessariamente che sarà software libero per chiunque ne abbia una copia. Per esempio, il software di pubblico dominio (software senza copyright) è software libero, ma chiunque può farne una versione modificata proprietaria. Analogamente, molti programmi liberi sono protetti da diritto d’autore, ma vengono distribuiti con semplici licenze permissive che permettono di farne versioni modificate proprietarie.

L’esempio emblematico della questione è l’X Window System. Sviluppato al MIT, e pubblicato come software libero con una licenza permissiva, fu rapidamente adottato da diverse società informatiche. Queste aggiunsero X ai loro sistemi Unix proprietari, solo in forma binaria, e coperto dello stesso accordo di non-diffusione. Queste copie di X non erano software più libero di quanto lo fosse Unix.

Gli autori dell’X Window System non ritenevano che questo fosse un problema, anzi se lo aspettavano ed era loro intenzione che accadesse. Il loro scopo non era la libertà, ma semplicemente il “successo”, definito come “avere tanti utenti”. Non erano interessati che questi utenti fossero liberi, ma solo che fossero numerosi.

Questo sfociò in una situazione paradossale, in cui due modi diversi di misurare la quantità di libertà risultavano in risposte diverse alla domanda “questo programma è libero”? Giudicando sulla base della libertà offerta dai termini distributivi usati dal MIT, si sarebbe dovuto dire che X era software libero. Ma misurando la libertà dell’utente medio di X, si sarebbe dovuto dire che X era software proprietario. La maggior parte degli utenti di X usavano le versioni proprietarie fornite con i sistemi Unix, non la versione libera.”

Il permesso d’autore (copyleft) e la GNU GPL

“Lo scopo di GNU consisteva nell’offrire libertà agli utenti, non solo nell’ottenere ampia diffusione. Avevamo quindi bisogno di termini di distribuzione che evitassero che il software GNU fosse

trasformato in software proprietario. Il metodo che usammo si chiama “permesso d’autore”.⁴

Il permesso d’autore (copyleft)⁵. usa le leggi sul diritto d’autore (copyright), ma le capovolge per ottenere lo scopo opposto: invece che un metodo per privatizzare il software, diventa infatti un mezzo per mantenerlo libero. Il succo dell’idea di permesso d’autore consiste nel dare a chiunque il permesso di eseguire il programma, copiare il programma, modificare il programma, e distribuirne versioni modificate, ma senza dare il permesso di aggiungere restrizioni. In tal modo, le libertà essenziali che definiscono il “free software” (software libero) sono garantite a chiunque ne abbia una copia, e diventano diritti inalienabili.

Perché un permesso d’autore sia efficace, anche le versioni modificate devono essere libere. Ciò assicura che ogni lavoro basato sul nostro sia reso disponibile per la nostra comunità, se pubblicato. Quando dei programmatori professionisti lavorano su software GNU come volontari, è il permesso d’autore che impedisce ai loro datori di lavoro di dire: “non puoi distribuire quei cambiamenti, perché abbiamo intenzione di usarli per creare la nostra versione proprietaria del programma”.

La clausola che i cambiamenti debbano essere liberi è essenziale se vogliamo garantire libertà a tutti gli utenti del programma. Le aziende che privatizzarono l’X Window System di solito avevano apportato qualche modifica per portare il programma sui loro sistemi e sulle loro macchine. Si trattava di modifiche piccole rispetto alla mole di X, ma non banali. Se apportare modifiche fosse una scusa per negare libertà agli utenti, sarebbe facile per chiunque approfittare di questa scusa.

Una problematica correlata è quella della combinazione di un programma libero con codice non libero. Una tale combinazione sarebbe inevitabilmente non libera; ogni libertà che manchi dalla parte non libera mancherebbe anche dall’intero programma. Permettere tali combinazioni aprirebbe non uno spiraglio, ma un buco grosso come una casa. Quindi un requisito essenziale per il permesso d’autore è tappare il buco: tutto ciò che venga aggiunto o combinato con un programma protetto da permesso d’autore dev’essere tale che il programma risultante sia anch’esso libero e protetto da permesso d’autore.

La specifica implementazione di permesso d’autore che utilizziamo per la maggior parte del software GNU è la GNU General Public License (licenza pubblica generica GNU), abbreviata in GNU GPL. Abbiamo altri tipi di permesso d’autore che sono utilizzati in circostanze specifiche. I manuali GNU sono anch’essi protetti da permesso d’autore, ma ne usano una versione molto più semplice, perché per i manuali non è necessaria la complessità della GPL.”

La Free Software Foundation

“Man mano che l’interesse per Emacs aumentava, altre persone parteciparono al progetto GNU, e decidemmo che era di nuovo ora di cercare finanziamenti. Così nel 1985 fondammo la Free Software Foundation (Fondazione per il software libero), una organizzazione senza fini di lucro per lo sviluppo di software libero. La FSF fra l’altro si prese carico della distribuzione dei nastri di Emacs; più tardi estese l’attività aggiungendo sul nastro altro software libero (sia GNU che non GNU) e vendendo manuali liberi.

La FSF accetta donazioni, ma gran parte delle sue entrate è sempre stata costituita dalle vendite: copie di software libero e servizi correlati. Oggi vende CD-ROM di codice sorgente, CD-ROM di programmi compilati, manuali stampati professionalmente (tutti con libertà di redistribuzione e modifica), e distribuzioni Deluxe (nelle quali compiliamo l’intera scelta di software per una piattaforma a richiesta).

⁴Nota di Stallman: Nel 1984 / 1985, Don Hopkins, persona molto creativa, mi mandò una lettera. Sulla busta aveva scritto diverse frasi argute, fra cui questa: “Permesso d’autore - tutti i diritti rovesciati”. Utilizzai l’espressione “permesso d’autore” per battezzare il concetto di distribuzione che allora andavo elaborando.

⁵Nota dei Traduttori: si tratta di un gioco di parole, che qui viene reso con “permesso d’autore”: copyright (diritto d’autore) è formato dalla parola “copy” (copia) e “right” (diritto, ma anche destra), opposto di “left” (sinistra, ma anche lasciato)

4. Open Source

I dipendenti della Free Software Foundation hanno scritto e curato la manutenzione di diversi pacchetti GNU. Fra questi spiccano la libreria C e la shell. La libreria C di GNU è utilizzata da ogni programma che gira su sistemi GNU/Linux per comunicare con Linux. È stata sviluppata da un membro della squadra della Free Software Foundation, Roland McGrath. La shell usata sulla maggior parte dei sistemi GNU/Linux è Bash, la Bourne Again Shell⁶⁷, che è stata sviluppata da Brian Fox, dipendente della FSF.

Finanziammo lo sviluppo di questi programmi perché il progetto GNU non riguardava solo strumenti di lavoro o un ambiente di sviluppo: il nostro obiettivo era un sistema operativo completo, e questi programmi erano necessari per raggiungere quell'obiettivo."

Il supporto per il software libero

"La filosofia del software libero rigetta una diffusa pratica commerciale in particolare, ma non è contro il commercio. Quando un'impresa rispetta la libertà dell'utente, c'è da augurarle ogni successo.

La vendita di copie di Emacs esemplifica un modo di condurre affari col software libero. Quando la FSF prese in carico quest'attività, dovetti trovare un'altra fonte di sostentamento. La trovai nella vendita di servizi relativi al software libero che avevo sviluppato, come insegnare argomenti quali programmazione di Emacs e personalizzazione di GCC, oppure sviluppare software, soprattutto adattamento di GCC a nuove architetture.

Oggi tutte queste attività collegate al software libero sono esercitate da svariate aziende. Alcune distribuiscono raccolte di software libero su CD-ROM, altre offrono consulenza a diversi livelli, dall'aiutare gli utenti in difficoltà, alla correzione di errori, all'aggiunta di funzionalità non banali. Si cominciano anche a vedere aziende di software che si fondano sul lancio di nuovi programmi liberi.

Attenzione, però: diverse aziende che si fregiano del marchio "open source" (software aperto) in realtà fondano le loro attività su software non libero che funziona insieme con software libero. Queste non sono aziende di software libero, sono aziende di software proprietario i cui prodotti attirano gli utenti lontano dalla libertà. Loro li chiamano "a valore aggiunto", il che riflette i valori che a loro farebbe comodo che adottassimo: la convenienza prima della libertà. Se noi riteniamo che la libertà abbia più valore, li dovremmo chiamare prodotti "a libertà sottratta".

Obiettivi tecnici

L'obiettivo principale di GNU era essere software libero. Anche se GNU non avesse avuto alcun vantaggio tecnico su Unix, avrebbe avuto sia un vantaggio sociale, permettendo agli utenti di cooperare, sia un vantaggio etico, rispettando la loro libertà.

Tuttavia risultò naturale applicare al lavoro le regole classiche di buona programmazione; per esempio, allocare le strutture dati dinamicamente per evitare limitazioni arbitrarie sulla dimensione dei dati, o gestire tutti i possibili codici a 8 bit in tutti i casi ragionevoli.

Inoltre, al contrario di Unix che era pensato per piccole dimensioni di memoria, decidemmo di non supportare le macchine a 16 bit (era chiaro che le macchine a 32 bit sarebbero state la norma quando il sistema GNU sarebbe stato completo), e di non preoccuparci di ridurre l'occupazione di memoria a meno che eccedesse il megabyte. In programmi per i quali non era essenziale la gestione di file molto grandi, spingemmo i programmatori a leggere in memoria l'intero file di ingresso per poi analizzare il file senza doversi preoccupare delle operazioni di I/O.

⁶Nota di Stallman: "Bourne Again Shell" è un gioco di parole sul nome di "Bourne Shell", che era la normale shell di Unix.

⁷Nota dei Traduttori: "Bourne Again" richiama l'espressione cristiana "born again", "rinato" (in Cristo).

Queste decisioni fecero sì che molti programmi GNU superassero i loro equivalenti Unix sia in affidabilità che in velocità di esecuzione.”

Donazioni di computer

“Man mano che la reputazione del progetto GNU andava crescendo, alcune persone iniziarono a donare macchine su cui girava Unix. Queste macchine erano molto utili, perché il modo più semplice di sviluppare componenti per GNU era di farlo su di un sistema Unix così da sostituire pezzo per pezzo i componenti di quel sistema. Ma queste macchine sollevavano anche una questione etica: se fosse giusto per noi anche solo possedere una copia di Unix.

Unix era (ed è) software proprietario, e la filosofia del progetto GNU diceva che non avremmo dovuto usare software proprietario. Ma, applicando lo stesso ragionamento per cui la violenza è ammessa per autodifesa, conclusi che fosse legittimo usare un pacchetto proprietario, se ciò fosse stato importante nel crearne un sostituto libero che permettesse ad altri di smettere di usare quello proprietario.

Tuttavia, benché fosse un male giustificabile, era pur sempre un male. Oggi non abbiamo più alcuna copia di Unix, perché le abbiamo sostituite con sistemi operativi liberi. Quando non fu possibile sostituire il sistema operativo di una macchina con uno libero, sostituimmo la macchina.”

L'elenco dei compiti GNU

“Mentre il progetto GNU avanzava, e un numero sempre maggiore di componenti di sistema venivano trovati o sviluppati, diventò utile stilare un elenco delle parti ancora mancanti. Usammo questo elenco per ingaggiare programmatori che scrivessero tali parti, e l'elenco prese il nome di elenco dei compiti GNU. In aggiunta ai componenti Unix mancanti inserimmo nell'elenco svariati progetti utili di programmazione o di documentazione che a nostro parere non dovrebbero mancare in un sistema operativo veramente completo.

Oggi non compare quasi nessun componente Unix nell'elenco dei compiti GNU; tutti questi lavori, a parte qualcuno non essenziale, sono già stati svolti. D'altro canto l'elenco è pieno di quei progetti che qualcuno chiamerebbe “applicazioni”: ogni programma che interessi a una fetta non trascurabile di utenti sarebbe un'utile aggiunta a un sistema operativo.

L'elenco comprende anche dei giochi, e così è stato fin dall'inizio: Unix comprendeva dei giochi, perciò era naturale che così fosse anche per GNU. Ma poiché non c'erano esigenze di compatibilità per i giochi, non ci attenemmo alla scelta di giochi presenti in Unix, preferendo piuttosto fornire un elenco di diversi tipi di giochi potenzialmente graditi agli utenti.”

La licenza GNU per le librerie

“La libreria C del sistema GNU utilizza un tipo speciale di permesso d'autore, la “Licenza Pubblica GNU per le Librerie”⁸, che permette l'uso della libreria da parte di software proprietario. Perché quest'eccezione?

Non si tratta di questioni di principio: non c'è nessun principio che dica che i prodotti software proprietari abbiano il diritto di includere il nostro codice (perché contribuire a un progetto fondato sul rifiuto di condividere con noi?). L'uso della licenza LGPL per la libreria C, o per qualsiasi altra libreria, è una questione di strategia.

La libreria C svolge una funzione generica: ogni sistema operativo proprietario e ogni compilatore includono una libreria C. Di conseguenza, rendere disponibile la nostra libreria C solo

⁸Nota dei Traduttori: Nel 1999 la FSF ha cambiato nome alla licenza LGPL che ora si chiama “Lesser GPL”, GPL attenuata, per non suggerire che si tratti della forma di licenza preferenziale per le librerie.



Figura 4.2.: GNU's not Unix

per i programmi liberi non avrebbe dato nessun vantaggio a tali programmi liberi, avrebbe solo disincentivato l'uso della nostra libreria.

C'è un'eccezione a questa situazione: sul sistema GNU (termine che include GNU/Linux) l'unica libreria C disponibile è quella GNU. Quindi i termini di distribuzione della nostra libreria C determinano se sia possibile o meno compilare un programma proprietario per il sistema GNU. Non ci sono ragioni etiche per permettere l'uso di applicazioni proprietarie sul sistema GNU, ma strategicamente sembra che impedirne l'uso servirebbe più a scoraggiare l'uso del sistema GNU che non a incoraggiare lo sviluppo di applicazioni libere.

Ecco perché l'uso della licenza LGPL è una buona scelta strategica per la libreria C, mentre per le altre librerie la strategia va valutata caso per caso. Quando una libreria svolge una funzione particolare che può aiutare a scrivere certi tipi di programmi, distribuirla secondo la GPL, quindi limitandone l'uso ai soli programmi liberi, è un modo per aiutare gli altri autori di software libero, dando loro un vantaggio nei confronti del software proprietario.

Prendiamo come esempio GNU-Readline, una libreria scritta per fornire a Bash la modificabilità della linea di comando: Readline è distribuita secondo la normale licenza GPL, non la LGPL. Ciò probabilmente riduce l'uso di Readline, ma questo non rappresenta una perdita per noi; d'altra parte almeno una applicazione utile è stata resa software libero proprio al fine di usare Readline, e questo è un guadagno tangibile per la comunità.

Chi sviluppa software proprietario ha vantaggi economici, gli autori di programmi liberi hanno bisogno di avvantaggiarsi a vicenda. Spero che un giorno possiamo avere una grande raccolta di librerie coperte dalla licenza GPL senza che esista una raccolta equivalente per chi scrive software proprietario. Tale libreria fornirebbe utili moduli da usare come i mattoni per costruire nuovi programmi liberi, e costituendo un sostanziale vantaggio per la scrittura di ulteriori programmi liberi."

Togliersi il prurito?

"Eric Raymond afferma che ogni buon programma nasce dall'iniziativa di un programmatore che si vuole togliere un suo personale prurito". È probabile che talvolta succeda così, ma molte parti essenziali del software GNU sono state sviluppate al fine di completare un sistema operativo libero. Derivano quindi da una idea e da un progetto, non da una necessità contingente.

Ad esempio, abbiamo sviluppato la libreria C di GNU perché un sistema di tipo Unix ha bisogno di una libreria C, la Bourne-Again Shell (bash) perché un sistema di tipo Unix ha bisogno di una shell, e GNU tar perché un sistema di tipo Unix ha bisogno di un programma tar⁹. Lo stesso vale per i miei programmi: il compilatore GNU, GNU Emacs, GDB, GNU Make.

⁹Ndr: "tar" come programma di archiviazione.

Alcuni programmi GNU sono stati sviluppati per fronteggiare specifiche minacce alla nostra libertà: ecco perché abbiamo sviluppato gzip come sostituto per il programma Compress, che la comunità aveva perduto a causa dei brevetti sull'algoritmo LZW. Abbiamo trovato persone che sviluppassero LessTif, e più recentemente abbiamo dato vita ai progetti GNOME e Harmony per affrontare i problemi causati da alcune librerie proprietarie (come descritto più avanti).

Stiamo sviluppando la GNU Privacy Guard per sostituire i diffusi programmi di crittografia non liberi, perché gli utenti non siano costretti a scegliere tra riservatezza e libertà.

Naturalmente, i redattori di questi programmi sono coinvolti nel loro lavoro, e varie persone vi hanno aggiunto diverse funzionalità secondo le loro personali necessità e i loro interessi. Tuttavia non è questa la ragione dell'esistenza di tali programmi."

Sviluppi inattesi

"All'inizio del progetto GNU pensavo che avremmo sviluppato l'intero sistema GNU e poi lo avremmo reso disponibile tutto insieme, ma le cose non andarono così.

Poiché i componenti del sistema GNU sono stati implementati su un sistema Unix, ognuno di essi poteva girare su sistemi Unix molto prima che esistesse un sistema GNU completo. Alcuni di questi programmi divennero diffusi e gli utenti iniziarono a estenderli e a renderli utilizzabili su nuovi sistemi: sulle varie versioni di Unix, incompatibili tra loro, e talvolta anche su altri sistemi.

Questo processo rese tali programmi molto più potenti e attirò finanziamenti e collaboratori al progetto GNU; tuttavia probabilmente ritardò di alcuni anni la realizzazione di un sistema minimo funzionante, perché il tempo degli autori GNU veniva impiegato a curare la compatibilità di questi programmi con altri sistemi e ad aggiungere nuove funzionalità ai componenti esistenti, piuttosto che a proseguire nella scrittura di nuovi componenti."

GNU-Hurd

"Nel 1990 il sistema GNU era quasi completo, l'unica parte significativa ancora mancante era il kernel. Avevamo deciso di implementare il nostro kernel come un gruppo di processi server che girassero sul sistema Mach. Mach è un microkernel sviluppato alla Carnegie Mellon University e successivamente all'Università dello Utah; GNU Hurd è un gruppo di server (o "herd of gnu": mandria di gnu) che gira su Mach svolgendo le funzioni del kernel Unix. L'inizio dello sviluppo fu ritardato nell'attesa che Mach fosse reso disponibile come software libero, come era stato promesso.

Una ragione di questa scelta progettuale fu di evitare quella che sembrava la parte più complessa del lavoro: effettuare il debugging del kernel senza un debugger a livello sorgente. Questo lavoro era già stato fatto, appunto in Mach, e avevamo previsto di effettuare il debugging dei server Hurd come programmi utente, con GDB. Ma questa fase si rivelò molto lunga, e il debugging dei server multi-thread che si scambiano messaggi si è rivelato estremamente complesso. Per rendere Hurd robusto furono così necessari molti anni."

Alix

"Originariamente il kernel GNU non avrebbe dovuto chiamarsi Hurd; il suo nome originale era Alix, come la donna di cui ero innamorato in quel periodo. Alix, che era amministratrice di sistemi Unix, aveva sottolineato come il suo nome corrispondesse a un comune schema usato per battezzare le versioni del sistema Unix: scherzosamente diceva ai suoi amici: "qualcuno dovrebbe chiamare un kernel come me". Io non dissi nulla ma decisi di farle una sorpresa scrivendo un kernel chiamato Alix.

4. Open Source

Le cose non andarono così. Michael Bushnell (ora Thomas), principale autore del kernel, preferì il nome Hurd, e chiamò Alix una parte del kernel, quella che serviva a intercettare le chiamate di sistema e a gestirle inviando messaggi ai server che compongono HURD.

Infine io e Alix ci lasciammo e lei cambiò nome; contemporaneamente la struttura di Hurd veniva cambiata in modo che la libreria C mandasse messaggi direttamente ai server, e così il componente Alix scomparve dal progetto. Prima che questo accadesse, però, un amico di Alix si accorse della presenza del suo nome nel codice sorgente di Hurd e glielo disse. Così il nome raggiunse il suo scopo.”

Linux e GNU/Linux

“GNU Hurd non è pronto per un uso non sperimentale, ma per fortuna è disponibile un altro kernel: nel 1991 Linus Torvalds sviluppò un Kernel compatibile con Unix e lo chiamò Linux. Attorno al 1992, la combinazione di Linux con il sistema GNU ancora incompleto produsse un sistema operativo libero completo (naturalmente combinarli fu un notevole lavoro di per sé). È grazie a Linux che oggi possiamo utilizzare una versione del sistema GNU.

Chiamiamo GNU/Linux questa versione del sistema, per indicare la sua composizione come una combinazione del sistema GNU col kernel Linux.”

Le sfide che ci aspettano

“Abbiamo dimostrato la nostra capacità di sviluppare un’ampia gamma di software libero, ma questo non significa che siamo invincibili e inarrestabili. Diverse sfide rendono incerto il futuro del software libero, e affrontarle richiederà perseveranza e sforzi costanti, talvolta per anni. Sarà necessaria quella determinazione che le persone sanno dimostrare quando danno valore alla propria libertà e non permettono a nessuno di sottrargliela. Le quattro sezioni seguenti parlano di queste sfide.”

1. Hardware segreto

“Sempre più spesso, i costruttori di hardware tendono a mantenere segrete le specifiche delle loro apparecchiature; questo rende difficile la scrittura di driver liberi che permettano a Linux e XFree86 di supportare nuove periferiche. Anche se oggi abbiamo sistemi completamente liberi, potremmo non averli domani se non saremo in grado di supportare i calcolatori di domani.

Esistono due modi per affrontare il problema. Un programmatore può ricostruire le specifiche dell’hardware usando tecniche di reverse engineering. Oppure si può scegliere hardware supportato dai programmi liberi: man mano che il nostro numero aumenta, la segretezza delle specifiche diventerà una pratica controproducente.

Il reverse engineering è difficile: avremo programmatori sufficientemente determinati da dedicarsi? Sì, se avremo costruito una forte consapevolezza che avere programmi liberi sia una questione di principio e che i driver non liberi non sono accettabili. E succederà che molti di noi accettino di spendere un po’ di più o perdere un po’ più di tempo per poter usare driver liberi? Sì, se il desiderio di libertà e la determinazione a ottenerla saranno diffusi.”

2. Librerie non libere

“Una libreria non libera che giri su sistemi operativi liberi funziona come una trappola per i creatori di programmi liberi. Le funzionalità attraenti della libreria fungono da esca; chi usa la libreria cade nella trappola, perché il programma che crea è inutile come parte di un sistema operativo libero (a rigore, il programma potrebbe esservi incluso, ma non funzionerebbe, visto che manca la libreria).

Peggio ancora, se un programma che usa la libreria proprietaria diventa diffuso, può attirare altri ignari programmatori nella trappola.

Il problema si concretizzò per la prima volta con la libreria Motif, negli anni '80. Sebbene non ci fossero ancora sistemi operativi liberi, i problemi che Motif avrebbe causato loro erano già chiari. Il progetto GNU reagì in due modi: interessandosi presso diversi progetti di software libero perché supportassero gli strumenti grafici X liberi in aggiunta a Motif, e cercando qualcuno che scrivesse un sostituto libero di Motif. Il lavoro richiese molti anni: solo nel 1997 LessTif, sviluppato dagli *Hungry Programmers*, divenne abbastanza potente da supportare la maggior parte delle applicazioni Motif. Tra il 1996 e il 1998 un'altra libreria non libera di strumenti grafici, chiamata Qt, veniva usata in una significativa raccolta di software libero: l'ambiente grafico KDE.

I sistemi liberi GNU/Linux non potevano usare KDE, perché non potevamo usare la libreria; tuttavia, alcuni distributori commerciali di sistemi GNU/Linux, non scrupolosi nell'attenersi solo ai programmi liberi, aggiunsero KDE ai loro sistemi, ottenendo così sistemi che offrivano più funzionalità, ma meno libertà. Il gruppo che sviluppava KDE incoraggiava esplicitamente altri programmatori a usare Qt, e milioni di nuovi utenti Linux non sospettavano minimamente che questo potesse costituire un problema. La situazione si faceva pericolosa. La comunità del software libero affrontò il problema in due modi: GNOME e Harmony.

GNOME (GNU Network Object Model Environment, modello di ambiente per oggetti di rete) è il progetto GNU per l'ambiente grafico (desktop). Intrapreso nel 1997 da Miguel de Icaza e sviluppato con il supporto di Red Hat Software, GNOME si ripromise di fornire funzionalità grafiche simili a quelle di KDE, ma usando esclusivamente software libero. GNOME offre anche dei vantaggi tecnici, come il supporto per svariati linguaggi di programmazione, non solo il C++. Ma il suo scopo principale era la libertà: non richiedere l'uso di alcun programma che non fosse libero.

Harmony è una libreria compatibile con Qt, progettata per rendere possibile l'uso del software KDE senza dover usare Qt. Nel novembre 1998 gli autori di Qt annunciarono un cambiamento di licenza che, una volta operativo, avrebbe reso Qt software libero. Non c'è modo di esserne certi, ma credo che questo fu in parte dovuto alla decisa risposta della comunità al problema posto da Qt quando non era libero (la nuova licenza è scomoda e iniqua, per cui rimane comunque preferibile evitare l'uso di Qt).

Come risponderemo alla prossima allettante libreria non libera? Riuscirà la comunità in toto a comprendere l'importanza di evitare la trappola? Oppure molti di noi preferiranno la convenienza alla libertà, creando così ancora un grave problema? Il nostro futuro dipende dalla nostra filosofia."

3. Brevetti sul software

"Il maggior pericolo a cui ci troviamo di fronte è quello dei brevetti sul software, che possono rendere inaccessibili al software libero algoritmi e funzionalità per un tempo che può estendersi fino a vent'anni. I brevetti sugli algoritmi di compressione LZW furono depositati nel 1983, e ancor oggi non possiamo distribuire programmi liberi che producano immagini GIF compresse. Nel 1998 un programma libero per produrre audio compresso MP3 venne ritirato sotto minaccia di una causa per violazione di brevetto. Ci sono modi per affrontare la questione brevetti: possiamo cercare prove che un brevetto non sia valido oppure possiamo cercare modi alternativi per ottenere lo stesso risultato. Ognuna di queste tecniche, però, funziona solo in certe circostanze; quando entrambe falliscono un brevetto può obbligare tutto il software libero a rinunciare a qualche funzionalità che gli utenti desiderano. Cosa dobbiamo fare quando ciò accade?

Chi fra noi apprezza il software libero per il valore della libertà rimarrà comunque dalla parte dei programmi liberi; saremo in grado di svolgere il nostro lavoro senza le funzionalità coperte da brevetto. Ma coloro che apprezzano il software libero perché si aspettano che sia tecnicamente superiore probabilmente grideranno al fallimento quando un brevetto ne impedisce lo sviluppo. Perciò, nonostante sia utile parlare dell'efficacia pratica del modello di sviluppo del tipo *bazaar*, e dell'af-

4. Open Source

fidabilità e della potenza di un dato programma libero, non ci dobbiamo fermare qui; dobbiamo parlare di libertà e di principi.”

4. Documentazione libera

“La più grande carenza nei nostri sistemi operativi liberi non è nel software, quanto nella carenza di buoni manuali liberi da includere nei nostri sistemi. La documentazione è una parte essenziale di qualunque pacchetto software; quando un importante pacchetto software libero non viene accompagnato da un buon manuale libero si tratta di una grossa lacuna. E di queste lacune attualmente ne abbiamo molte.

La documentazione libera, come il software libero, è una questione di libertà, non di prezzo. Il criterio per definire libero un manuale è fondamentalmente lo stesso che per definire libero un programma: si tratta di offrire certe libertà a tutti gli utenti. Deve essere permessa la ridistribuzione (compresa la vendita commerciale), sia in formato elettronico che cartaceo, in modo che il manuale possa accompagnare ogni copia del programma.

Autorizzare la modifica è anch'esso un aspetto cruciale; in generale, non credo sia essenziale permettere alle persone di modificare articoli e libri di qualsiasi tipo. Per esempio, non credo che voi o io dobbiamo sentirci in dovere di autorizzare la modifica di articoli come questo, articoli che descrivono le nostre azioni e il nostro punto di vista.

Ma c'è una ragione particolare per cui la libertà di modifica è cruciale per la documentazione dei programmi liberi. Quando qualcuno esercita il proprio diritto di modificare il programma, aumentandone o alterandone le funzionalità, se è coscienzioso modificherà anche il manuale, in modo da poter fornire una documentazione utile e accurata insieme al programma modificato. Un manuale che non permetta ai programmatori di essere coscienziosi e completare il loro lavoro non soddisfa i bisogni della nostra comunità.

Alcuni limiti sulla modificabilità non pongono alcun problema; per esempio, le richieste di conservare la nota di copyright dell'autore originale, i termini di distribuzione e la lista degli autori vanno bene. Non ci sono problemi nemmeno nel richiedere che le versioni modificate dichiarino esplicitamente di essere tali, così pure che intere sezioni non possano essere rimosse o modificate, finché queste sezioni vertono su questioni non tecniche. Restrizioni di questo tipo non creano problemi perché non impediscono al programmatore coscienzioso di adattare il manuale perché rispecchi il programma modificato. In altre parole, non impediscono alla comunità del software libero di beneficiare appieno del manuale.

D'altro canto, deve essere possibile modificare tutto il contenuto tecnico del manuale e poter distribuire il risultato in tutti i formati usuali, attraverso tutti i normali canali di distribuzione; diversamente, le restrizioni creerebbero un ostacolo per la comunità, il manuale non sarebbe libero e avremmo bisogno di un altro manuale.

Gli sviluppatori di software libero avranno la consapevolezza e la determinazione necessarie a produrre un'intera gamma di manuali liberi? Ancora una volta, il nostro futuro dipende dalla nostra filosofia.”

Dobbiamo parlare di libertà

“Stime recenti valutano in dieci milioni il numero di utenti di sistemi quali Debian GNU/Linux e Red Hat Linux. Il software libero ha creato tali vantaggi pratici che gli utenti stanno approdando a esso per pure ragioni pratiche.

Gli effetti positivi di questa situazione sono evidenti: maggior interesse a sviluppare software libero, più clienti per le imprese di software libero e una migliore capacità di incoraggiare le aziende a sviluppare software commerciale libero invece che prodotti software proprietari.

L'interesse per il software, però, sta crescendo più in fretta della coscienza della filosofia su cui è basato, e questa disparità causa problemi. La nostra capacità di fronteggiare le sfide e le minacce descritte in precedenza dipende dalla determinazione nell'essere impegnati per la libertà. Per essere sicuri che la nostra comunità abbia tale determinazione, dobbiamo diffondere l'idea presso i nuovi utenti man mano che entrano a far parte della comunità.

Ma in questo stiamo fallendo: gli sforzi per attrarre nuovi utenti nella comunità sono di gran lunga maggiori degli sforzi per l'educazione civica della comunità stessa. Dobbiamo fare entrambe le cose, e dobbiamo mantenere un equilibrio fra i due impegni."

Open Source

"Parlare di libertà ai nuovi utenti è diventato più difficile dal 1998, quando una parte della comunità decise di smettere di usare il termine *free software* e usare al suo posto *open source*. Alcune delle persone che suggerirono questo termine intendevano evitare che si confondesse *free* con *gratis*, un valido obiettivo. D'altra parte, altre persone intendevano mettere da parte lo spirito del principio che aveva dato la spinta al movimento del software libero e al progetto GNU, puntando invece ad attrarre i dirigenti e gli utenti commerciali, molti dei quali afferiscono a una ideologia che pone il profitto al di sopra della libertà, della comunità, dei principi. Perciò la retorica di *open source* si focalizza sulla possibilità di creare software di buona qualità e potente ma evita deliberatamente le idee di libertà, comunità, principio. Le riviste che si chiamano Linux... sono un chiaro esempio di ciò: sono piene di pubblicità di software proprietario che gira sotto GNU/Linux; quando ci sarà il prossimo Motif o Qt, queste riviste avvertiranno i programmatori di starne lontano o accetteranno la sua pubblicità? L'appoggio delle aziende può contribuire alla comunità in molti modi; a parità di tutto il resto è una cosa utile. Ma ottenere questo appoggio parlando ancor meno di libertà e principi può essere disastroso; rende ancora peggiore lo sbilanciamento descritto tra diffusione ed educazione civica. "Software libero" (*free software*) e *sorgente aperto* (*open source*) descrivono più o meno la stessa categoria di software, ma dicono cose differenti sul software e sui valori. Il progetto GNU continua a usare il termine "software libero" per esprimere l'idea che la libertà sia importante, non solo la tecnologia."

Prova!

"La filosofia di Yoda ("Non c'è provare") suona bene, ma per me non funziona. Ho fatto la maggior parte del mio lavoro angustiato dal timore di non essere in grado di svolgere il mio compito e nel dubbio, se fossi riuscito, che non fosse sufficiente per raggiungere l'obiettivo. Ma ci ho provato in ogni caso perché nessuno tranne me si poneva tra il nemico e la mia città. Sorprendendo me stesso, qualche volta sono riuscito.

A volte ho fallito, alcune delle mie città sono cadute; poi ho trovato un'altra città minacciata e mi sono preparato a un'altra battaglia. Con l'andar del tempo ho imparato a cercare le possibili minacce e a mettermi tra loro e la mia città, facendo appello ad altri hacker perché venissero e si unissero a me.

Oggi giorno spesso non sono da solo. È un sollievo e una gioia quando vedo un reggimento di hacker che scavano trincee per difendere il confine e quando mi rendo conto che questa città può sopravvivere; per ora. Ma i pericoli diventano più grandi ogni anno, e ora Microsoft ha esplicitamente preso di mira la nostra comunità. Non possiamo dare per scontato il futuro della libertà; non diamolo per scontato! Se volete mantenere la vostra libertà dovete essere pronti a difenderla."

4.2. Open Sources Definition

di Bruce Perens

“Il tipico utente di computer possiede una discreta quantità di software che ha comprato nel tempo e che ormai non adopera più. Magari ha aggiornato il computer, o ha cambiato marca, e i vecchi programmi hanno smesso di funzionare. Magari il software è diventato obsoleto. O semplicemente il programma non lo soddisfa. Forse ha comprato due o più computer e non vuole pagare per una seconda copia del software. Quale che sia la ragione, il software per cui ha pagato anni fa non è più adeguato.

Tutto questo è inevitabile? Non sarebbe bello avere diritto a un aggiornamento gratuito ogni volta che il software lo richiede? Se, passando da un Mac a un PC, si potesse cambiare la versione del software gratis? Se, quando il software non funziona o non è abbastanza potente, si potesse migliorarlo o perfino ripararlo da soli? Se il software continuasse a essere supportato anche quando l'azienda produttrice abbia cessato l'attività? Non sarebbe bello usare il software sulla workstation al lavoro, sul computer di casa e sul portatile, anziché su un solo computer? È probabile che, in quel caso, si starebbe ancora usando il software pagato anni prima. Questi sono alcuni dei diritti che l'Open Source riconosce. La Open Source Definition è una carta dei diritti dell'utente di computer. Definisce certi diritti che una licenza software deve garantire per poter essere certificata come Open Source. I produttori che non rendono Open Source il loro programmi trovano difficile competere con chi lo fa, dal momento che gli utenti imparano ad apprezzare quei diritti che avrebbero dovuto sempre essere loro. Programmi come il sistema operativo Linux e il browser Web Netscape sono diventati popolarissimi, scalzando altro software sottoposto a licenze più restrittive. Aziende che usano software Open Source godono il vantaggio del suo rapidissimo sviluppo, spesso a opera cooperativa di numerose aziende, e in gran parte fornito da soggetti individuali che operano miglioriie sulla base di proprie necessità.

I volontari che hanno reso possibili prodotti come Linux ci sono, e le aziende possono cooperare, solo grazie ai diritti che vengono con l'Open Source. Il programmatore medio si sentirebbe stupido nel riversare tanto lavoro in un programma per vedere poi le sue miglioriie vendute dal proprietario senza che lui ne riceva alcun ritorno. Quegli stessi programmatori contribuiscono volentieri all'Open Source perché esso assicura loro questi diritti:

- > il diritto di fare copie del programma e di distribuirle;
- > il diritto d'accesso al codice sorgente del software, condizione necessaria per poterlo modificare;
- > il diritto di apportare migliorie al programma.

Questi diritti sono importanti per coloro che collaborano a un software perché li mantengono tutti al medesimo livello. Chiunque lo voglia è libero di vendere un programma Open Source, così i prezzi rimarranno bassi e sarà rapido lo sviluppo per raggiungere nuovi mercati. Chiunque investa il suo tempo a costruire conoscenza in un programma Open Source lo può supportare, e questo permette agli utenti la possibilità di fornire a loro volta supporto, o l'economia dovuta a un gran numero di fornitori di supporto concorrenti. Qualunque programmatore può adattare un programma Open Source a misura di mercati specifici per raggiungere clienti nuovi. Chi lo fa non è costretto a pagare diritti o concessioni di licenza. La ragione per il successo di una strategia che può suonare alquanto comunista proprio nel momento in cui il fallimento del comunismo stesso è visibile ovunque, è nel fatto che l'economia dell'informazione è sostanzialmente diversa da quella degli altri prodotti. I costi della copia di un'informazione come un programma software è infinitesimo. L'elettricità non costa quasi nulla, l'uso dell'attrezzatura poco di più. È come, per fare un paragone, se si duplicasse una pagnotta usando un solo etto di farina.”

La storia

“Il concetto di free software non è nuovo. Quando le università cominciarono ad adottare i computer, essi erano strumenti per la ricerca. Il software veniva scambiato liberamente e i programmatori venivano pagati per l’atto della programmazione, non per i programmi in sé. Solo più tardi, quando il mondo degli affari e del commercio adottò i computer, i programmatori cominciarono a mantenersi limitando i diritti d’uso del loro software e facendosi pagare per ogni copia. Il free software come idea politica è stato reso popolare da Richard Stallman dal 1984, allorché formò la Free Software Foundation e il progetto GNU a essa collegato. La premessa di Stallman è che la gente dovrebbe avere più libertà e dovrebbe imparare ad apprezzarla. Egli progettò un insieme di diritti che sentiva necessari a ogni utente e li codificò nella GNU Public License o GPL. Stallman battezzò scherzosamente la sua licenza copyleft in quanto lasciava intatto il diritto alla copia. Stallman stesso sviluppò lavori fondamentali di free software quali il compilatore C GNU e GNU Emacs, un editor di testo che alcuni hanno trovato così seducente da concepirne quasi un culto. Il suo lavoro ispirò molti altri a fornire free software sotto la GPL. Per quanto non promossa con il medesimo fervore libertario, la Open Source Definition include molte delle idee di Stallman, e può ben considerarsi un derivato della sua opera.

La Open Source Definition cominciò la sua vita come un documento di linea di condotta della distribuzione Debian GNU/Linux. Debian, uno dei primi sistemi Linux, tuttora popolare, fu costruito interamente con free software. Tuttavia, dal momento che c’erano altre licenze oltre al copyleft che comportavano la gratuità, Debian ebbe qualche problema nel definire che cosa fosse gratis, e i produttori non resero mai chiara la loro politica di free software al resto del mondo. All’epoca, trovandomi a capo del progetto Debian, affrontai questi problemi proponendo un Contratto Sociale Debian e la Guida Debian del Free Software, nel luglio del 1997. Molti sviluppatori Debian inviarono critiche e miglioramenti che io incorporai nei documenti. Il Contratto Sociale documentava l’intenzione di Debian di costituire il proprio sistema interamente con free software, e la Guida rendeva facilmente possibile la classificazione del software come free software o meno, confrontando la licenza software con la guida stessa.

La Guida Debian fu oggetto di molte lodi nella comunità del free software, specialmente fra gli sviluppatori Linux, che a quel tempo stavano preparando la loro propria rivoluzione software sviluppando il primo vero sistema operativo gratuito. Quando Netscape decise di rendere libero il suo browser Web, contattò Eric Raymond. Raymond, la Margaret Mead del free software, è autore di numerosi articoli di taglio antropologico che illustrano il fenomeno del free software e la cultura che vi è cresciuta intorno: scritti che furono i primi di un genere e che hanno messo sotto la luce dei riflettori questo fenomeno fino ad allora oscuro. La dirigenza di Netscape rimase suggestionata in particolare dal saggio di Raymond *La cattedrale e il bazaar*, la cronaca di uno sviluppo free software coronato da successo con volontari non pagati, e gli chiese una consulenza, sotto patto di riservatezza, mentre sviluppavano una licenza per il loro free software. Raymond insisté che la licenza di Netscape dovesse adeguarsi alla guida Debian per poter essere presa sul serio come free software. Raymond e io ci eravamo incontrati qualche volta all’Hacker Conference, una raduno su invito di programmatori creativi e non convenzionali. Avevamo corrisposto via email su vari argomenti. Mi contattò nel febbraio del 1997 con l’idea per l’Open Source. Raymond temeva che la mentalità conservatrice dell’ambiente degli affari venisse scoraggiata dal grado di libertà di Stallman, che era al contrario popolarissimo fra i programmatori di mentalità più liberale. Era impressione di Raymond che ciò stesse sclerotizzando lo sviluppo di Linux nel mondo business laddove esso fioriva invece nell’ambiente della ricerca. Raymond ebbe incontri con uomini d’affari nell’industria Linux che stava muovendo solo allora i primi passi; insieme, essi concepirono un programma di marketing del free software indirizzato ai colletti bianchi. Furono coinvolti Larry Augustin di VA Research e Sam Ockman (che abbandonò più tardi VA per formare Penguin Computing), nonché altri non di mia conoscenza.

4. Open Source

Alcuni mesi prima dell'Open Source, avevo elaborato l'idea dell'Open Hardware, concetto simile rivolto agli strumenti hardware e alle loro interfacce anziché ai programmi software. A tutt'oggi l'Open Hardware non ha avuto il successo dell'Open Source, ma il progetto è ancora attivo; se ne può sapere di più a <http://www.openhardware.org>.

Secondo Raymond, la Guida Debian era il documento più adatto a definire l'Open Source, ma serviva una denominazione più generale e la rimozione dei riferimenti specifici a Debian. Modificai la Guida Debian fino a ricavarne la Open Source Definition. Avevo formato per Debian un ente chiamato Software in the Public Interest, e mi offrii di registrare un marchio per Open Source in modo da poter associare il suo uso alla definizione. Raymond acconsentì, e io registrai una certificazione (una forma speciale di marchio che potesse applicarsi secondo i termini ai prodotti altrui). Circa un mese dopo la registrazione del marchio, apparve chiaro che Software in the Public Interest avrebbe potuto non essere la dimora migliore per il marchio Open Source, e trasferii dunque la proprietà del marchio a Raymond. Raymond e io abbiamo da allora formato la Open Source Initiative, un'organizzazione esclusivamente destinata alla gestione della campagna Open Source e della sua certificazione di marchio. Mentre scrivo, l'iniziativa Open Source è retta da un comitato di sei componenti scelti fra fornitori di free software di chiara fama, e sta cercando di espandere il suo comitato fino a una decina di persone.

Al momento del suo concepimento, la campagna Open Source fu oggetto di molte critiche perfino da parte del contingente Linux che già aveva approvato il concetto di free software. Molti rilevarono che il termine Open Source era già in uso nel ramo della raccolta di dati per le campagne politiche. Altri pensarono che il termine Open fosse già usurato. Per altri ancora era preferibile il nome Free Software, già consolidato. Io opinai che l'abuso del termine Open sarebbe stato sempre meglio dell'ambiguità di free nella lingua inglese, in cui sta a significare tanto libero quanto gratuito, la seconda accezione essendo di gran lunga la più comune nel mondo del commercio di computer e di software. Più tardi, Richard Stallman obiettò alla mancanza di enfasi sulla libertà che secondo lui la campagna dimostrava, e al fatto che, mentre l'Open Source acquistava popolarità, il suo ruolo nella genesi del free software, e quello della sua Free Software Foundation, venivano ignorati: si lamentò di essere stato cassato dalla storia. Peggiorò la situazione la tendenza degli operatori del settore di contrapporre Raymond a Stallman, quasi essi proponessero filosofie concorrenti anziché, sia pur con metodi diversi, propagandare lo stesso concetto. Io stesso contribuì probabilmente a esacerbare gli animi mettendo Stallman e Raymond l'uno contro l'altro in dibattiti pubblici alla Linux Expo e alla Open Source Expo. Caratterizzare i due come avversari diventò un'attività tanto consueta che una discussione via email, non destinata alla pubblicazione, apparve sul periodico on-line Salon. A quel punto, chiesi a Raymond di moderare i toni di un dialogo in cui, per la verità, egli non aveva mai inteso entrare.

Quando la Open Source Definition fu scritta, esisteva già un gran numero di prodotti che potevano rientrare nella categoria. Il problema erano quei programmi che non vi rientravano, e che pure gli utenti trovavano irresistibili."

KDE, Qt e Troll Tech

"Il caso di KDE, Qt e Troll Tech è pertinente a questo saggio perché il gruppo KDE e Troll Tech cercarono di porre un prodotto non-Open Source entro l'infrastruttura di Linux, incontrando una resistenza inattesa. Le grida di pubblico scandalo e la minaccia che il loro prodotto venisse rimpiazzato da un altro, completamente Open Source, convinse alla fine Troll Tech a convertirsi a una licenza pienamente Open Source. È un segno interessante dell'accoglienza entusiastica riservata dalla comunità alla Open Source Definition il fatto che Troll dovette adeguare la propria licenza, pena l'insuccesso del suo prodotto. KDE fu il primo esperimento di un desktop grafico gratuito per Linux. Le applicazioni KDE erano esse stesse sotto GPL, ma dipendevano da una libreria grafica proprietaria nota come Qt, di Troll Tech. I termini della licenza di Qt ne proibivano la modifica

o l'uso con qualunque display software che non fosse il senescente X Window System. Ogni uso diverso richiedeva allo sviluppatore una licenza del costo di 1500 dollari. Troll Tech fornì versioni di Qt per Windows di Microsoft e per Macintosh, e questa fu la sua principale fonte d'entrate. La licenza pseudo-gratuita per i sistemi X intendeva indirizzare i contributi degli sviluppatori Linux verso demo, esempi e accessori per i loro costosi prodotti Windows e Mac.

Per quanto i problemi della licenza di Qt apparissero evidenti, la prospettiva di un desktop grafico per Linux era così attraente che molti utenti furono disposti a chiudere un occhio sulla sua natura non-Open Source. I promotori di Open Source trovarono che KDE fosse in difetto perché avevano l'impressione che gli sviluppatori stessero cercando di confondere la definizione di free software allo scopo di includervi elementi solo parzialmente gratuiti, come Qt. Gli sviluppatori KDE replicarono che i loro programmi erano Open Source, anche se non esistevano versioni eseguibili di quei programmi che non richiedessero una libreria non-Open Source. Io e altri sostenemmo che le applicazioni KDE non erano che frammenti Open Source di programmi non-Open Source, e che una versione Open Source di Qt sarebbe stata necessaria prima che ci si potesse riferire a KDE come a un Open Source.

Gli sviluppatori KDE tentarono di risolvere parzialmente il problema della licenza di Qt negoziando con Troll Tech un accordo (KDE Free Qt Foundation) in cui Troll e KDE avrebbero congiuntamente controllato i rilasci delle versioni gratuite di Qt, e Troll Tech avrebbe rilasciato Qt sotto una licenza conforme a Open Source nel caso che l'azienda venisse acquisita o cessasse l'attività.

Un altro gruppo diede inizio al progetto GNOME, un concorrente interamente Open Source di KDE che mirava a fornire maggiori funzioni e sofisticazioni; un gruppo separato avviò il progetto Harmony per produrre un clone di Qt completamente Open Source che avrebbe supportato KDE. Mentre le dimostrazioni di GNOME avvenivano fra il plauso e Harmony stava per diventare usabile, Troll Tech capì che Qt non avrebbe riscosso successo nel mondo Linux se non avesse cambiato licenza. Troll Tech rilasciò dunque una licenza interamente Open Source per Qt, disinnescando il conflitto ed eliminando i motivi alla base del progetto Harmony. Il progetto GNOME continua tuttora, volto adesso a un KDE migliore in termini di funzionalità e di raffinatezza piuttosto che in termini di licenza.

Prima di rilasciare la sua nuova licenza Open Source, Troll Tech me ne fece avere copia perché la verificassi, con la preghiera che rimanesse riservata finché non fossero in grado di annunciarla.

Nel mio entusiasmo di far pace con il gruppo KDE e in un imbarazzante gesto di autoinganno, preannunciai con otto ore di anticipo la licenza su una mailing list KDE. Quell'email, per il mio rimorso, fu raccolta immediatamente da Slashdot e da altre riviste online.

La nuova licenza Troll Tech è notevole perché approfitta di una scappatoia nella Open Source Definition che permette ai file patch di essere trattati diversamente dall'altro software. Vorrei provvedere a chiudere questa scappatoia in una revisione a venire della Open Source Definition, ma il nuovo testo non dovrebbe tuttavia collocare Qt al di fuori dell'Open Source.

Al momento in cui scrivo, i promotori di Open Source stanno crescendo in misura esponenziale. I recenti contributi Open Source di IBM e di Ericsson hanno fatto i titoli dei giornali. Due distribuzioni Linux, Yggdrasil e Debian, stanno rilasciando distribuzioni di sistemi Linux completi, ivi incluse molte applicazioni che sono interamente Open Source; e molte altre, fra cui Red Hat, ci sono assai vicine. Quando il sistema GNOME sarà completo, sarà stato realizzato un sistema operativo con desktop GUI Open Source in grado di competere con Microsoft NT."

Analisi della Open Source Definition

"Questa sezione presenta nella sua interezza il testo della Open Source Definition, corredata di commenti (in corsivo). La versione canonica della Open Source Definition si trova su:

4. Open Source

<http://www.opensource.org/osd.html>

Alcuni pedanti hanno voluto trovare delle ambiguità di poco conto nella Open Source Definition. Mi sono astenuto da rivederla dal momento che ha poco più d'un anno di vita e vorrei che il pubblico la considerasse stabile. Il futuro imporrà qualche adeguamento lessicale, ma quasi nessuna modifica allo scopo del documento."

La Open Source Definition

"Open Source non significa solo accesso al codice sorgente. I termini di distribuzione di un programma Open Source devono essere consoni ai criteri seguenti.

Si noti che la Open Source Definition non è propriamente una licenza software. È una specifica di quanto è ammesso in una licenza software perché vi si possa riferire come a un'Open Source. La Open Source Definition non è intesa per essere un documento di valore legale. L'inclusione della Open Source Definition nelle licenze software, quale quella proposta per il Progetto di Documentazione di Linux, sembra suggerirmi la stesura di una versione più rigorosa che sia appropriata per quell'uso.

Ai fini dell'Open Source, devono applicarsi insieme tutti i termini che seguono, in tutti i casi. Per esempio, devono applicarsi alle versioni derivate di un programma così come al programma originale. Non è sufficiente applicarne alcune e non altre, e non è sufficiente se i termini non vengono applicati sistematicamente. Dopo aver dovuto affrontare delle interpretazioni particolarmente *semplici* della Open Source Definition, sono tentato di aggiungere: sto dicendo a voi!

1. Ridistribuzione libera

La licenza non può impedire ad alcuna parte in causa la vendita o la cessione del software come componente di una distribuzione di software aggregato che contenga programmi provenienti da sorgenti diverse. La licenza non può richiedere diritti o il pagamento di altre concessioni per tale vendita. Questo significa che potete fare tutte le copie che volete del software e venderle o cederle, e non dovete pagare nessuno per questo privilegio. L'espressione *distribuzione di software aggregato che contenga programmi provenienti da sorgenti diverse* era intesa a chiudere una scappatoia nella Licenza Artistica - una licenza piuttosto malfatta, a mio parere - escogitata in origine per il Perl. Oggi, quasi tutti i programmi che usano la Licenza Artistica sono disponibili anche sotto GPL. Quella clausola non è più necessaria e sarà probabilmente tolta da una futura versione della Open Source Definition.

2. Codice sorgente

Il programma deve includere il codice sorgente e deve consentire la distribuzione tanto in codice sorgente che in forma compilata. Laddove una qualunque forma del prodotto non sia distribuita corredata del codice sorgente, devono essere disponibili mezzi ben pubblicizzati per scaricare il codice sorgente, senza costi addizionali, via Internet. Il codice sorgente deve essere la forma preferenziale nella quale un programmatore modifichi un programma. Codice deliberatamente offuscato non è ammesso. Forme intermedie quali l'output di un preprocessore o di un traduttore non sono ammesse. Il codice sorgente è un preliminare necessario alla riparazione o alla modifica di un programma. L'intento qui è che il codice sorgente sia distribuito con l'opera iniziale e con tutte le opere derivate.

3. Opere derivate

La licenza deve permettere modifiche e opere derivate e deve consentire la loro distribuzione sotto i medesimi termini della licenza del software originale. Il software serve a poco se non se ne può

fare la manutenzione (riparazione dei bug, porting su nuovi sistemi, miglorie) e la modifica è indispensabile alla manutenzione. L'intento è qui di permettere modifiche d'ogni sorta. Deve essere permessa la distribuzione di un'opera modificata sotto gli stessi termini di licenza dell'opera originale. Tuttavia, non è richiesto che ogni produttore di un'opera derivata debba usare gli stessi termini di licenza, ma solo che possa farlo qualora lo voglia. Diverse licenze si esprimono diversamente in materia: la licenza BSD vi permette di mantenere private le modifiche, la GPL no. Alcuni autori di software ritengono che questa clausola possa consentire a persone prive di scrupoli di modificare il loro software in maniera che possa causare imbarazzo all'autore originale. Quello che temono è che qualcuno possa deliberatamente provocare un malfunzionamento del software in modo che l'autore originale appaia un programmatore scadente. Altri paventano un possibile uso criminale del software tramite l'aggiunta di funzioni-cavallo di Troia o di tecnologie illegali in alcuni Paesi, come la crittografia. Tutti questi atti, tuttavia, sono coperti dal codice penale. Un comune fraintendimento a proposito delle licenze è che esse debbano specificare ogni cosa, per esempio *questo software non va usato per compiere delitti*. Dovrebbe tuttavia essere chiaro che nessuna licenza ha esistenza valida al di fuori del corpo del diritto civile e penale. Considerare una licenza come qualcosa separato dal corpo delle leggi applicabili è tanto sciocco quanto considerare un documento in lingua inglese separato dal vocabolario di quella lingua, un caso in cui nessuna parola avrebbe un significato definito.

4. Integrità del codice sorgente dell'autore

La licenza può proibire che il codice sorgente venga distribuito in forma modificata solo se la licenza permette la distribuzione di *patch file* con il codice sorgente allo scopo di modificare il programma al momento della costruzione. Alcuni autori temevano che altri potessero distribuire codice sorgente con modifiche che sarebbero state percepite come opera dell'autore originale e quindi avrebbero potuto gettare ombra su di lui. Questa clausola dà loro un modo di imporre una separazione fra le modifiche e la loro opera, senza proibire le prime. C'è chi considera antiestetico che le modifiche debbano venir distribuite in un file *patch* separato dal codice sorgente, anche se distribuzioni Linux come Debian e Red Hat usano questa procedura per tutte le modifiche apportate ai programmi che distribuiscono. Esistono programmi per riversare automaticamente le patch nel sorgente principale, e questi programmi si possono eseguire automaticamente quando si scompatta un pacchetto di sorgente. Questa clausola, dunque, dovrebbe causare poca o nessuna difficoltà. Si noti anche che questa clausola dice che, nel caso di file patch, la modifica avviene quando si fa il build del programma. Questa scappatoia è impiegata nella Licenza Pubblica di Qt per prescrivere una diversa, anche se meno restrittiva, licenza per i file patch, in contraddizione con la sezione 3 della Open Source Definition. C'è una proposta per chiudere questa scappatoia nella definizione e mantenere nello stesso tempo Qt entro i confini dell'Open Source. La licenza deve permettere esplicitamente la distribuzione di software costruito da codice sorgente modificato. La licenza può richiedere che le opere derivate vadano sotto nome o numero di versione differenti da quelli del software originale. Questo significa che Netscape, per esempio, può insistere per poter essere la sola a chiamare una versione del programma Netscape Navigator (tm), mentre tutte le versioni gratuite del programma debbano chiamarsi Mozilla o in altro modo.

5. Nessuna discriminazione contro persone o gruppi

La licenza non deve discriminare alcuna persona o gruppo di persone. Una licenza fornita dai Rettori dell'Università della California a Berkeley proibiva l'uso di un programma di progettazione elettronica da parte delle forze di polizia del Sud Africa. Apprezzato come merita questo sentimento in tempi di apartheid, va detto che esso non ha più senso oggi. Alcune persone si trovano ancora con software acquistato sotto quella licenza, e le loro versioni derivate devono portare la stessa

4. Open Source

restrizione. Le licenze Open Source non devono contenere tale clausola, indipendentemente dalla nobiltà dell'intento.

6. Nessuna discriminazione di settori

La licenza non deve proibire ad alcuno l'uso del programma in uno specifico campo o per un determinato proposito. Per esempio, non può impedire che il programma venga usato a scopi commerciali o nella ricerca genetica. Il software dev'essere impiegabile allo stesso modo in una clinica che pratichi aborti e in un'organizzazione antiabortista. Queste discussioni politiche sono di pertinenza del Congresso degli Stati Uniti, non delle licenze del software. Alcuni trovano questa mancanza di discernimento gravemente offensiva!

7. Distribuzione della licenza

I diritti relativi al programma devono applicarsi a tutti coloro ai quali il programma sia ridistribuito, senza necessità di esecuzione di una licenza aggiuntiva da parte di questi. La licenza dev'essere automatica, senza la richiesta di alcuna firma. Purtroppo, negli Stati Uniti non ci sono stati validi precedenti giudiziari del potere della licenza senza firma quando questa venga passata da una seconda a una terza parte. Tuttavia, questo argomento considera la licenza come facente parte della legge sul contratto, mentre qualcuno obietta che dovrebbe essere considerata come legge di copyright, campo in cui si danno più precedenti per quel tipo di licenza. Un buon precedente ci sarà senz'altro nei prossimi anni, data la popolarità di questa licenza e il boom dell'Open Source.

8. La licenza non dev'essere specifica a un prodotto

I diritti relativi a un programma non devono dipendere dall'essere il programma parte di una particolare distribuzione software. Se il programma è estratto da quella distribuzione e usato o distribuito entro i termini della licenza del programma stesso, tutte le parti a cui il programma sia ridistribuito dovrebbero avere gli stessi diritti che vengono garantiti in unione alla distribuzione software originale. Questo significa che non si può impedire a un prodotto identificato come Open Source di essere gratuito solo se lo si usa con una marca particolare di distribuzione Linux, ecc. Deve rimanere gratuito se anche lo si separa dalla distribuzione software da cui proviene.

9. La licenza non deve contaminare altro software

La licenza non deve porre restrizioni ad altro software che sia distribuito insieme a quello licenziato. Per esempio, la licenza non deve pretendere che tutti gli altri programmi distribuiti sullo stesso media siano software Open Source. Una versione di GhostScript (programma di rendering PostScript) richiede che i media sui quali viene distribuito contengano solo programmi software gratuiti. Questo non è consentito dalla licenza Open Source. Per fortuna, l'autore di GhostScript distribuisce un'altra versione del programma (un po' più vecchia) sotto una licenza Open Source genuina. Si noti che c'è differenza fra derivazione e aggregazione. Derivazione è quando un programma incorpora di fatto in sé parti di un altro programma. Aggregazione è quando due programmi vengono inclusi sullo stesso CD-ROM. Questa sezione della Open Source Definition riguarda l'aggregazione, non la derivazione. La sezione 4 riguarda la derivazione.

10. Licenze esemplari

Le licenze GNU GPL, BSD, X Consortium e Artistica sono esempi di licenze da considerarsi conformi alla Open Source Definition. Altrettanto dicasi della MPL.

Questo sarebbe una fonte di guai nel giorno in cui una di queste licenze si modificasse e non fosse più Open Source: dovremmo pubblicare immediatamente una revisione della Open Source Definition. Ciò è pertinente per la verità al testo esplicativo, non alla Open Source Definition in sé.”

Analisi delle licenze e loro conformità all’Open Source

“Per comprendere la Open Source Definition dobbiamo esaminare alcune pratiche comuni nelle licenze in quanto si riferiscono all’Open Source.”

Public Domain

“La diffusa convinzione che molto del free software sia di dominio pubblico è errata. Ciò avviene perché l’idea di free software o Open Source confonde molti, che quindi definiscono erroneamente questi programmi come di pubblico dominio perché è il concetto più prossimo a quanto è loro familiare. I programmi, tuttavia, sono molto chiaramente protetti da diritti e sottoposti a licenza: solo, si tratta di una licenza che dà al pubblico più diritti di quelli a cui sia abituato.

Un programma di pubblico dominio è un programma sul quale l’autore abbia rinunciato a tutti i suoi diritti di copyright. Non si può esattamente dire che sia dotato di una licenza; è proprietà personale, per usarlo come meglio si crede. Dal momento che si può trattarlo come personale proprietà, con un programma di pubblico dominio si può fare quello che si vuole. Si può perfino rilicenziare un programma di pubblico dominio, rimuovendo quella versione dal pubblico dominio, o togliendo il nome del suo autore e trattarlo come opera propria.

Se si sta spendendo molto lavoro su un programma di pubblico dominio, si consideri la possibilità di applicarvi il proprio copyright e di rimetterlo sotto licenza. Per esempio, se non si desidera che una terza parte operi delle modifiche che possa poi mantenere riservate, si applichi la GPL o una licenza simile alla propria versione del programma. La versione da cui si è partiti rimarrà nel pubblico dominio, ma la propria versione sarà sotto una licenza che dovrà essere osservata da chi la usa o ne derivi altre.

Un programma di pubblico dominio si rende privato facilmente, dichiarando un copyright e applicandovi la propria licenza, oppure semplicemente dichiarando *Tutti i diritti riservati.*”

Le licenze Free Software in generale

“Se si ha una raccolta di free software come un disco Linux, si potrebbe credere che il programma su quel disco sia proprio. Ma questo non è del tutto vero. I programmi coperti da copyright sono proprietà di chi detiene il copyright, anche quando arrivano con una licenza Open Source come la GPL. La licenza del programma garantisce alcuni diritti, e altri si hanno sotto la definizione di uso corretto nella legge sul copyright.

È importante notare che un autore non deve necessariamente limitarsi a porre una sola licenza su un programma che pubblica. Un programma può essere posto sotto GPL, e una versione può anche essere venduta con una licenza commerciale, non-Open Source. Proprio di questa strategia si valgono molti che desiderano creare un programma Open Source e allo stesso tempo guadagnarci qualche cosa. Chi non vuole una licenza Open Source può pagare per il privilegio, fornendo all’autore una fonte d’entrate.

Tutte le licenze che esamineremo hanno una caratteristica comune: declinano qualunque garanzia. Lo scopo è quello di proteggere il proprietario del software da qualunque responsabilità connessa al programma. Appare una richiesta ragionevole, dato che il programma viene ceduto a costo zero: l’autore non riceve dal programma una fonte d’entrata sufficiente per sostenere un’assicurazione sulle responsabilità ed eventuali spese legali.

4. Open Source

Se gli autori di free software perdessero il diritto di declinare tutte le garanzie e si trovassero a essere citati in tribunale in base alle prestazioni dei programmi che hanno scritto, smetterebbero di fornire software gratuito al mondo. È nel nostro interesse di utenti aiutare gli autori a proteggere questo diritto.”

La GNU General Public License

“Si veda l’appendice B per il testo completo della GPL. La GPL è un manifesto politico tanto quanto è una licenza software, e la maggior parte del testo è inteso a spiegare la motivazione teorica dietro la licenza. Questo dibattito politico ha allontanato alcuni e fornito alcune delle ragioni per cui sono state scritte altre licenze per il free software. Tuttavia, la GPL è stata stilata con l’assistenza di giuristi ed è per questo assai meglio scritta della maggior parte delle licenze di quella famiglia. Io consiglio caldamente di usare la GPL, o la sua variante per librerie LGPL, ogni volta che sia possibile. Se si sceglie un’altra licenza, o se ne stila una nuova, ci devono essere delle buone ragioni per farlo. Chi formula la propria licenza dovrebbe sapere bene che non è un passo da fare con leggerezza. Le complicazioni inaspettate di una licenza affrettata possono affliggere gli utenti di un software per molti anni a venire.

Il testo della GPL non è a sua volta sotto GPL. La sua licenza è semplice: Chiunque può copiare e distribuire copie esatte di questo documento di licenza, ma non ne sono ammesse modifiche. Un punto importante, qui, è che il testo delle licenze di software Open Source di solito non è Open Source esso stesso. Ovviamente, una licenza non potrebbe offrire protezione di alcun tipo se a chiunque fosse consentito apportarvi delle modifiche.

Le clausole della GPL soddisfano la Open Source Definition. La GPL non richiede alcuna delle clausole consentite dal Paragrafo 4 della Open Source Definition. Integrità del codice sorgente dell’autore.

La GPL non permette di mantenere private le modifiche apportate. Le modifiche devono essere distribuite sotto la GPL. In questo modo, l’autore di un programma sotto GPL ha maggiori probabilità di ricevere modifiche da altri, comprese società commerciali che modificano il suo software per i propri scopi.

La GPL non ammette l’incorporazione di un programma sotto GPL in un programma proprietario. La definizione di GPL di programma proprietario lo indica come ogni programma con una licenza che non dia tanti diritti quanti la GPL.

Esistono alcune scappatoie nella GPL che permettono di usarla in un programma non interamente Open Source. Le librerie software che vengono normalmente distribuite con il compilatore o con il sistema operativo che si usa possono essere collegate a software GPL: ne risulta un programma parzialmente libero. Chi detiene il copyright (di norma l’autore del programma) è la persona che mette il programma sotto GPL e ha il diritto di violare la propria licenza. Questa scappatoia è stata usata dagli autori di KDE per distribuire il loro programma Qt prima che Troll Tech ponesse su Qt una licenza Open Source. Tuttavia, questo diritto non si estende ad alcuna terza parte che ridistribuisca il programma: esse devono seguire tutti i termini della licenza, anche quelli che vengono violati dal detentore del copyright, il che rende problematico ridistribuire un programma che contenga Qt sotto GPL.

Gli sviluppatori KDE sembrano inclini a rimediare a questo problema applicando al loro software la LGPL piuttosto che la GPL. La retorica politica presente nella GPL non è gradita a tutti. Non manca chi ha scelto, per il suo software, licenze non altrettanto adatte per semplice avversione alle idee di Richard Stallmann, pur di non aver voluto vederle ripetute nei propri pacchetti software.”

La GNU Library Public License

“La LGPL è un derivato della GPL escogitato per le librerie software. A differenza della GPL, un programma sotto LGPL può venire incorporato entro un programma proprietario. La libreria di linguaggio C fornita con i sistemi Linux è un esempio di software sotto LGPL: essa può essere usata per costruire programmi proprietari, diversamente Linux risulterebbe utile solamente agli autori di free software. Una copia di un programma sotto LGPL può essere convertita in qualunque momento in una sotto GPL. Una volta che ciò succede, quella copia non è più riconvertibile in un programma sotto LGPL, e altrettanto dicasi di qualunque suo derivato. Le rimanenti clausole della LGPL sono simili a quelle della GPL: di fatto essa include la GPL facendovi riferimento.”

Le licenze X, BSD e Apache

“La licenza X e le sue affini BSD e Apache sono molto diverse dalla GPL e dalla LGPL. Queste licenze consentono di fare quasi tutto ciò che si vuole con il software ‘licenziato’ sotto di esse, e questo perché il software originariamente coperto dalle licenze X e BSD era sovvenzionato con sussidi del Governo degli Stati Uniti. Dal momento che i cittadini statunitensi avevano già pagato il software con i soldi delle tasse, fu loro garantito il diritto di fare del software tutto ciò che volessero.

La concessione più importante, assente dalla GPL, è che si può mantenere private le modifiche licenziate sotto licenza X. In altre parole, si può ottenere il codice sorgente di un programma sotto X, modificarlo e poi vendere versioni binarie del programma senza distribuire il codice sorgente delle modifiche e senza applicarvi la licenza X. Tutto ciò rimane comunque Open Source, poiché la Open Source Definition non richiede che le modifiche debbano sempre ricadere sotto la licenza originale.

Molti altri sviluppatori hanno adottato la licenza X e le sue varianti, compresi i progetti BSD (Berkeley System Distribution) e Apache Web server. Un dettaglio molesto della licenza BSD è costituito da una clausola che prescrive che ogni volta si faccia cenno a una caratteristica di un programma sotto BSD in una sua pubblicità, si menzioni (generalmente in una nota a pie di pagina) il fatto che il software è stato sviluppato all’Università della California.

Ora, tener traccia di quale software abbia quella licenza in una cosa immensa come una distribuzione Linux, e ricordare quindi di menzionare l’Università della California ogni volta che uno di questi programmi venga citato in una pubblicità, è un vero mal di testa per i gestori commerciali del progetto. Nel momento in cui scrivo, la distribuzione Debian GNU/Linux contiene oltre 2500 pacchetti software, e se anche solo una piccola parte di essi fosse sotto BSD, la pubblicità per un sistema Linux come Debian dovrebbe contenere molte pagine solo di note! Tuttavia, la licenza dell’X Consortium non ha quella clausola della pubblicità. Se si pensa di usare una licenza tipo BSD si usi invece una licenza X.”

La Licenza Artistica

“Sebbene questa licenza sia stata in origine sviluppata per il Perl, è stata dopo allora adoperata per altro software. A mio parere si tratta di una licenza formulata con grande sciattezza, in quanto impone dei requisiti e fornisce poi delle scappatoie che rendono facile aggirarli. Forse è questa la ragione per cui quasi tutto il software sotto Licenza Artistica, ha oggi una seconda licenza, offrendo la scelta fra la Licenza Artistica e la GPL.

La Sezione 5 della Licenza Artistica vieta la vendita del software, ma permette che sia venduta una distribuzione di software aggregato di più di un programma. In questo modo, se raggruppa- te un programma sotto Licenza Artistica con un `helloworld.c` di cinque righe di codice, potete vendere il bundle. Questa caratteristica della Licenza Artistica è stata la sola causa della scappatoia dell’aggregato nel primo paragrafo della Open Source Definition. Dal momento che l’uso della Licenza Artistica è in netto declino, stiamo pensando di togliere quella scappatoia. Ciò renderebbe la

4. Open Source

Licenza Artistica una licenza non-Open Source. Non è questo un passo che faremo leggermente, e ci vorrà probabilmente più di un anno di riflessione e di dibattito prima che questo accada.

La Licenza Artistica richiede che le modifiche siano rese gratuite, ma fornisce poi una scappatoia (nella Sezione 7) che permette di mantenerle private e perfino di porre sotto dominio pubblico parti del programma sotto Licenza Artistica!”

La Netscape Public License e la Mozilla Public License

“La NPL è stata sviluppata da Netscape quando rese Open Source il suo prodotto Netscape Navigator. Per la precisione, la versione Open Source si chiama Mozilla; Netscape si riserva il marchio Navigator per il suo prodotto. Eric Raymond ed io agimmo come consulenti a titolo gratuito durante lo sviluppo di questa licenza. Io cercai, senza successo, di persuadere Netscape a usare la GPL, e quando essa declinò, contribuì a comporre una licenza che si conformasse alla Open Source Definition.

Una caratteristica importante della NPL è che contiene privilegi speciali che si applicano a Netscape e a nessun altro. Essa dà a Netscape il privilegio di rilicenziare le modifiche fatte al suo software. Netscape può mantenere private quelle modifiche, migliorarle, e rifiutarsi di restituire il risultato. Questa clausola si è resa necessaria perché, quando Netscape decise per l’Open Source, aveva contratti con altre aziende che la impegnavano a fornir loro Navigator sotto una licenza non Open Source. Netscape ha creato la MPL o Mozilla Public License per rimediare a questa situazione. La MPL è molto simile alla NPL, ma non contiene la clausola che permette a Netscape di rimettere le modifiche sotto licenza.

La NPL e la MPL consentono di mantenere private le modifiche apportate.

Molte aziende hanno adottato per i loro programmi una variante della MPL. Non è una buona cosa, perché la NPL è stata progettata per la particolare situazione contrattuale in cui Netscape si trovava nel momento in cui la licenza veniva scritta, e non è detto che sia altrettanto adatta a usi diversi. Dovrebbe restare la licenza di Netscape e di Mozilla, e altri dovrebbero usare le licenze GPL o X.”

Scegliere una licenza

“Non conviene formulare una licenza nuova se è possibile usarne una di quelle qui elencate. La propagazione di molte licenze diverse e incompatibili opera a detrimento del software Open Source, perché frammenti di un programma non possono essere usati in un altro programma sotto licenza incompatibile.

Ci si tenga alla larga dalla Licenza Artistica, a meno che non si intenda studiarla a fondo ed eliminarne le scappatoie. Fatto ciò, è tempo di prendere delle decisioni.

1. Si vuole che il pubblico possa mantenere private le modifiche, o no? Se si vuole che chi ha apportato modifiche al proprio software ne rimandi il codice sorgente, si applichi una licenza che lo prescriva. La GPL e la LGPL sono delle buone scelte. Se non dispiace che il pubblico mantenga private le modifiche, si usino la licenza X o la licenza Apache.
2. Si vuole consentire a qualcuno di far confluire il proprio programma nel suo software proprietario? Se sì, si usi la LGPL, che lo permette esplicitamente senza consentire al pubblico di rendere privato il codice, oppure si usi la licenza X o Apache, che permettono che le modifiche siano mantenute private.
3. Si desidera che chi lo voglia possa comprare sotto licenza commerciale versioni non Open Source del proprio programma? Se sì, si doti il software di doppia licenza. Io consiglio la GPL

Tipologie	A	B	C	D
GPL				
LGPL	•			
BSD	•	•		
NPL	•	•		•
MPL	•	•		
Dominio Pubblico	•	•	•	

- A** Può essere miscelato con software commerciale
- B** Le modifiche possono essere mantenute private e non restituite all'autore originale
- C** Può essere ri-licenziato da chiunque
- D** Contiene privilegi speciali sulle modifiche per chi detiene il copyright originale

Tabella 4.1.: Caratteristiche delle licenze software.

come licenza Open Source; si può trovare una licenza commerciale adatta all'uso in libri come *Copyright Your Software* edito da Nolo Press.

4. Si vuole che chiunque usi il proprio software debba pagare per il privilegio? Se le cose stanno così, forse l'Open Source non è adatta. Se basta che solo alcune persone paghino, si può mantenere Open Source il programma. La maggior parte degli autori Open Source considerano i loro programmi come contributi al bene pubblico, e non badano al fatto di essere pagati oppure no.

Per un quadro chiaro consultare la tabella 4.1 a pagina 55.

Il Futuro

“Al momento in cui questo saggio andava in stampa, IBM entrava nel mondo Open Source e la comunità dei venture capital lo sta scoprendo. Intel e Netscape hanno investito in Red Hat, un distributore Linux. VA Research, integratore di server Linux e hardware per workstation, ha annunciato l'ingresso di un investitore esterno. Sendmail Inc., creata per commercializzare l'onnipresente programma di posta elettronica Sendmail, ha annunciato la disponibilità di fondi per sei milioni di dollari.

L'applicazione di posta protetta Postfix di IBM ha una licenza Open Source, e un altro prodotto IBM, il compilatore Java Jikes, ha una licenza che, nell'istante in cui scrivo, mira, per il momento con parziale successo, a soddisfare le specifiche dell'Open Source Definition. Parrebbe che IBM intenda modificare la licenza di Jikes perché sia per intero Open Source, e che a questo scopo stia raccogliendo pareri nella comunità.

Due promemoria interni della Microsoft, noti sono il nome di Halloween Document, sono trapelati al pubblico online. Questi promemoria mostrano in modo inequivocabile come Microsoft si senta minacciata da Open Source e da Linux, e che Microsoft lancerà un'offensiva contro di loro per proteggere i suoi mercati. E' chiaro che dobbiamo prepararci a vivere tempi interessanti. Credo che vedremo Microsoft usare due principali strategie: interfacce sotto copyright e brevetti. Microsoft estenderà i protocolli di rete, che contengono caratteristiche proprietarie Microsoft in quelli che non verranno resi disponibili al free software. Essa, con altre aziende, farà ricerca aggressivamente in nuove direzioni dell'informatica e brevetterà tutto quanto potrà prima che si possano sfruttare quelle tecniche nel free software; quindi ci chiuderà fuori con le concessioni sui diritti di brevetto.

4. Open Source

Sono autore di un saggio per la webzine Linux World su come si possano battere i nemici dell'Open Source sul fronte dei brevetti.

La buona notizia è che Microsoft si è spaventata! Nel secondo degli Halloween Document, un membro dello staff Microsoft racconta della sua sensazione d'euforia nel vedere come poteva modificare facilmente parti del sistema Linux perché facesse esattamente quello che voleva, e com'era più facile per un impiegato Microsoft fare questo su Linux di quanto non lo fosse modificare NT. I tentativi di nuocerici provenienti dall'interno sono i più pericolosi. Credo che vedremo altri sforzi per diluire la definizione di Open Source fino a includervi prodotti parzialmente gratuiti, come abbiamo visto avvenire con la libreria Qt in KDE prima che Troll Tech vedesse la luce e rilasciasse una licenza Open Source. Microsoft e altri potrebbero danneggiarci rilasciando un sacco di software gratuito quel tanto da attrarre utenti, ma senza avere le piene libertà dell'Open Source. Non è impensabile che essi possano stroncare lo sviluppo di certe categorie di software Open Source rilasciando soluzioni abbastanza valide, abbastanza quasi-gratis. Tuttavia, la forte reazione che si è avuta contro il progetto KDE prima che la libreria Qt divenisse completamente Open Source, non è di buon augurio per imprese analoghe di Microsoft e compagnia.

Finora abbiamo scampato i cavalli di Troia. Supponiamo che qualcuno che ci vuol male fornisca del software che contiene un cavallo di Troia - un espediente per sconfiggere la protezione in un sistema Linux. Supponiamo, poi, che la medesima persona resti in attesa che il software con il cavallo di Troia sia largamente distribuito e quindi ne pubblicizzi la vulnerabilità agli attacchi alla sicurezza. Il pubblico si sarà per allora accorto che il nostro sistema Open Source può lasciarci più vulnerabili a questa sorta di attacchi che non il sistema chiuso di Microsoft; questo potrebbe ridurre la fiducia generale nel software Open Source. Potremmo obiettare che Microsoft ha la sua parte di bug di sicurezza anche se non lascia possibilità di inserirli a persone esterne e che il modello a codice sorgente aperto dell'Open Source rende più facile scoprire questi bug. Qualunque bug del genere che comparisse in Linux sarebbe riparato il giorno dopo essere stato scoperto, mentre un omologo in Windows o non sarebbe mai scoperto o dovrebbe aspettare il rimedio per anni. Ma dobbiamo rinforzare ancora la nostra difesa contro i cavalli di Troia. Identificare con sicurezza chi contribuisce alla creazione di software e delle modifiche è la difesa migliore di cui disponiamo, dal momento che ci permette di valerci del diritto penale contro chi escogita cavalli di Troia. Quando ero dirigente della distribuzione GNU/Linux di Debian, istituimmo un sistema che consentiva di identificare in modo affidabile tutti i manutentori del software e permetteva loro di partecipare a loro volta a una rete a crittografia a chiave pubblica che ci avrebbe consentito di verificare da chi proveniva il nostro software. Questo tipo di sistema si deve espandere fino a comprendere tutti gli sviluppatori Open Source.

Enormi sono i miglioramenti da intraprendere prima che Linux sia davvero alla portata dell'utente medio. L'interfaccia grafica per gli utenti è chiaramente qualcosa che manca, e a questo sono rivolti i progetti KDE e GNOME. La prossima frontiera è l'amministrazione di sistema, Linuxconf vi sta parzialmente provvedendo, ma si trova ben lungi dall'essere uno strumento completo d'amministrazione di sistema per l'utente sprovvisto. Se il sistema COAS di Caldera avrà successo, potrebbe diventare la base per una soluzione completa al problema dell'amministrazione di sistema. Tuttavia, Caldera ha avuto dei problemi nel mantenere un'allocazione di risorse sufficienti a COAS per terminarne lo sviluppo, e altri sviluppatori hanno abbandonato la partita perché non notavano progressi. La pleora di distribuzioni Linux appare oggi in pieno rivolgimento, con Red Hat percepita come vincitrice e Caldera come seconda. Red Hat ha mostrato finora un solido impegno verso il concetto di Open Source, ma un nuovo presidente e voci di un'offerta pubblica iniziale (Initial Public Offering, IPO) potrebbero significare un indebolimento di quest'impegno, specialmente se concorrenti come Caldera, molto più tiepidi verso l'Open Source, riusciranno a inserirsi nel mercato di Red Hat. Se l'impegno delle distribuzioni Linux commerciali verso l'Open Source diventerà problematico, questo genererà probabilmente uno sforzo per rimpiazzarle con tentativi

Open Source simili al GNU/Linux di Debian ma più diretti al mercato commerciale di quanto non sia stata Debian. Malgrado queste sfide, io predico la vittoria dell'Open Source. Linux è divenuto strumento di test per gli studenti d'informatica, che, una volta laureati, porteranno con sé quegli strumenti nei loro posti di lavoro. Molti laboratori di ricerca hanno adottato il modello Open Source in quanto la condivisione delle informazioni è essenziale al metodo scientifico, e l'Open Source consente al software di essere condiviso facilmente. Il mondo business sta adottando il modello Open Source perché consente a gruppi di aziende di collaborare nella risoluzione di un problema senza la minaccia di una causa anti-trust, e per l'impulso di cui gode quando i contributi pubblici di programmazione rendono gratuite le migliorie al software. Alcune grandi società hanno adottato l'Open Source come strategia per combattere e scongiurare l'avvento di un'altra Microsoft pronta a dominare il settore informatico. Ma l'indicazione più affidabile sul futuro dell'Open Source viene dal suo passato: in pochi anni, dal niente siamo arrivati ad avere un robusto corpus di software che è in grado di risolvere tanti problemi diversi e che si avvia a raggiungere il milione di utenti. Non c'è ragione di rallentare la corsa proprio adesso."

4. *Open Source*

Parte III.
GNU/Linux

5. Il fenomeno GNU/Linux

GNU/Linux è una libera e distribuibile versione di UNIX sviluppata da Linus Torvalds presso l'Università di Helsinki in Finlandia. GNU/Linux è un *melting pot* di conoscenze ed esperienze che hanno trovato in quest'ultimo una piazza virtuale dove crescere rapidamente. All'interno del sistema operativo non viene utilizzato in nessun modo codice licenziato da enti commerciali e buona parte del software che ruota attorno segue questa "corrente" di pensiero abbracciata dal Progetto GNU della Free Software Foundation¹. Linux infatti rappresenta unicamente il kernel mentre GNU/Linux sarebbe la giusta formulazione per indicare il kernel di Linux e tutto il software di gestione offerto dal progetto GNU che rende un sistema operativo utilizzabile. Per maggiori informazioni in merito consiglio il documento disponibile sul sito GNU - <http://www.gnu.org/gnu/linux-and-gnu.html>.²

5.1. La nascita

Il kernel di Linux è stato originariamente sviluppato da Linus Torvalds. Il progetto iniziale era ispirato a Minix, un piccola versione di UNIX sviluppata da Andy Tanenbaum, e proprio nel gruppo di discussione comp.os.minix venne rilasciato il seguente messaggio da Torvalds:

"After that it was plain sailing: hairy coding still, but I had some devices, and debugging was easier. I started using C at this stage, and it certainly speeds up development. This is also when I start to get serious about my megalomaniac ideas to make 'a better Minix than Minix'. I was hoping I'd be able to recompile gcc under Linux some day... Two months for basic setup, but then only slightly longer until I had a disk-driver (seriously buggy, but it happened to work on my machine) and a small filesystem. That was about when I made 0.01 available [around late August of 1991]: it wasn't pretty, it had no floppy driver, and it couldn't do much anything. I don't think anybody ever compiled that version. But by then I was hooked, and didn't want to stop until I could chuck out Minix."

Nessun annuncio venne fatto per la versione 0.01. I sorgenti di quest'ultimo non erano eseguibili e contenevano unicamente le rudimentali sorgenti del kernel da compilare su un server Minix. Nell'ottobre 1991 venne annunciata la versione 0.02 di Linux, la prima versione ad essere definita "ufficiale". Quest'ultima rimaneva pur sempre scarna ma includeva la shell dei comandi bash (GNU Bourne Again Shell) e gcc, il compilatore GNU. Era il primo ed importante passo ed un'intera comunità di sviluppatori abbracciò il progetto.

Linus scrisse su comp.os.minix:

"Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers? Are you sure without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on Minix? No more all-nighters to get a nifty program working? Then this post might be just for you. As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers."

¹Cambridge, Massachusetts.

²Si ringrazia Bradley M. Kuhn della Free Software Foundation per aver consigliato la presente nota nel manuale.

5. Il fenomeno GNU/Linux



Figura 5.1.: Linus Torvalds fotografato dopo un talk.

It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02... but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it."

Dopo la versione 0.03 ci fu un rapido susseguirsi di versioni grazie alla natura del progetto che concentrava l'opera di una crescente comunità telematica. La versione 1.0 venne rilasciata nel dicembre del 1993 dopo essere considerata stabile e bug-free, ovvero senza problemi. Il kernel di Linux rilasciato era compatibile con molti standard UNIX come IEEE POSIX.1, System V e BSD e poteva "funzionare" su processori di classe i386, disponibili in gran parte dei personal computer esistenti.

Le frasi "Linux is not Unix" o "GNU is not Unix" hanno un senso; il marchio UNIX è licenziato da X/Open che su richiesta e a pagamento valuta un sistema operativo e definisce o meno l'utilizzo del nome "UNIX". Linux è il paradosso perché malgrado la mancanza del "titolo" ha saputo imporsi ugualmente e diventare la linea guida nello sviluppo software per sistemi UNIX, scavalcando per propria natura ogni limite relativo a licenze con particolari note restrittive.

Dopo circa 8 anni dal messaggio inizialmente riportato lo spirito di Torvalds non è affatto cambiato. Il 10 Marzo 2000 scriveva:

"I just made a 2.3.51 release, and the next kernel will be the first of the pre-2.4.x kernels. That does NOT mean that I'll apply a lot of last-minute patches: it only means that I'll let 2.3.51 be out there over the weekend to hear about any embarrassing problems so that we can start the pre-2.4 series without the truly stupid stuff. There's some NFSv3 and other stuff pending, but those who have pending stuff should all know who they are, and for the rest it's just time to say nice try, see you in 2.5.x. The pre-2.4.x series will probably go on for a while, but these are the bug fixes only trees. These are also the I hope a lot of people test them trees, because without testing we'll never get to the eventual goal, which is a good and stable 2.4.x in the reasonably near future."

"I hope a lot of people test them", ovvero la versione "spero che molta gente la testi" per riportare eventuali mancanze o errori da correggere prima della versione finale. Ironia e modestia, le chiavi di successo. In un messaggio ironico spedito il primo giorno di Aprile 2000, Torvalds scriveva:

“Dear Linux Users,

I’m pleased to announce jointly with Microsoft(tm)(r) Corporation release of Linux 2000 Professional Enterprise. As you probably already know I’m busy with my family and I already have full-time job with Transmeta. Thus, it has been necessary for me to look for some responsible partner who would help me develop Linux. After extensive search, I have decided upon Microsoft Corporation which has been known on market for long time from their high quality software. Thus the upcoming Linux 2.4.0 will become Linux 2000(tm)(r). Pricing will be determined at later time. However, I would like to take opportunity now to remind people who have unlicensed version of Linux to delete it from hard disk and then wait until official release of Linux 2000(tm)(r) will become available. Effective April 1st 2000, midnight, all older versions of Linux are illegal under Digital Millennium Copyright Act.”

Un bel “pesce d’aprile”! :)

L’annuncio della versione 2.4.0.test10-final fatto da Torvalds presentava quanto segue:

From: Linus Torvalds

Subject: Linux-2.4.0-test10

Date: Tue, 31 Oct 2000 12:41:55 -0800 (PST)

Ok, test10-final is out there now. This has no known bugs that I consider show-stoppers, for what it’s worth.

And when I don’t know of a bug, it doesn’t exist. Let us rejoice. In traditional kernel naming tradition, this kernel hereby gets anointed as one of the “greased weasel” kernel series, one of the final steps in a stable release.

We’re still waiting for the Vatican to officially canonize this kernel, but trust me, that’s only a matter of time. It’s a little known fact, but the Pope likes penguins too.

Linus

Effetto “giubileo”.

Un pinguino come amico

Linux viene spesso rappresentato da un pinguino che è diventata la mascotte ufficiale del sistema operativo. L’immagine ufficiale (figura 5.2) è stata creata da Larry Ewing³.

Il nome di questo amichevole pinguino, Tux, è stato suggerito da James Hughes in un messaggio apparso sulla mailing-list ufficiale del kernel di linux in data 10 giugno 1996. Tux può essere interpretato come l’abbreviazione di Torvalds UniX.

5.2. Distribuzioni

GNU/Linux si può scaricare gratuitamente dalla rete oppure lo si può ottenere tramite le “distribuzioni” esistenti, nate per diffondere il sistema operativo su supporti fisici come floppy o cd-rom e colmare la mancanza di connessioni in rete stabili e veloci.

³<http://www.isc.tamu.edu/~lewing/linux/>

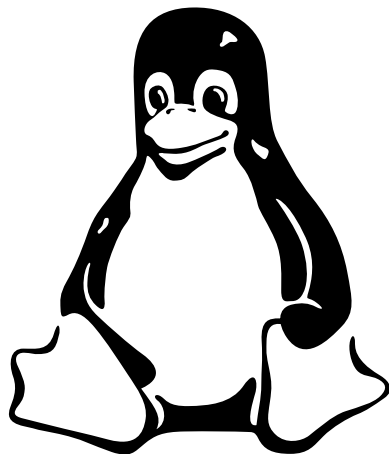


Figura 5.2.: Tux, un pinguino come amico

La scelta giusta

Le sostanziali differenze tra le distribuzioni nascono con l'aggiunta di ulteriore software per l'installazione e per la gestione, supporto tecnico nonché materiale cartaceo come manuali e documentazione di vario genere. E' difficile valutare questo valore aggiunto e stabilire quale distribuzione possa ritenersi migliore di un'altra visto che ognuna ha una serie di caratteristiche proprie che la distinguono sotto l'aspetto della sicurezza, della gestione o della facilità d'uso.

Si potrebbe produrre una immensa serie di motivazioni e considerazioni ma il consiglio che si vuole dare è quello di leggere le note sottostanti e fare una scelta per iniziare:

- Se dovete gestire un server Internet oppure godete di connettività fissa la scelta migliore è la distribuzione Debian (logotipo ufficiale nell'immagine 5.3⁴), molto curata sotto gli aspetti della sicurezza e facilmente aggiornabile in rete con pochi semplici comandi. E' la distribuzione che attualmente utilizzo, riscontrando limiti e pregi. Può essere una distribuzione "difficile" per un utente che si affaccia per la prima volta ad un sistema Unix ma una volta superate le barriere iniziali si possono "gustare" i pregi. Rimane in ogni caso la scelta migliore per gli sviluppatori software, gli ambiti scientifici e gli Internet Service Provider.
- Se dovete utilizzare GNU/Linux come workstation di lavoro una buona scelta può ricadere sulle distribuzioni Red Hat, Mandrake o SuSE, facili da installare per una persona alle prime armi e semplici da gestire con il sistema di gestione dei pacchetti software denominato rpm, ampiamente presentato in questo manuale.
- Se infine rientrate nella categoria "smanettoni" (parola che può suscitare sorrisi ma che è comune e largamente utilizzata) potreste trovare la vostra panacea nella distribuzione Slackware, soluzione ideale per chi desidera conoscere il proprio sistema nei minimi particolari.

I nomi indicati sono sostanzialmente quelli delle distribuzioni maggiormente utilizzate ed è facile trovare, oltre alle distribuzioni per processori della famiglia i386 (Intel e compatibili), versioni per processori di diverso genere come Alpha, Sparc, PPC.

⁴Debian Open Use Logo License, Copyright ©1999 Software in the Public Interest. This logo or a modified version may be used by anyone to refer to the Debian project, but does not indicate endorsement by the project. It is not required, but we would appreciate, that when used on a web page, the image be a link to <http://www.debian.org/>



Figura 5.3.: Debian GNU/Linux - <http://www.debian.org>

Dove trovare le distribuzioni

La diffusione di GNU/Linux lo ha reso ambito e acclamato tanto da rendere molto semplice poter trovare una distribuzione completa del sistema operativo. Potete informarvi presso una qualsiasi attività commerciale che distribuisce materiale informatico o passare direttamente in edicola per acquistare una delle tante riviste che allegano cd-rom con GNU/Linux. Numerosi anche i rivenditori online, tra i quali indico i seguenti:

Rivenditori internazionali

- > Infomagic - <http://www.infomagic.com>
- > Cheapbytes - <http://www.cheapbytes.com>

Sempre più frequentemente le distribuzioni diffuse attivano e-commerce dai propri siti Internet e quindi è consigliabile visitare direttamente quest'ultimi per le proprie necessità:

- > Caldera - <http://www.caldera.com>
- > Corel - <http://linux.corel.com>
- > Debian - <http://www.debian.org>
- > Mandrake - <http://www.linux-mandrake.com>
- > LinuxPPC - <http://www.linuxppc.org>
- > Red Hat - <http://www.redhat.com>
- > Slackware - <http://www.cdrom.com>
- > Stampede - <http://www.stampede.org>
- > S.u.S.E. - <http://www.suse.com>
- > Turbo Linux - <http://www.pht.com>
- > VA Research - <http://www.valinux.com>

5.3. Documentazione

Una caratteristica importante nel mondo GNU/Linux è la memoria storica. Esiste un'incredibile mole di documentazione disponibile in rete e pronta a risolvere ogni singolo problema, una quantità talmente elevata di informazioni che spesso un nuovo utente si trova disorientato. Analizzeremo di seguito le varie tipologie informative presenti e i metodi più rapidi per la consultazione.

Pagine Manuali / Guide in linea

Rappresentano una forma molto comoda per consultare informazioni relative a programmi installati nel sistema. Hanno una organizzazione sostanzialmente comune che presenta un'area descrittiva, dove viene spiegato lo scopo del programma, una sezione relativa alla sintassi d'uso del comando con le varie opzioni, consigli di vario genere e comandi eventuali logicamente correlati.

Per poter consultare queste guide è necessario utilizzare il comando `man` nel seguente modo:

```
$ man nomecomando
```

Per uscire dalla visualizzazione della guida in linea è necessario digitare il tasto `q`.

How-To

Sono manuali orientati a specifici argomenti; esistono How-to per una moltitudine di argomenti diretti a risolvere particolari e ben determinate problematiche. Esiste da anni un progetto che cura la gestione razionale di questi manuali che si chiama LDP⁵, <http://www.linuxdoc.org/>.

Il progetto è nello spirito tipico che contraddistingue GNU/Linux, ovvero opera di volontari che impiegano parte del proprio tempo per lasciare le loro esperienze a chi ne necessiterà successivamente. Siccome la gran parte della documentazione informatica nasce e viene sviluppata in lingua inglese si è formato un gruppo *parallelo* chiamato ILDP⁶, con il particolare scopo di tradurre la documentazione esistente in lingua italiana.

Mailing List

Esistono numerose mailing list, strumenti che permettono ad una molteplicità di persone di *comunicare* tramite e-mail e condividere queste informazioni. Esistono mailing list per progetti software ben determinati e mailing list create da gruppi di utenti GNU/Linux di carattere generico. Se desiderate conoscere le argomentazioni di ogni lista e trovare le modalità di iscrizione si consiglia di visitare i seguenti siti Internet:

- > ErLug (Emilia Romagna Linux User Group) - <http://erlug.linux.it>
- > ILS (Italian Linux Society) - <http://www.linux.it>
- > Pluto - <http://www.pluto.linux.it>

I siti indicati sono fonti informative di vario genere da considerare come punti di partenza nella rete Internet per ampliare le proprie conoscenze.

⁵Linux Documentation Project.

⁶Italian Linux Documentation Program - <http://www.pluto.linux.it/ILDP/>

Newsgroup

Esiste una serie di gruppi di discussione in lingua italiana e fanno parte della gerarchia `it.comp.os.linux` divisi in macrocategorie a seconda degli scopi finali degli utenti e visualizzabili con un comune programma client per i newsgroup.

Ulteriore Documentazione Italiana

Nello spirito che contraddistingue GNU/Linux, esistono numerose forme documentative curate da volontari e disponibili gratuitamente in rete.

Appunti Linux

Appunti Linux di Daniele Giacomini rappresenta un corposo insieme di informazioni utili per gestire le attività del vostro personale server GNU/Linux. La documentazione copre argomenti relativi all'amministrazione di sistema, la programmazione e i servizi Internet.

<http://www.swlibero.org/>

LDR

Linux Domanda e Risposta di Gaetano Paolone, rappresenta una interessante raccolta di F.A.Q. particolarmente indirizzate all'utente novizio e di medio livello. Può essere molto utile per la risoluzione di alcune problematiche e per imparare nuove soluzioni per l'ottimizzazione del sistema.

<http://web.tiscalinet.it/linuxfaq/>

5.4. Valore aggiunto

Le persone che spesso utilizzano strumenti informatici spesso si trovano a *combattere* con soluzioni inadeguate alle proprie esigenze anche se ampiamente diffuse commercialmente. E' necessario capire che non sempre il prodotto hardware o software diffuso e famoso coincide con un prodotto di qualità. Spesso sono politiche di marketing e comunicazione a *convincere* le persone che un prodotto è valido anche se le sue caratteristiche sono inferiori a quanto promesso.

“Se dovete mangiare entrate in un supermercato e scegliete tra i prodotti presenti. Ogni prodotto, essendo un bene alimentare, presenta una etichetta dove è esposta una varietà di dati importanti: il produttore, gli ingredienti utilizzati, le analisi chimiche e perché no... comunicazione tramite la grafica e l'apparenza del prodotto. Il produttore stesso vuole tutelarvi mostrando la sua qualità attraverso la trasparenza delle informazioni.”

L'ambito informatico può avere un concetto molto vicino all'esempio visto, soprattutto in ambito sociale dove i mezzi considerati non sono più una alternativa ma un passaggio obbligato. Conoscere le caratteristiche di un prodotto può essere difficile per i non addetti ai lavori ma è una forma di trasparenza che migliora la scelta e l'evoluzione dei prodotti stessi verso la qualità totale. Focalizzandoci sul software bisogna inevitabilmente fare una distinzione. Esistono diversi sistemi operativi e software che sono vincolati a quest'ultimi. L'analisi immediata mostra che nel mondo esistono sistemi operativi largamente utilizzati per scelta diretta o inconsapevole e nella maggior parte dei casi quest'ultimi appartengono a compagnie private che *non mostrano* gli ingredienti dei loro prodotti. Nella società informatica questo non ha più ragione di esistere e la filosofia Open Sources è la soluzione evolutiva. Tutti devono avere la possibilità di *sapere quello che comprano*. Dal lato

5. Il fenomeno GNU/Linux

degli sviluppatori, poter visualizzare le sorgenti di un sistema operativo può voler dire migliorarne le funzionalità e renderlo maggiormente stabile ed affidabile. Il software “chiuso” e proprietario è sviluppato da un gruppo di persone. Il software Open Sources può vedere coinvolta una intera comunità di persone che apportano migliorie quotidianamente muovendosi verso la qualità totale e offrendo trasparenza assoluta. GNU/Linux rientra in quest’ultimo esempio. Nato e sviluppato in gran parte in rete, ha visto il contributo di una intera comunità di sviluppatori sparsa in tutto il mondo, un progetto collaborativo su larga scala che ha fatto crescere in pochi anni questo sistema operativo *free*, aperto. Molte persone possono affermare che GNU/Linux non sia la migliore soluzione. Io stesso ribadisco che non è attualmente la soluzione per tutti i casi ma ha una caratteristica di successo: è *software vivo costantemente in evoluzione verso la qualità totale*, capace di crescere e modellarsi alle vostre esigenze e alla vostra personalità. GNU/Linux è la soluzione più democratica e trasparente di una società moderna sempre più informatizzata e in rete.

Parlando con un conoscente che sviluppa software in ambito medico e chirurgico è stato messo in evidenza qualcosa di veramente angosciante che lascia riflettere. Il software sviluppato da quest’ultimo funziona su un sistema operativo proprietario e largamente diffuso che spesso si blocca causando errori, perdita di dati ed inevitabile disagio.

Immaginate un intervento chirurgico al cuore che viene interrotto perché il computer che gestisce le informazioni e i dati si è bloccato e vi mostra una finestra blu con gli errori riscontrati (la BSOD, *Blue Screen Of Death*). Giustificereste la mancanza di stabilità di questo sistema operativo proprietario? Giustificereste voi stessi sapendo che avete determinato questa scelta sbagliata?

Il dibattito è aperto.

F.A.Q.

Domanda: “GNU/Linux è free nel senso di gratuito?” Il concetto free di GNU/Linux include anche l’aspetto economico ma esiste un significato altamente superiore. GNU/Linux è liberamente distribuibile in base alle norme espresse dalla licenza GPL e “aperto” ovvero tutti possono vedere il codice sorgente ed apportare modifiche per migliorarlo ed ottimizzarlo alle crescenti esigenze informatiche. Si può capire che il concetto “free” utilizzato per GNU/Linux è altamente differente da quello di “public domain”.

Domanda: “Se richiedo una versione di GNU/Linux, via Internet, da un distributore italiano o internazionale, dovrò pagare tale versione?” Ogni distribuzione offre soluzioni differenti con costi differenti. Le variazioni di prezzo sono dovute spesso al materiale correlato alla distribuzione software di GNU/Linux, manuali cartacei, gadget e supporto tecnico ma esistono soluzioni minime che includono la sola distribuzione software e sono molto economiche.

Domanda: “Ho recentemente notato che molte distribuzioni GNU/Linux diversificano le proprie distribuzioni per workstation e per soluzioni server. Se voglio utilizzare il sistema operativo all’interno di un’azienda devo optare per la seconda soluzione?” La diversificazione delle distribuzioni è più che altro una *scelta di immagine*. Qualsiasi distribuzione può rappresentare una valida soluzione “server” con i dovuti e necessari pacchetti software liberamente disponibili in rete. E’ altresì vero che una soluzione “server” potrebbe non necessitare di pacchetti software superflui e in tal senso la motivazione della scelta di *diversificazione* ha lati unicamente positivi.

Domanda: “Nei negozi di materiale informatico ho trovato con facilità molte distribuzioni GNU/Linux ma non riesco a trovare Debian. Cosa è necessario fare per procurarsela?” Debian GNU/Linux è una distribuzione strettamente vicina alla filosofia del software libero. Non

esiste una struttura commerciale simile a quella delle altre distribuzioni e per questo non è possibile trovare “un pacchetto” commerciale per quest’ultima. Esistono versioni liberamente scaricabili dalla rete e semplici da masterizzare e utilizzare immediatamente. Lo stesso sistema di gestione dei pacchetti software non prevede supporto aggiuntivo a pagamento: è tutto in linea su Internet e moltissimi sono i mirror della distribuzione dove è possibile collegarsi in maniera semplice, trasparente e senza costi aggiuntivi per aggiornamento del sistema.

Il consiglio che do è quello di rivolgersi al LUG (Linux User Group) più vicino e chiedere se è possibile ottenere questa distribuzione oppure scaricare direttamente le immagini ISO disponibili. Per maggiori riferimenti si può visitare il sito <http://www.debian.org>.

5. *Il fenomeno GNU/Linux*

6. Linux User Group

L'esplosione dell'elettronica di massa ha modificato i comportamenti delle persone a livello sociale. Posso ricordare l'impatto che ebbero i primi personal computer come il Commodore VIC20 e lo ZX Spectrum verso la metà degli anni '80 ma indirettamente ho potuto cogliere un clima simile per il mondo HAM Radio. Le persone che si appassionarono a tali tecnologie era facile trovarle in mercatini particolari come le Fiere dell'elettronica, inizialmente ridotte a piccole aggregazioni di persone e ora momenti collettivi di ritrovo e scambio commerciale di oggetti e componenti elettronici. La telematica è stato un fattore di coesione ulteriore tra queste persone ampliando i confini territoriali esistenti. Quando le BBS (Bulletin Board System) erano gli unici strumenti per collegarsi tramite il proprio computer e il modem ad altri sistemi di questo tipo le persone potevano scambiare bit con una certa facilità e allo stesso modo messaggi e ulteriori informazioni, futili o meno. Tutto questo aveva costi e prestazioni imparagonabili con quello che rappresenta ora Internet ma era un passo evolutivo verso una nuova forma di comunicazione. Quando all'inizio degli anni '90, Internet prendeva il sopravvento sulle reti BBS, i vantaggi furono notevoli: i costi telefonici potevano essere ridotti visto che le connessioni erano per lo più verso provider presenti nel proprio distretto comunale piuttosto che interurbane costose. Anche le potenzialità della rete, sotto l'aspetto dei servizi, cambiava radicalmente ampliando i confini metrici delle comunicazioni. In questo clima di *apertura mentale* è cresciuto GNU/Linux e gli utenti che a sua volta utilizzavano quest'ultimo, nel rapido scambio d'opinioni e richieste con gli sviluppatori.

6.1. Linux User Group

Abbiamo precedentemente descritto i gruppi di persone formati con l'avvento dell'elettronica e la telematica. Un Linux User Group è una entità di persone che si avvicina e collabora per ampliare le proprie conoscenze in merito all'utilizzo del sistema operativo GNU/Linux e cerca di creare memoria storica per aiutare ulteriori persone tramite la creazione di documentazione e rapporti interpersonali. Esistono Linux User Group in ogni parte del mondo e l'Italia non è da meno.

Quando *parlare di Linux* poteva suonare quasi eretico in un determinato ambiente informatico o commerciale, si creò un micro-sistema che univa gli ambienti universitari tramite Internet. In quel preciso istante un frammento di storia ritornava attuale. Nacque il Pluto, il primo gruppo di aggregazione italiano, il quale si propose gli obiettivi di diffondere il *software free* o *software libero*. Il polo universitario che diede maggiore spazio alle attività fu l'università di Padova che offrì proprie risorse per sostenere l'iniziativa. Seguirono altri istituti educativi universitari che vedevano in GNU/Linux uno strumento eccellente sotto l'aspetto educativo e pratico.

L'azione del Pluto LUG è stata di fondamentale importanza per l'espansione della comunità italiana degli utenti GNU/Linux, comunità che è cresciuta come numero di partecipanti, comunità che ha saputo creare ulteriori poli di aggregazione come i LUG *locali* con copertura territoriale di carattere regionale, provinciale o limitatamente comunale. Sindacare su questa evoluzione della comunità italiana del software libero è superfluo: ogni trasformazione porta novità positive/negative ma se si realizza è una realtà e come tale merita considerazione.

Un Linux User Group può svolgere diversi compiti per raggiungere gli obiettivi che si è preposto nella diffusione del software libero ma devono essere sottolineati comportamenti non corretti

6. Linux User Group

Nome	URL
Ass.Culturale ErLUG	http://erlug.linux.it
Firenze LUG	http://firenze.linux.it
ILS (Italian Linux Society)	http://www.linux.it
Pluto	http://pluto.linux.it
Roma LUG	http://lugroma.eu.org
Torino LUG	http://torino.linux.it

Tabella 6.1.: Mappa Linux User Group Italiani.

e coerenti verso le persone che il gruppo rappresenta. Un LUG non dovrebbe prestare consulenze tecnico/commerciali a fine di lucro o legarsi a correnti politiche perchè non fanno parte del proprio *ambiente caratteristico*. Un Linux User Group, oltre a predisporre soluzioni tecniche per una comunità virtuale (servizi Internet) può altresì organizzare meeting per i propri utenti, giornate incontro tra appassionati nello stile dei *campus* americani. Dovrebbe ricevere eventuali sponsorizzazioni se quest'ultime possono migliorare l'efficienza e l'efficacia nel raggiungimento dei propri obiettivi.

La mappa italiana dei LUG

Grazie all'opera di Lindo Nepi, viene riportata di seguito una lista dei maggiori e più attivi Linux User Group (tabella 6.1).

La lista completa e aggiornata è disponibile in linea all'indirizzo <http://www.linux.it/LUG/>.

6.2. Meeting nazionali

L'interesse verso il software libero scaturisce collaborazione tra le persone e quest'ultima può sfociare nelle numerose manifestazioni pubbliche per la condivisione di idee ed esperienze, sotto gli aspetti aziendali, scientifici o semplicemente ludici. Nel corso dell'anno 2000 i punti di incontro che hanno richiamato maggiore interesse sono i seguenti:

- > **Linux Meeting - Bologna**
Manifestazione organizzata dall'associazione culturale ErLUG presso la Facoltà di Economia, Università di Bologna. Riferimento web: <http://www.linuxmeeting.net>
- > **LIME 2000**
Manifestazione organizzata dal LUG Roma presso la Facoltà di Ingegneria, Università di Roma. Riferimento web: <http://lugroma.eu.org>
- > **Pluto Meeting**
Manifestazione organizzata dal Pluto LUG a Terni. Riferimento web: <http://meeting.pluto.linux.it>

I riferimenti web, oltre a presentare il materiale realizzato per gli interventi illustrano informazioni e contatti per le future manifestazioni.

Bibliografia

- [1] GNU is not Unix
<http://www.gnu.org>
- [2] Unix Heritage Society
<http://minnie.cs.adfa.edu.au/TUHS/>
- [3] PDP-11 Unix Preservation
<http://minnie.cs.adfa.edu.au/PUPS/>
- [4] *The Evolution of the Unix Time-Sharing System* di Dennis Ritchie
<http://cm.bell-labs.com/cm/cs/who/dmr/hist.html>
- [5] La storia di Multics
<http://www.multicians.org/history.html>
- [6] Oxford University's Virtual Museum of Computing
<http://archive.comlab.ox.ac.uk/other/museums/computing.html>
- [7] Crittografia
<http://ca.alinet.it/crittografia.html>
- [8] Andrea Colombo, Le nuove tecnologie di crittografia
http://impresa-stato.mi.camcom.it/im_43/colo.htm
- [9] InterLex, Introduzione alla firma digitale
<http://www.interlex.com/docdigit/intro/introl.htm>
- [10] The GNU Privacy Handbook, 1999
<http://www.bluemarble.net/~jashley/gnupg/manual/book1.html>
<http://www.bluemarble.net/~jashley/gnupg/manual.ps>
- [11] Introduction to Public-Key Cryptography
<http://developer.netscape.com/docs/manuals/security/pkin/index.htm>
- [12] Kille S., RFC 1779, A String Representation of Distinguished Names, 1995
<http://www.cis.ohio-state.edu/htbin/rfc/rfc1779.html>
- [13] Introduction to SSL
<http://developer.netscape.com/docs/manuals/security/sslin/index.htm>
- [14] OpenSSL
<http://www.openssl.org>
- [15] Das OpenSSL Handbuch, DFN-PCA, 1999
<http://www.pca.dfn.de/dfnpca/certify/ssl/handbuch/>
- [16] R. Housley, W. Ford, W. Polk, D. Solo, RFC 2459: Internet X.509 Public Key Infrastructure – Certificate and CRL Profile 1999
<http://www.cis.ohio-state.edu/htbin/rfc/rfc2459.html>
- [17] Susan G. Kleinmann, Sven Rudolph, Joost Witteveen, The Debian GNU/Linux FAQ, 1999
<ftp://ftp.debian.org/debian/doc/FAQ/>

Indice

I. Storia	9
1. Hackers	11
1.1. Jargon: la nascita di un gergo	11
1.2. Cracker e Phreaker	12
1.3. Linux e l'arte di fare "hacking"	12
1.4. L'importanza della sicurezza informatica	13
1.5. Riflessioni	14
2. Internet e Telematica	15
2.1. Le origini	15
2.2. Le comunità virtuali	17
2.3. Servizi e strumenti di comunicazione	18
2.4. DNS e i Nomi Dominio	19
3. Unix	23
3.1. Da Multics a Unix	23
3.2. La "Networking Release 1"	25
3.3. Free Software Foundation	25
II. Filosofia	27
4. Open Source	29
4.1. Il progetto GNU	29
4.2. Open Sources Definition	44
III. GNU/Linux	59
5. Il fenomeno GNU/Linux	61
5.1. La nascita	61
5.2. Distribuzioni	63
5.3. Documentazione	66
5.4. Valore aggiunto	67
6. Linux User Group	71
6.1. Linux User Group	71
6.2. Meeting nazionali	72

Estensioni di domini nazionali

Codice	Nazione	Codice	Nazione
.ap	Antartide	.ar	Argentina
.at	Austria	.au	Australia
.be	Belgio	.br	Brasile
.ca	Canada	.ch	Svizzera
.cl	Cile	.cr	Costa Rica
.cs	Cecoslovacchia	.cy	Cipro
.de	Germania	.dk	Danimarca
.ec	Ecuador	.ee	Estonia
.es	Spagna	.fi	Finlandia
.fr	Francia	.gl	Groenlandia
.gr	Grecia	.hk	Honk Kong
.hr	Croazia	.hu	Ungheria
.ie	Irlanda	.il	Israele
.in	India	.is	Islanda
.it	Italia	.jp	Giappone
.kr	Corea	.kw	Kuwait
.lu	Lussemburgo	.mx	Messico
.my	Malaisia	.nt	Olanda
.no	Norvegia	.nz	Nuova Zelanda
.pl	Polonia	.pr	Portorico
.pt	Portogallo	.ru	Federazione Russa
.se	Svezia	.sg	Singapore
.sk	Slovacchia	.sl	Slovenia
.th	Tailandia	.tn	Tunisia
.tr	Turchia	.tw	Taiwan
.uk	Regno Unito	.us	Stati Uniti
.ve	Venezuela	.za	Sud Africa

Acronimi

Acronimo	Significato	Traduzione
AFAICT	As far as I can tell	Per quanto possa dirne
AFAIK	As far as I know	Per quanto ne sò
AFK	Away from keyboard	Lontano dalla tastiera
B4	Before	Prima di
BBL	Be back later	Torno più tardi
BTW	By the way	A proposito
BWQ	Buzz word quotient	Quoziente di parolona
CUL	See you later	Ci vediamo più tardi
EOF	End of file	Fine del file
F2F	Face to face	Faccia a faccia
FAQ	Frequently Asked Question	Domande poste di frequente
FOC	Free of charge	Gratis
FUBAR	Fucked up beyond all recognition	Fottuto oltre ogni limite
FYI	For your information	Per tua informazione
<G>	Grin	Ghigno
GA	Go ahead	Vai avanti
GAL	Get a life	Svagati, pensa ad altro
IME	In my experience	Secondo la mia esperienza
IMHO	In my humble opinion	Dal mio umile punto di vista
IMO	In my opinion	Dal mio punto di vista
IOW	In other words	In altre parole
IRL	In real life	Nella realtà
JAM	Just a minute	Un minuto
L8R	Later	A dopo
LOL	Laughs Out Loud	Risate di gusto
MUD	Multi User Dungeon	Gioco di ruolo multi-utente
OAQ	Over and out	Passo e chiudo
OBTW	Oh, by the way...	Ah, a proposito...
OIC	Oh, I see...	Ah, ho capito...
OMG	Oh, my god...	Oh, mio dio...
RFD	Request for discussion	Richiesta di discussione
ROFL	Rolls on floor laughing	Rotolarsi dalle risate
RTFM	Read the fuckin' manual	Leggi il fottuto manuale
TNX	Thanks	Grazie

Smiles

Caratteri	Significato intrinseco
: -)	Sorriso per trasmettere ironia e gioia
;-)	Ammiccamento
: -(Delusione
: -	Indifferente
: ->	Commento sarcastico e pungente
> :-	Arrabbiato
>> :-	Molto arrabbiato
: ->	Barbuto
%+(Picchiato di recente
: ^)	Naso rotto
3 :-	Cornuto
x-)	Strabico
S:-)	Elvis, rockabilly
: -!	Gesto di trattenimento
8-)	Occhialuto
: 8)	Maiale
= :-)	Punk
O:-)	Angelo
: -w	Lingua biforcuta
: -O	Incredulità, a bocca aperta
: -P	Pernacchia, gesto scherzoso
d:-)	Uomo con berretto
ř_ř	Incredulità
^_^	Risata sogghignata
O_O	Alto grado di incredulità
_	Accecato, occhi lucidi
\$_\$_	Opportunista, che pensa solo ai soldi