

RICHARD STALLMAN

SOFTWARE LIBERO  
PENSIERO LIBERO

VOLUME PRIMO

## **Raramuri**

*Sono super-maratoneti, detengono ogni record mondiale di corsa dai 100 km in su, ma vivono nell'anonimato e nella povertà più profondi nella Sierra Madre del Messico del Nord. Sono gli indios Tarahumara, una tribù dimenticata dai bianchi, da essi considerati il diavolo, e dal loro stesso Dio. Si sono dati un nome poetico, Raramuri, «piedi che corrono», perché su queste lunghissime distanze volano come se volessero salire al cielo.*

*Non c'è nessuno che li batta, perché per loro i piedi sono delle ali. Vivono di agricoltura e di una strana caccia, quella ai cervi, non con l'arco e le frecce ma coi piedi, la loro unica arma: sfiancano gli animali correndo loro dietro giorni e giorni, finché la preda non si abbatte esausta. Roba da leggenda...*

Ennio Caretto

“Corriere della Sera” - 31/07/2002

# Introduzione

## Un esperimento globale per l'affermazione della libertà

*Offrire al mondo programmi informatici che possano essere liberamente usati e copiati, modificati e distribuiti, gratis o a pagamento. Questa la scommessa lanciata nell'ormai lontano 1984 da Richard Matthew Stallman. Qualcosa (apparentemente) impossibile perfino a concepirsi, in un'epoca in cui informatica era (ed è) sinonimo di monopoli, produzioni industriali, mega-corporation. Un approccio tanto semplice quanto rivoluzionario, il concetto stesso di software libero, che ci riporta finalmente con i piedi per terra. E la cui pratica quotidiana è ispirata a un principio anch'esso basilare ma troppo spesso dimenticato: la libera condivisione del sapere, qui e ora, la necessità di (ri)prendere in mano la libertà individuale di creare, copiare, modificare e distribuire qualsiasi prodotto dell'ingegno umano. Ponendo così le condizioni per un ribaltamento totale proprio di quell'apparato pantagruelico che ha piegato l'attuale ambito informatico alla mercé di un pugno di colossi, inarrivabili e monopolistici.*

*Nella rapida trasformazione degli equilibri in gioco nell'odierna rivoluzione tecnologica e industriale, il software libero va dunque scardinando certezze antiche, aprendo al contempo le porte a scenari del tutto nuovi e inimmaginabili. Senza affatto escluderne i riflessi nel mondo della piccola e grande imprenditoria e a livello commerciale: basti ricordare l'ampio utilizzo del sistema operativo GNU/Linux (spesso indicato, in maniera imprecisa, solo come 'Linux') sia su macchine high-end come pure su quelle più economiche e dispositivi portatili vari, mentre il 70 per cento dei server web su internet girano su Apache, pro-*

*gramma di software libero. Considerando insomma la centralità assunta dal software in quanto comparto industriale strategico all'interno di una poliedrica età dell'informazione, c'è da scommettere che la rivoluzione innescata da Richard Stallman continuerà a produrre un'onda assai lunga negli anni e nei decenni a venire.*

*Predisposto all'isolamento sociale ed emotivo, fin da ragazzo Stallman dimostra un'acuta intelligenza unita a una sviscerata attrazione per le discipline scientifiche. Laureatosi in fisica ad Harvard nel 1974, alla carriera di accademico frustrato preferisce l'ambiente creativo degli hacker che danno vita al Laboratorio di Intelligenza Artificiale presso il prestigioso MIT (Massachusetts Institute of Technology) di Boston. Si tuffa così nella cultura hacker di quegli anni, imparando i linguaggi di programmazione e lo sviluppo dei sistemi operativi. È qui che, poco più che ventenne, scrive il primo text editor estendibile, Emacs. Ma soprattutto abbraccia lo stile di vita anti-burocratico, creativo e insofferente di ogni autorità costituita, tipico della prima generazione di computer hacker al MIT. Nei primi anni '60 si deve a costoro, ad esempio, la nascita di Spacewar, il primo video game interattivo, che includeva tutte le caratteristiche dell'hacking tradizionale: divertente e casuale, perfetto per la distrazione serale di decine di hacker, dava però concretezza alle capacità di innovazione nell'ambito della programmazione. Ovviamente, era del tutto libero (e gratuito), di modo che il relativo codice venne ampiamente condiviso con altri programmatori. Pur se non sempre queste posizioni di apertura e condivisione erano parimenti apprezzate da hacker e ricercatori "ufficiali", nella rapida evoluzione del settore informatico i due tipi di programmatori finirono per impostare un rapporto basato sulla collaborazione, una sorta di una relazione simbiotica. La generazione successiva, cui apparteneva Richard Stallman, aspirava a calcare le orme di quei primi hacker, particolarmente a livello etico. Onde potersi definire tale, all'hacker era*

*richiesto qualcosa in più che scrivere programmi interessanti; doveva far parte dell'omonima cultura e onorarne le tradizioni in maniera analoga alle corporazioni medievali, pur se con una struttura sociale non così rigida. Scenario che prese corpo in istituzioni accademiche d'avanguardia, quali MIT, Stanford e Carnegie Mellon, emanando al contempo quelle norme non ancora scritte che governavano i comportamenti dell'hacker – l'etica hacker.*

*Proprio per garantire massima consistenza e aderenza a tale etica, dopo non poche vicissitudini, all'inizio del 1984 Stallman lascia il MIT per dedicarsi anima e corpo al lancio del progetto GNU e della successiva Free Software Foundation. Come scrive Sam Williams nella biografia 'ufficiosa' di Stallman (Codice Libero, Apogeo, 2003), il «passaggio di Richard Matthew Stallman da accademico frustrato a leader politico nel corso degli ultimi vent'anni, testimonia della sua natura testarda e della volontà prodigiosa, di una visione ben articolata sui valori di quel movimento per il software libero che ha aiutato a costruire». A ciò va aggiunta l'alta qualità dei programmi da lui realizzati man mano, «programmi che ne hanno cementato la reputazione come sviluppatore leggendario». Un attivismo spietato, il suo, sempre al servizio della libertà di programmazione, di parola, di pensiero. Non certo casualmente alla domanda se, di fronte alla quasi-egemonia del software proprietario, oggi il movimento del software libero rischi di perdere la capacità di stare al passo con i più recenti sviluppi tecnologici, Stallman non ha dubbi: «Credo che la libertà sia più importante del puro avanzamento tecnico. Sceglerei sempre un programma libero meno aggiornato piuttosto che uno non-libero più recente, perché non voglio rinunciare alla libertà personale. La mia regola è, se non posso dividerlo, allora non lo uso».*

*Questo in estrema sintesi il percorso seguito finora dall'ideatore del movimento del software libero, rimandando ulteriori approfondimen-*

*ti alle risorse segnalate in appendice. Ma per quanti hanno scarsa familiarità con simili dinamiche e con lo Stallman-pensiero, oppure per chi vuole esplorare tematiche più ampie, questa collezione di saggi è certamente l'ideale. Primo, perché copre vent'anni di interventi pubblici da parte di colui che viene (giustamente) considerato il "profeta" del software libero. Secondo, perché nella raccolta vengono sottolineati gli aspetti sociali dell'attività di programmazione, chiarendo come tale attività possa creare davvero comunità e giustizia. Terzo, perché nel panorama dell'informazione odierna spesso fin troppo rapida e generica, ancor più in ambito informatico, è vitale tenersi correttamente aggiornati su faccende calde, tipo le crescenti potenzialità del copyleft (noto anche come "permesso d'autore") oppure i pericoli dei brevetti sul software. La raccolta riporta inoltre una serie di documenti storici cruciali: il "Manifesto GNU" datato 1984 (leggermente rivisto per l'occasione), la definizione di software libero, la spiegazione del motivo per cui sia meglio usare la definizione 'software libero' anziché 'open source'. Il tutto mirando ad un pubblico il più vasto possibile: "non occorre avere un background in computer science per comprendere la filosofia e le idee qui esposte", come recita infatti la nota introduttiva del libro originale – Free Software, Free Society: Selected Essays of Richard M. Stallman.*

*L'edizione italiana di quest'ultima è stata scomposta in due distinti volumi: quello che avete per le mani, dove sono raccolte le prime due sezioni della versione inglese, verrà seguito a breve da un secondo con i testi rimanenti. Tra questi, vanno fin d'ora segnalate le trascrizioni di alcuni importanti interventi dal vivo di Stallman (quali "Copyright e globalizzazione nell'epoca delle reti informatiche" e "Software libero: libertà e cooperazione"), oltre al testo integrale delle varie licenze GNU, a partire dalla più affermata, la GPL, General Public License.*

*Si è optato per due volumi italiani onde rendere più agile e godibile l'intera opera originale, considerando lo spessore e la complessità spesso*

*presenti nei vari saggi. Presi nella loro interezza, questi forniranno al lettore un quadro ampio e articolato su questioni pressanti, non soltanto per l'odierno ambito informatico. Proprio perché Stallman non si risparmia affatto, gettando luce sul passato e soprattutto sul futuro di tematiche al crocevia tra etica e legge, business e software, libertà individuale e società trasparente.*

*Senza infine dimenticare come a complemento del tutto sia già attiva un'apposita area sul sito web di Stampa Alternativa (<http://www.stampalternativa.it/freesoft/index.html>) dove circolano interventi vari in tema di software libero e dove troverà spazio l'intera versione italiana del libro. Oltre naturalmente alle relative modifiche, ovvero le segnalazioni di lettori e utenti riguardo errori, contributi, aggiornamenti e quant'altro possibile. Il materiale qui raccolto sarà ulteriormente disponibile sul sito dell'Associazione Software Libero, il quale ospita il gruppo dei traduttori italiani dei testi del progetto GNU (<http://www.softwarelibero.it/gnudoc/>) che ha validamente contribuito alla stesura di questo lavoro. Un lavoro, va detto nel caso qualcuno avesse ancora dei dubbi, portato avanti interamente via internet tra i vari soggetti coinvolti, dalla fase di progettazione a quella di consegna dei materiali definitivi, e ricorrendo al software non proprietario per quanto possibile.*

*Un progetto in evoluzione continua, quindi, in sintonia con la pratica di massima apertura e condivisione su cui vive e prospera il movimento del software libero a livello globale – espressione concreta di un esperimento teso all'affermazione della libertà di tutti e di ciascuno.*

**Bernardo Parrella**

<berny@cybermesa.com>

marzo 2003

*La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.*



**Parte Prima**

**Il progetto GNU  
e il software libero**



# Il progetto GNU

## La prima comunità di condivisione del software

Quando cominciai a lavorare nel laboratorio di Intelligenza Artificiale del MIT [Massachusetts Institute of Technology] 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 una cosa vecchia quanto i computer, proprio come condividere le ricette è antico come l'arte culinaria. Ma noi lo facevamo più di chiunque altro.

Il laboratorio di Intelligenza Artificiale usava un sistema operativo a partizione di tempo (timesharing) chiamato ITS (Incompatible Timesharing System) che il gruppo di hacker del laboratorio aveva progettato e scritto in linguaggio assembler per il Digital PDP-10, uno dei grossi elaboratori di quel periodo. Come membro di questa comunità, hacker di sistema nel gruppo laboratorio, il mio compito era quello di migliorare il sistema.

Non chiamavamo il nostro software “software libero”, poiché questa espressione ancora non esisteva, ma proprio di questo si trattava. Ogni volta che persone di altre università o aziende volevano convertire il nostro programma per adattarlo al proprio sistema e utilizzarlo, gliene davamo volentieri il permesso. Se si notava qualcuno usare un programma sconosciuto e interessante, gli si poteva sempre chiedere di vederne il codice sorgente, in modo da poterlo leggere, modificare, o cannibalizzarne alcune parti per creare un nuovo programma.

L'uso del termine “hacker” per indicare qualcuno che “infrange i sistemi di sicurezza” è una confusione creata dai mezzi di informazione. Noi hacker ci rifiutiamo di riconoscere questo significato, e continuiamo a utilizzare il termine nel senso di “uno che ama programmare, e a cui piace essere bravo a farlo”<sup>1</sup>.

## La comunità si dissolve

La situazione cambiò drasticamente all'inizio degli anni '80, con la dissoluzione della comunità hacker del laboratorio d'Intelligenza Artificiale seguita dalla decisione della Digital di cessare la produzione del computer PDP-10. Nel 1981 la Symbolics, nata da una costola del laboratorio stesso, gli aveva sottratto quasi tutti gli hacker e l'esiguo gruppo rimasto fu incapace di sostenersi (il libro *Hackers* di Steve Levy narra questi eventi, oltre a fornire una fedele ricostruzione della comunità ai suoi albori [in italiano: *Hackers*, Shake Edizioni Underground, 1996]). Quando nel 1982 il laboratorio di Intelligenza Artificiale acquistò un nuovo PDP-10, i sistemisti decisero di utilizzare il sistema timesharing non libero della Digital piuttosto che ITS. Poco tempo dopo la Digital decise di cessare la produzione della serie PDP-10. La sua architettura, elegante e potente negli anni '60, non poteva essere estesa in modo naturale ai maggiori spazi di intervento che andavano materializzando-

<sup>1</sup> È difficile dare una definizione semplice di qualcosa talmente variegato come l'hacking, ma credo che la maggior parte degli “hacks” abbiano in comune la giocosità, la bravura e l'esplorazione. Perciò hacking vuol dire esplorare i limiti di quel che è possibile fare, in uno spirito di scaltra giocosità. Quelle attività che evidenziano queste caratteristiche conquistano il valore di hacking. Si può aiutare a correggere le interpretazioni poco corrette ponendo la semplice distinzione tra intrusioni nei sistemi di sicurezza e hacking – usando il termine “cracking” per tali intrusioni. Coloro che si dedicano a quest'attività vengono definiti “cracker”. Alcuni di loro potrebbero anche essere degli hacker, come altri potrebbero giocare a scacchi o a golf; ma la maggior parte non lo sono. – On Hacking, RMS, 2002.

si negli anni '80. Questo stava a significare che quasi tutti i programmi che formavano ITS divenivano obsoleti. Ciò rappresentò l'ultimo chiodo conficcato nella bara di ITS; 15 anni di lavoro andati in fumo.

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 utilizzando 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 inespresse, che essi danno per scontate; vogliono che siano accettate acriticamente. Esaminiamole, dunque.

Un primo assunto 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.

Un terzo assunto 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 quei programmi. Quest'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.

## **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 d'Intelligenza Artificiale 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.

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 al sapiente ebraico Hillel:

*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. Essendo ateo, non seguivo alcuna guida religiosa, ma a volte mi trovavo ad ammirare qualcosa che qualcuno di loro ha detto.

## “Free” come libero

Il termine “free software” [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:

- l’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 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” [vale solo per l’inglese], 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 [termine 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 [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 TeX 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.

Per questi motivi, 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 d’Intelligenza Artificiale 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, “vrij”). Era un compilatore progettato per trattare più linguaggi, fra cui C e Pascal, e per generare codice binario per diverse architetture. Scrissi 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 compilatore 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

Cominciai 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 né 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 interessava loro 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 tra-

sformato in software proprietario. Il metodo che usammo si chiama copyleft.

Il permesso d'autore [copyleft] 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" 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 sareb-

be 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. Nel 1984 o 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" [copyleft] per battezzare il concetto di distribuzione che allora andavo elaborando.

## **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 (FSF), 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).

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 Shell, 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. [“Bourne Again Shell” è un gioco di parole sul nome “Bourne Shell”, che era la normale shell di Unix; “Bourne again” richiama l'espressione cristiana “born again”, “rinato” (in Cristo)].

## **Il supporto per il software libero**

La filosofia del software libero rigetta in particolare una diffusa pratica commerciale, 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, dovette trovare un'altra fonte di sostentamento. La trovai nella ven-

dita 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" 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 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 doverci preoccupare delle operazioni di I/O.

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 abbia-

mo 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", che permette l'u-

so 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 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<sup>2</sup>, una libreria scritta per

<sup>2</sup> La libreria GNU Readline fornisce una serie di funzioni utilizzabili da applicazioni che consentono all'utente la modifica delle linee di comando man mano che queste vengono composte.

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.

[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].

### **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 un'idea e da un progetto, non da una necessità contingente.

Per 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 un programma tar. Lo stesso vale per i miei programmi: il compilatore GNU, GNU Emacs, GDB, GNU Make.

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<sup>3</sup>. 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 funzio-

<sup>3</sup> L'algoritmo Lempel-Ziv-Welch è usato per la compressione dei dati.

nante, 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 micro-kernel sviluppato alla Carnegie Mellon University e successivamente all'Università dello Utah; GNU Hurd è un gruppo di server (o "herd of gnus": 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.

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.

## 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<sup>4</sup> 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.

<sup>4</sup> XFree86 è un programma che fornisce un ambiente desktop che interfaccia con le periferiche dell'hardware, quali mouse e tastiera; gira su molte piattaforme diverse.

## 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<sup>5</sup>, 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 LesTif, 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 mini-

<sup>5</sup> Motif è un'interfaccia grafica e manager di finestre che gira su X Windows.

mamente 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)<sup>6</sup>.

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.

<sup>6</sup> Nel settembre 2000 Qt venne ri-ridistribuito sotto la GNU GPL, il che ha sostanzialmente risolto questo problema.

## 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 completare ugualmente il lavoro. 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 "a cattedrale"<sup>7</sup>, e dell'affidabilità e della potenza di un dato programma libero, non ci dobbiamo fermare qui; dobbiamo parlare di libertà e di principi.

<sup>7</sup> Probabilmente intendevo scrivere "del modello a bazaar", poiché era questa l'alternativa nuova e inizialmente controversa.

## 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 redistribuzione (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 cosciente 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 coscienti 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 dal 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 GNU/Linux 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.

### **Provaci!**

La filosofia di Yoda (“Non esiste provarci”) 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.

Oggiogiorno 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.

La versione originale inglese di questo saggio è apparsa nel libro *Open Sources: Voices from the Open Source Revolution* (O'Reilly, 1999), e in italiano in *Open Sources: Voci dalla rivoluzione open source* (Apogeo, 1999).

Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Il manifesto GNU

*Il manifesto GNU venne scritto all'inizio del progetto GNU, per stimolarne la partecipazione e il sostegno. Nei primi anni è stato aggiornato in maniera ridotta per documentarne gli sviluppi, ma oggi sembra meglio lasciarlo inalterato per come lo ha visto la maggior parte della gente. Da allora, ci siamo accorti di alcuni equivoci comuni che l'uso di una diversa terminologia potrebbe aiutare a evitare. Nel corso degli anni sono state aggiunte delle note a chiarimento di tali equivoci.*

**Cos'è GNU? Gnu non è Unix!**

GNU, che sta per “Gnu's Not Unix” (Gnu Non è Unix), è il nome del sistema software completo e Unix-compatibile che sto scrivendo per distribuirlo liberamente a chiunque lo possa utilizzare<sup>1</sup>. Molti altri volontari mi stanno aiutando. Abbiamo gran necessità di contributi in tempo, denaro, programmi e macchine.

Fino a ora abbiamo un editor Emacs fornito di Lisp per espanderne i comandi, un debugger simbolico, un generatore di parser com-

<sup>1</sup> *Qui la scelta delle parole è stata poco accurata. L'intenzione era che nessuno dovesse pagare per il permesso di usare il sistema GNU. Ma le parole non lo esprimono chiaramente, e la gente le interpreta spesso come asserzione che GNU debba sempre essere distribuito in forma gratuita o a basso prezzo. Non è mai stato questo l'intento; più oltre il manifesto parla della possibile esistenza di aziende che forniscano il servizio di distribuzione a scopo di lucro. Di conseguenza ho imparato a distinguere tra “free” nel senso di libero e “free” nel senso di gratuito. Il software libero è il software che gli utenti sono liberi di distribuire e modificare. Alcuni lo avranno gratuitamente, altri dovranno pagare per ottenere le loro copie, e se dei finanziamenti aiutano a migliorare il software tanto meglio. La cosa importante è che chiunque ne abbia una copia sia libero di cooperare con altri nell'usarlo.*

patibile con yacc, un linker e circa 35 utility. È quasi pronta una shell (interprete di comandi). Un nuovo compilatore C portabile e ottimizzante ha compilato se stesso e potrebbe essere pubblicato quest'anno. Esiste un inizio di kernel, ma mancano molte delle caratteristiche necessarie per emulare Unix. Una volta terminati il kernel e il compilatore sarà possibile distribuire un sistema GNU utilizzabile per lo sviluppo di programmi. Useremo TeX come formattatore di testi, ma lavoriamo anche su un nroff. Useremo inoltre il sistema a finestre portabile libero X. Dopo di che aggiungeremo un Common Lisp portabile, il gioco Empire, un foglio elettronico e centinaia di altre cose, oltre alla documentazione in linea. Speriamo di fornire, col tempo, tutte le cose utili che normalmente si trovano in un sistema Unix, e anche di più.

GNU sarà in grado di far girare programmi Unix, ma non sarà identico a Unix. Apporteremo tutti i miglioramenti che sarà ragionevole fare basandoci sull'esperienza maturata con altri sistemi operativi. In particolare, abbiamo in programma nomi più lunghi per i file, numeri di versione per i file, un filesystem a prova di crash, forse completamento automatico dei nomi dei file, supporto indipendente dal terminale per la visualizzazione e forse col tempo un sistema a finestre basato sul Lisp, attraverso il quale più programmi Lisp e normali programmi Unix siano in grado di condividere lo schermo. Sia C che Lisp saranno linguaggi per la programmazione di sistema. Per le comunicazioni vedremo di supportare UUCP, Chaosnet del MIT e i protocolli di Internet.

GNU è inizialmente orientato alle macchine della classe 68000/16000 con memoria virtuale, perché sono quelle su cui è più facile farlo girare. Lasciemo agli interessati il lavoro necessario a farlo girare su macchine più piccole.

Vi preghiamo, per evitare confusioni, di pronunciare la 'G' nella

parola ‘GNU’ quando indica il nome di questo progetto [questa avvertenza serve a chiarire che in inglese “GNU” va pronunciato con la g dura, gh-nu, piuttosto che come “new”, niu; identica la pronuncia italiana].

## **Perché devo scrivere GNU**

Credo che il punto fondamentale sia che, se a me piace un programma, io debba dividerlo con altre persone a cui piace. I venditori di software usano il criterio “divide et impera” con gli utenti, facendo sì che non condividano il software con altri. Io mi rifiuto di spezzare così la solidarietà con gli altri utenti. La mia coscienza non mi consente di firmare un accordo per non rivelare informazioni o per una licenza d’uso del software. Ho lavorato per anni presso il Laboratorio di Intelligenza Artificiale per resistere a queste tendenze e ad altri atteggiamenti sgradevoli, ma col tempo queste sono andate troppo oltre: non potevo rimanere in una istituzione dove ciò viene fatto a mio nome contro la mia volontà. Per poter continuare a usare i computer senza disonore, ho deciso di raccogliere un corpus di software libero in modo da andare avanti senza l’uso di alcun software che non sia libero. Mi sono dimesso dal Laboratorio di Intelligenza Artificiale per togliere al MIT ogni scusa legale che mi impedisca di distribuire GNU.

## **Perché GNU sarà compatibile con Unix**

Unix non è il mio sistema ideale, ma non è poi così male. Le caratteristiche essenziali di Unix paiono essere buone e penso di poter colmare le lacune di Unix senza rovinarne le caratteristiche. E adottare un sistema compatibile con Unix può risultare pratico anche per molti altri.

## **Come sarà reso disponibile GNU**

GNU non è di pubblico dominio. A tutti sarà permesso di modificare e ridistribuire GNU, ma a nessun distributore sarà concesso di porre restrizioni sulla sua ridistribuzione. Questo vuol dire che non saranno permesse modifiche proprietarie (18k caratteri). Voglio essere sicuro che tutte le versioni di GNU rimangano libere.

## **Perché molti altri programmatori desiderano essere d'aiuto**

Ho trovato molti altri programmatori molto interessati a GNU che vogliono dare una mano.

Molti programmatori sono scontenti della commercializzazione del software di sistema. Li può aiutare a far soldi, ma li costringe in generale a sentirsi in conflitto con gli altri programmatori, invece che solidali. L'atto di amicizia fondamentale tra programmatori è condividere programmi; le politiche di commercializzazione attualmente in uso essenzialmente proibiscono ai programmatori di trattare gli altri come amici. Gli acquirenti del software devono decidere tra l'amicizia e l'obbedienza alle leggi. Naturalmente molti decidono che l'amicizia è più importante. Ma quelli che credono nella legge non si sentono a proprio agio con queste scelte. Diventano cinici e pensano che programmare sia solo un modo per fare soldi. Lavorando e utilizzando GNU invece che programmi proprietari, possiamo comportarci amichevolmente con tutti e insieme rispettare la legge. Inoltre GNU è un esempio che ispira gli altri e una bandiera che li chiama a raccolta perché si uniscano a noi nel condividere il software. Questo ci può dare una sensazione di armonia che sarebbe irraggiungibile se usassimo software che non sia libero. Per circa la metà dei programmatori che conosco è una soddisfazione importante, che il denaro non può sostituire.

## Come si può contribuire

Chiedo ai produttori di computer donazioni in denaro e macchine, e ai privati donazioni in programmi e lavoro.

Donare delle macchine può far sì che su di esse giri ben presto GNU. Le macchine devono essere sistemi completi e pronti all'uso approvati per l'utilizzo in aree residenziali e non devono richiedere raffreddamento o alimentazione di tipo sofisticato.

Ho conosciuto moltissimi programmatori desiderosi di contribuire a GNU a metà tempo. Per la gran parte dei progetti, un lavoro a metà tempo distribuito risulterebbe troppo difficile da coordinare, perché le varie parti scritte indipendentemente non funzionerebbero insieme. Ma per scrivere un sostituto di Unix questo problema non si pone, perché un sistema Unix completo contiene centinaia di programmi di servizio, ognuno con la propria documentazione separata, e con gran parte delle specifiche di interfaccia date dalla compatibilità con Unix. Se ogni partecipante scrive un solo programma da usare al posto di una utility di Unix, il quale funzioni correttamente al posto dell'originale su un sistema Unix, allora questi programmi funzioneranno bene una volta messi assieme. Anche considerando qualche imprevisto dovuto a Murphy<sup>2</sup>, assemblare tali componenti è un lavoro fattibile. Il kernel invece richiederà una più stretta cooperazione, e verrà sviluppato da un gruppo piccolo e affiatato.

Donazioni in denaro possono mettermi in grado di assumere alcune persone a tempo pieno o a metà tempo. Lo stipendio non sarà alto rispetto agli standard dei programmatori, ma io cerco persone per le quali lo spirito della comunità GNU sia importante quanto il dena-

<sup>2</sup> Questo è un riferimento alla "Legge di Murphy", una legge umoristica secondo la quale, qualora esista la possibilità che qualcosa vada male, allora andrà male.

ro. Io lo vedo come un modo di permettere a degli appassionati di dedicare tutte le loro energie al lavoro su GNU senza essere costretti a guadagnarsi da vivere in un altro modo.

## **Perché tutti gli utenti dei computer ne trarranno beneficio**

Una volta scritto GNU, ognuno potrà avere liberamente del buon software di sistema, così come può avere l'aria<sup>3</sup>.

Questo significa molto di più che far risparmiare a ciascuno il costo di una licenza Unix: vuol dire evitare l'inutile spreco di ripetere ogni volta lo sforzo della programmazione di sistema. Queste energie possono essere invece impiegate ad avanzare lo stato dell'arte.

I sorgenti completi del sistema saranno a disposizione di tutti. Di conseguenza, un utente che abbia necessità di apportare dei cambiamenti al sistema sarà sempre in grado di farlo da solo o di commissionarne le modifiche a un programmatore o a un'impresa. Gli utenti non saranno più in balia di un solo programmatore o di una impresa che, avendo la proprietà esclusiva dei sorgenti, sia la sola a poter fare le modifiche.

Le scuole avranno la possibilità di fornire un ambiente molto più educativo, incoraggiando gli studenti a studiare e migliorare il software di sistema. I laboratori di informatica di Harvard avevano una politica per cui nessun programma poteva essere installato nel sistema senza che i sorgenti fossero pubblicamente consultabili, e la praticarono rifiutandosi effettivamente di installare alcuni programmi. Questo comportamento mi è stato di grande ispirazione. Infine, scompariranno le necessità burocratiche di tener conto di

<sup>3</sup> Questo è un altro punto dove non sono riuscito a distinguere chiaramente tra i due significati di "free". La frase, così com'è, non è falsa – si possono ottenere gratuitamente copie del software GNU, o dagli amici o attraverso la rete. Ma in effetti suggerisce un'idea sbagliata.

chi sia il proprietario del software di sistema e di chi abbia il diritto di farci cosa.

Ogni sistema per imporre tariffe d'uso di un programma, comprese le licenze d'uso per le copie, è sempre estremamente costoso in termini sociali a causa del complesso meccanismo necessario per decidere quanto (cioè per quali programmi) ognuno debba pagare, e solo uno stato di polizia può costringere tutti all'obbedienza. Immaginate una stazione spaziale dove l'aria deve essere prodotta artificialmente a un costo elevato: far pagare ogni litro d'aria consumato può essere giusto, ma indossare la maschera col contatore tutto il giorno e tutta la notte è intollerabile, anche se tutti possono permettersi di pagare la bolletta. E le videocamere poste in ogni dove per controllare che nessuno si tolga mai la maschera sono offensive. Meglio finanziare l'impianto di ossigenazione con una tassa pro capite e buttar via le maschere.

Copiare un programma in tutto o in parte è tanto naturale per un programmatore quanto respirare ed è altrettanto produttivo. Dovrebbe essere altrettanto libero.

### **Alcune obiezioni facilmente confutabili agli obiettivi GNU**

“La gente non lo userà se è gratuito, perché non potrà avere l'assistenza”.

“Un programma deve essere a pagamento, per poter fornire supporto adeguato”.

Se la gente preferisse pagare per GNU più l'assistenza piuttosto che avere GNU gratis senza assistenza, allora un'impresa che fornisce assistenza a chi si è procurato GNU gratis potrebbe operare con profitto.

Si deve distinguere tra il supporto sotto forma di lavoro di pro-

grammazione e la semplice gestione. Il primo non è ottenibile da un venditore di software. Se il problema non è sentito da un numero sufficiente di clienti allora il venditore dirà al cliente di arrangiarsi.

Per chi deve poter contare su questo tipo di supporto l'unica soluzione è di disporre dei sorgenti e degli strumenti necessari, in modo da poter commissionare il lavoro a chi sia disposto a farlo, invece che rimanere in balia di qualcuno. Con Unix il prezzo dei sorgenti rende ciò improponibile per la maggior parte delle imprese. Con GNU questo sarà invece facile. Si darà sempre il caso che non siano disponibili persone competenti, ma questo non potrà essere imputato al sistema di distribuzione. GNU non elimina tutti i problemi del mondo, solo alcuni.

Allo stesso tempo, gli utenti che non sanno nulla di computer hanno bisogno di manutenzione, cioè di cose che potrebbero fare facilmente da soli ma che non sono in grado di fare.

Servizi di questo genere potrebbero essere forniti da aziende che vendono solo gestione e manutenzione. Se è vero che gli utenti sono disposti a pagare per un prodotto con servizio, allora saranno anche disposti a pagare per il servizio avendo avuto il prodotto gratuitamente. Le aziende di servizi si faranno concorrenza sul prezzo e sulla qualità; gli utenti d'altra parte non saranno legati a nessuna di esse in particolare. Nel frattempo, coloro che non avranno bisogno del servizio saranno sempre in grado di usare il programma senza pagare il servizio.

“Non si può raggiungere molta gente senza pubblicità, e per finanziarla si deve far pagare il programma”.

“È inutile reclamizzare un programma gratuito”.

Ci sono molte forme di pubblicità gratuita o a basso costo che pos-

sono essere usate per informare un gran numero di utenti di computer riguardo a cose come GNU. Ma può essere vero che la pubblicità può raggiungere molti più utenti di microcomputer. Se fosse veramente così, una ditta che reclamizzasse il servizio di copia e spedizione per posta di GNU a pagamento dovrebbe aver abbastanza successo commerciale da rientrare dai costi della pubblicità e da guadagnarci. In questo modo, pagano la pubblicità solo gli utenti che ne beneficiano.

D'altro canto, se molta gente ottiene GNU da amici e queste aziende non hanno successo, vorrà dire che la pubblicità non era necessaria per diffondere GNU. Perché tutti questi difensori del libero mercato non vogliono lasciare che sia il libero mercato a decidere?<sup>4</sup>.

“La mia azienda ha bisogno di un sistema operativo proprietario per essere più avanti della concorrenza”.

Con GNU, i sistemi operativi non rientreranno più fra gli elementi di concorrenza. La vostra azienda non potrà essere concorrenziale in quest'area, ma egualmente non potranno esserlo i concorrenti. Vi farete concorrenza in altre aree, mentre in questa godrete di mutui benefici. Se vendete sistemi operativi non apprezzerete GNU, ma è un problema vostro. Se avete un'attività di altro tipo, GNU vi può evitare di essere spinti nel costoso campo della vendita di sistemi operativi.

Mi piacerebbe che lo sviluppo di GNU fosse sostenuto da dona-

<sup>4</sup> La Free Software Foundation raccoglie la maggior parte dei suoi fondi da un servizio di distribuzione, anche se è più un ente senza fini di lucro che un'azienda. Se nessuno sceglie di ottenere copie del software ordinandole alla FSF, questa sarà impossibilitata a proseguire la propria opera. Ma questo non vuole dire che siano giustificate restrizioni proprietarie per costringere gli utenti a pagare. Se una piccola frazione degli utenti ordina le sue copie dalla FSF, questo sarà sufficiente per tenerla a galla. Quindi chiediamo agli utenti di aiutarci in questo modo. Hai fatto la tua parte?

zioni da parte di numerosi produttori e utenti, riducendo così la spesa per tutti<sup>5</sup>.

“Ma i programmatori non meritano una ricompensa per la loro creatività?”.

Se qualcosa merita una ricompensa questo è il contribuire al bene sociale. La creatività può essere un contributo al bene sociale, ma solo nella misura in cui la società è libera di usarne i risultati. Se i programmatori meritano una ricompensa per la creazione di programmi innovativi, allora con la stessa logica meritano una punizione se pongono restrizioni all'uso di questi programmi.

“Un programmatore non dovrebbe poter chiedere una ricompensa per la sua creatività?”.

Non c'è niente di male nel chiedere di esser pagati per il proprio lavoro, o mirare a incrementare le proprie entrate, fintanto che non si utilizzino metodi che siano distruttivi. Ma i metodi comuni nel campo del software, al giorno d'oggi, sono distruttivi.

Spremere denaro dagli utenti di un programma imponendo restrizioni sull'uso è distruttivo perché riduce i modi in cui il programma può essere usato. Questo diminuisce la quantità di ricchezza che l'umanità ricava dal programma. Quando c'è una scelta deliberata di porre restrizioni, le conseguenze dannose sono distruzione deliberata.

La ragione per cui un buon cittadino non usa questi metodi distruttivi per diventare più ricco è che, se lo facessero tutti, diventeremmo tutti più poveri a causa delle distruzioni reciproche. Questa è etica kantiana, la Regola Aurea: poiché non mi piacciono le conse-

<sup>5</sup> Un gruppo di imprese di software ha recentemente costituito dei finanziamenti per sostenere la manutenzione del nostro compilatore C.

guenze che risulterebbero se tutti impedissero l'accesso alle informazioni, devo considerare sbagliato che uno lo faccia. In particolare, il desiderio di una ricompensa per la propria creatività non giustifica il privare il mondo nel suo insieme di tutta o parte di questa creatività.

“Ma i programmatori non moriranno di fame?”.

Potrei rispondere che nessuno è obbligato a fare il programmatore. La maggior parte di noi non è in grado di andare per strada a fare il mimo, ma ciò non vuol dire che siamo condannati a passare la vita per strada a fare i mimi, e morire di fame. Facciamo un altro lavoro.

Ma è la risposta sbagliata, perché accetta l'assunzione implicita di chi pone la domanda, e cioè che senza proprietà del software non è possibile pagare ai programmatori il becco di un quattrino. Un'assunzione del tipo tutto o niente.

La vera ragione per cui i programmatori non moriranno di fame è che sarà per loro egualmente possibile essere pagati per programmare, solo non pagati così tanto come ora.

Porre restrizioni sulle copie non è l'unico modello di affari nel campo del software. È il modello più comune perché è il più redditizio. Se fosse vietato, o rifiutato dagli utenti, l'industria del software si sposterebbe su altri modelli organizzativi, adottandone altri ora meno comuni. Ci sono sempre numerosi modi per organizzare un qualunque tipo di affari.

Probabilmente, programmare nel nuovo modello organizzativo non sarà più così redditizio come lo è ora. Ma questo non è un argomento contro il cambiamento. Che gli addetti alle vendite ricevano i salari che ora ricevono non è considerata un'ingiustizia. Se i programmatori avessero gli stessi stipendi (in pratica guadagnerebbero molto di più), non sarebbe nemmeno quella un'ingiustizia.

“Ma le persone non hanno diritto di controllare come la loro creatività viene usata?”.

Il “controllo sull’uso delle proprie idee” in realtà costituisce un controllo sulle vite degli altri; e di solito viene usato per rendere più difficili le loro vite.

Le persone che hanno studiato con cura i vari aspetti del diritto alla proprietà intellettuale (come gli avvocati) dicono che non c’è alcun diritto intrinseco alla proprietà intellettuale. I tipi dei supposti diritti alla proprietà intellettuale riconosciuti dal governo furono creati da specifici atti legislativi per scopi specifici.

Per esempio, la legislazione sui brevetti fu introdotta per incoraggiare gli inventori a rivelare i dettagli delle loro invenzioni. Lo scopo era avvantaggiare la società, più che avvantaggiare gli inventori. A quel tempo la validità di 17 anni per un brevetto era breve se confrontata con la velocità di avanzamento dello stato dell’arte. Poiché i brevetti riguardano solo i produttori, per i quali il costo e lo sforzo degli accordi di licenza sono piccoli in confronto all’organizzazione della produzione, spesso i brevetti non costituiscono un gran danno. E non ostacolano la gran parte degli individui che usano prodotti coperti da brevetto.

L’idea del copyright non esisteva in tempi antichi, quando gli autori copiavano estesamente altri autori in opere non narrative. Questa pratica era utile, ed è il solo modo attraverso cui almeno parte del lavoro di alcuni autori è sopravvissuto. La legislazione sul copyright fu creata espressamente per incoraggiare l’originalità. Nel campo per cui fu inventata, cioè i libri, che potevano essere copiati a basso costo solo con apparecchiature tipografiche, non fece molto danno e non pose ostacoli alla maggior parte dei lettori.

Tutti i diritti di proprietà intellettuale sono solo licenze concesse dalla società perché si riteneva, correttamente o meno, che conce-

derle avrebbe giovato alla società nel suo complesso. Ma data una situazione particolare dobbiamo chiederci: facciamo realmente bene a concedere queste licenze? Che atti permettiamo di compiere con esse?

Il caso dei programmi ai giorni nostri differisce enormemente da quello dei libri un secolo fa. Il fatto che la via più facile per passare una copia di un programma sia da persona a persona, che il programma abbia un codice sorgente e un codice oggetto che sono cose distinte, e infine il fatto che un programma venga usato più che letto e gustato, combinandosi creano una situazione in cui qualcuno che impone un copyright minaccia la società nel suo insieme, sia materialmente che spiritualmente, una situazione in cui quel qualcuno non dovrebbe farlo, che la legge lo permetta o no.

“La competizione fa sì che le cose siano fatte meglio”.

Il paradigma della competizione è la gara: premiando il vincitore incoraggia ognuno a correre più veloce. Quando veramente il capitalismo funziona in questo modo, fa un buon lavoro; ma chi lo difende ha torto nell'asserire che agisce sempre così. Se i corridori dimenticano il motivo per cui è offerto il premio e si concentrano solo sul vincere non curandosi di come, possono trovare altre strategie, come ad esempio attaccare gli altri concorrenti. Se i corridori si azzuffano, arrivano tutti in ritardo al traguardo.

Il software proprietario e segreto è l'equivalente morale dei corridori che si azzuffano. Triste a dirsi, l'unico arbitro che abbiamo pare non muovere alcuna obiezione alle zuffe, al più le regola (“ogni dieci metri puoi tirare un pugno”). Dovrebbe invece dividerli e penalizzarli anche se solo provassero a combattere.

“Ma senza un incentivo economico non smetterebbero tutti di programmare?”.

In realtà molta gente programmerebbe senza alcun incentivo economico. Programmare ha un fascino irresistibile per alcune persone, solitamente per quelli che ci riescono meglio. Non mancano certo i musicisti professionisti che insistono pur non avendo speranza di guadagnarsi da vivere suonando.

Ma in realtà questa domanda, benché posta spesso, non è appropriata. La paga per i programmatori non sparirà, semplicemente diminuirà. Quindi la domanda corretta è: “qualcuno si metterà mai a programmare per un minore incentivo economico?”.

La mia esperienza dice che sì, ci si metterà.

Per più di dieci anni molti tra i migliori programmatori del mondo hanno lavorato nel Laboratorio di Intelligenza Artificiale per molti meno soldi di quanti ne avrebbero potuti ricevere in ogni altro posto. Hanno avuto soddisfazioni non economiche di moltissimi tipi, ad esempio fama e riconoscenza. E la creatività è anche divertente, un premio di per sé.

Poi molti se ne sono andati quando hanno avuto la possibilità di fare lo stesso interessante lavoro per un mucchio di soldi.

Ciò che i fatti mostrano è che la gente programma per altre ragioni che non siano il denaro; ma se viene data la possibilità di fare la stessa cosa per un mucchio di soldi, allora cominceranno ad aspettarsi e a richiederli. Le organizzazioni che pagano poco sono svantaggiate in confronto a quelle che pagano molto, ma non sarebbero necessariamente in questa posizione se quelle che pagano molto fossero bandite.

“Abbiamo un disperato bisogno dei programmatori. Se ci chiedono di smettere di aiutare i nostri vicini dobbiamo obbedire”.

Non si è mai così disperati da dover obbedire a questo genere di pretese. Ricorda: milioni in difesa, ma non un centesimo in tributi [è una famosa frase di George Washington].

“I programmatori devono guadagnarsi da vivere in qualche modo”.

A breve termine è vero. Ma ci sono un'infinità di modi in cui i programmatori possono guadagnarsi da vivere senza vendere i diritti d'uso dei programmi. Questo metodo è comune ai giorni nostri perché porta la maggior quantità di denaro a programmatori e aziende, non perché sia l'unica strada per guadagnarsi da vivere. È facile trovarne altre, nel caso lo si voglia.

Ecco una serie di esempi:

- Un produttore che immette sul mercato un nuovo computer pagherà per il porting dei sistemi operativi sul nuovo hardware.
- I servizi a pagamento di insegnamento, gestione e manutenzione possono impiegare dei programmatori.
- Persone con idee nuove possono distribuire i programmi gratuitamente chiedendo donazioni agli utenti soddisfatti, o vendendo servizi di gestione. Ho incontrato persone che già lavorano con successo in questo modo.
- Utenti con necessità simili possono formare gruppi e pagare. Un gruppo potrebbe stipulare un contratto con un'impresa di programmazione per scrivere i programmi che i membri del gruppo vorrebbero usare.

Tutti i tipi di sviluppo possono essere finanziati da una Tassa per il Software:

- Supponiamo che chiunque compri un computer debba pagare un  $x$  per cento del costo del computer come tassa per il software. Il governo girerebbe questi fondi a un'agenzia come la NSF [più o meno l'equivalente del nostro CNR] per impiegarli nello sviluppo del software.

- Ma se è lo stesso acquirente a fare una donazione per lo sviluppo del software, potrebbe ottenere un credito nei confronti di queste tasse. Potrebbe fare una donazione a un progetto di sua scelta – tipicamente, scelto perché spera di usarne i risultati quando questo verrà completato. Potrebbe ottenere un credito per ogni donazione fatta, fino al valore totale della tassa che dovrebbe pagare.
- Il gettito complessivo di questa tassa potrebbe essere deciso dal voto di chi la paga, pesato secondo l'ammontare pagato.

Le conseguenze:

- La comunità degli utenti di computer sosterebbe lo sviluppo del software.
- La comunità sceglierebbe il livello di sostegno necessario.
- Gli utenti che fossero interessati a sapere su che progetto vengano spesi i loro soldi avrebbero la possibilità di gestire personalmente la cosa.

Nel lungo periodo, rendere liberi i programmi è un passo verso l'epoca della fine del bisogno, quando nessuno sarà obbligato a lavorare molto duramente solo per guadagnarsi di che vivere. La gente sarà libera di dedicarsi ad attività divertenti, come programmare, dopo aver passato le dieci ore settimanali necessarie in compiti come legiferare, fare consulenza familiare, riparare i robot e prevedere il moto degli asteroidi. Non ci sarà bisogno di guadagnarsi da vivere con la programmazione.

Abbiamo già ridotto moltissimo la quantità di lavoro che la società nel suo complesso deve fare per ottenere la sua produttività attuale, ma poco di questo si è tradotto in benessere per i lavoratori perché è necessario accompagnare l'attività produttiva con molta attività non produttiva. Le cause principali sono la burocrazia e gli sforzi a tutto campo contro la concorrenza. Il software libero ridurrà di

molto questo drenaggio di risorse nell'area della produzione del software. Dobbiamo farlo affinché i guadagni tecnici in produttività si traducano in meno lavoro per noi.

Originariamente scritto nel 1984. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# La definizione di software libero

Sosteniamo questa definizione di software libero per indicare chiaramente ciò che deve essere vero di un particolare programma software perché sia considerato software libero.

Il “software libero” è una questione di libertà, non di prezzo. Per capire il concetto, bisognerebbe pensare alla “libertà di parola” e non alla “birra gratis” [il termine ‘free’ in inglese significa sia ‘gratuito’ che ‘libero’, in italiano il problema non esiste].

L’espressione “software libero” si riferisce alla libertà dell’utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software. Più precisamente, esso si riferisce a quattro tipi di libertà per gli utenti del software:

- Libertà di eseguire il programma, per qualsiasi scopo (libertà 0).
- Libertà di studiare come funziona il programma e adattarlo alle proprie necessità (libertà 1). L’accesso al codice sorgente ne è un prerequisito.
- Libertà di ridistribuire copie in modo da aiutare il prossimo (libertà 2).
- Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio (libertà 3). L’accesso al codice sorgente ne è un prerequisito.

Un programma è software libero se l’utente ha tutte queste libertà. In particolare, se è libero di ridistribuire copie, con o senza modi-

fiche, gratis o addebitando delle spese di distribuzione a chiunque e ovunque. Essere liberi di fare queste cose significa (tra l'altro) che non bisogna chiedere o pagare nessun permesso.

Bisogna anche avere la libertà di fare modifiche e usarle privatamente nel proprio lavoro o divertimento senza doverlo dire a nessuno. Se si pubblicano le proprie modifiche, non si deve essere tenuti a comunicarlo a qualcuno in particolare o in qualche modo particolare.

La libertà di usare un programma significa libertà per qualsiasi tipo di persona od organizzazione di utilizzarlo su qualsiasi tipo di sistema informatico, per qualsiasi tipo di attività e senza dover successivamente comunicare con lo sviluppatore o con qualche altra entità specifica.

La libertà di ridistribuire copie deve includere le forme binarie o eseguibili del programma e anche il codice sorgente, sia per le versioni modificate che non modificate. (La distribuzione di programmi in forma eseguibile è necessaria per consentire un'agevole installazione dei sistemi operativi liberi). È legittimo anche se non c'è alcun modo di produrre una forma binaria o eseguibile, ma si deve avere la libertà di ridistribuire tali forme nel caso si trovi o si sviluppi un modo per farlo.

Affinché le libertà di fare modifiche e di pubblicare versioni migliorate abbiano senso, si deve avere accesso al codice sorgente del programma. Perciò, l'accessibilità al codice sorgente è una condizione necessaria per il software libero.

Queste libertà per essere reali devono essere irrevocabili fin tanto che non si fa qualcosa di sbagliato: se lo sviluppatore del software ha il potere di revocare la licenza anche senza che l'utente sia causa di tale revoca, il software non è libero.

Tuttavia, certi tipi di regole sul come distribuire il software libero

sono accettabili quando non entrano in conflitto con le libertà principali. Per esempio, il permesso d'autore [copyleft] è (detto in due parole) la regola per cui, quando il programma è ridistribuito, non è possibile aggiungere restrizioni per negare ad altre persone le libertà principali. Questa regola non entra in conflitto con le libertà principali, anzi le protegge.

Indipendentemente dal fatto che si siano ottenute copie di software GNU a pagamento o gratuitamente, si ha sempre la libertà di copiare e cambiare il software, e anche di venderne copie.

“Software libero” non vuol dire “non-commerciale”. Un programma libero deve essere disponibile per uso commerciale, sviluppo commerciale e distribuzione commerciale. Lo sviluppo commerciale di software libero non è più inusuale: questo software commerciale libero è molto importante.

Regole su come fare un pacchetto di una versione modificata sono accettabili, a meno che esse in pratica non blocchino la libertà di distribuire versioni modificate. Regole del tipo «se rendi disponibile il programma in questo modo, lo devi rendere disponibile anche in quell'altro modo» possono essere pur esse accettabili, con le stesse condizioni. (Si noti che tale regola lascia ancora aperta la possibilità di distribuire o meno il programma). È anche accettabile che la licenza richieda che, se avete distribuito una versione modificata e un precedente sviluppatore ne richiede una copia, dobbiate inviargliene una.

Nel progetto GNU, noi usiamo il “copyleft” [permesso d'autore] per proteggere queste libertà legalmente per tutti. Ma esiste anche software libero senza copyleft. Crediamo che ci siano importanti ragioni per cui sia meglio usare il permesso d'autore, ma se un programma è software libero senza il permesso d'autore, possiamo comunque utilizzarlo.

Qualche volta le leggi sul controllo delle esportazioni e le sanzioni sul commercio possono limitare la libertà di distribuire copie di programmi verso paesi esteri. I programmatori non hanno il potere di eliminare o di aggirare queste restrizioni, ma quello che possono e devono fare è rifiutare di imporle come condizioni d'uso del programma. In tal modo, le restrizioni non influiranno sulle attività e sulle persone al di fuori della giurisdizione degli stati che applicano tali restrizioni.

Quando si parla di software libero, è meglio evitare di usare espressioni come “gratuito”, perché esse pongono l'attenzione sul prezzo, e non sulla libertà. Parole comuni quali “pirateria” implicano opinioni che speriamo non vogliate sostenere. [Al riguardo, il secondo volume dei saggi conterrà il testo *Termini da evitare*]. Abbiamo inoltre stilato un elenco di traduzioni del termine “free software” in varie lingue (<http://www.gnu.org/philosophy/fs-translations.html>).

Infine, si noti che criteri come quelli indicati in questa definizione di software libero richiedono un'attenta interpretazione. Per decidere se una determinata licenza software si qualifichi come licenza per il software libero, noi la consideriamo basata su questi criteri al fine di determinare se corrisponde al loro spirito così come alle precise parole. Se una licenza include restrizioni irragionevoli, la rifiutiamo, anche se in questi criteri non anticipiamo il problema. Qualche volta le richieste di una licenza sollevano un problema che richiede un'analisi dettagliata, oltre a discussioni con un avvocato prima di poter decidere se la richiesta sia accettabile. Quando raggiungiamo una conclusione riguardo a un nuovo problema, spesso aggiorniamo questi criteri per fare in modo che sia più facile capire perché determinate licenze siano adeguate o meno.

Se siete interessati a sapere se una determinata licenza abbia le caratteristiche per essere una licenza di software libero, consultate il nostro

elenco delle licenze (<http://www.gnu.org/licenses/license-list.html>). Se la licenza che vi interessa non vi è elencata, potete interpellarci inviandoci un'e-mail a <[licensing@gnu.org](mailto:licensing@gnu.org)>.

Originariamente scritto nel 1996. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Perché il software non dovrebbe avere padroni

La tecnologia dell'informazione digitale contribuisce al progresso mondiale rendendo più facile copiare e modificare le informazioni. I computer promettono di rendere questo più facile per tutti noi.

Non tutti vogliono che sia così facile. Il sistema del copyright [diritto d'autore] dà ai programmi software dei "proprietari", molti dei quali mirano a nascondere i potenziali vantaggi del software ad altri. Vorrebbero essere i soli a poter copiare e modificare il software che usiamo.

Il sistema del diritto d'autore è nato e cresciuto con la stampa, una tecnologia per la produzione di massa di copie. Il copyright si adatta bene a questa tecnologia perché pone restrizioni solo ai produttori di massa di copie. Non riduce le libertà dei lettori di libri. Un lettore ordinario, che non possiede una sua tipografia, può copiare i libri solo a mano e pochi lettori sono stati perseguiti per questo. La tecnologia digitale è più flessibile della stampa tipografica: quando l'informazione è in forma digitale, la si può copiare facilmente per condividerla con altri. Questa grande flessibilità si adatta male ad un sistema come quello del diritto d'autore. Questo spiega le misure sempre più sgradevoli e draconiane che vengono oggi usate per far rispettare il diritto d'autore sul software. Consideriamo queste quattro regole della Software Publishers Association (SPA):

- Propaganda massiccia per dire che è sbagliato disobbedire ai proprietari per aiutare gli amici.

- Richieste insistenti di informatori che forniscano informazioni su compagni di lavoro e colleghi.
- IncurSIONI (con l'aiuto della polizia) in scuole e uffici, durante le quali viene detto alle persone che devono provare che non fanno copie illegali.
- Citazione in giudizio (da parte del governo degli Stati Uniti, su richiesta della SPA) di persone come David LaMacchia del MIT, non per aver copiato software (non è stato accusato di averne copiato), ma per avere lasciato senza sorveglianza strumenti per la copia e per non averne censurato l'uso. (Il 27 gennaio 1975 il caso su David LaMacchia è stato archiviato e non è stato ancora presentato appello).

Tutte queste quattro pratiche assomigliano a quelle usate nella ex Unione Sovietica dove ogni fotocopiatrice aveva una guardia per impedire le copie proibite e dove le persone dovevano copiare le informazioni in segreto e passarsele di mano in mano come "samizdat". Naturalmente c'è una differenza: il motivo per il controllo dell'informazione nell'Unione Sovietica era politico; negli Stati Uniti il motivo è il profitto. Quello che ci riguarda sono le azioni, non il loro motivo. Ogni tentativo di bloccare la condivisione delle informazioni, quale ne sia il motivo, porta agli stessi metodi e alla stessa severità.

I proprietari di software usano vari tipi di argomenti per ottenere il potere di controllare in che modo usiamo l'informazione.

### **L'uso dei nomi**

I proprietari di software usano sia parole calunniose come "pirateria" e "furto", sia terminologia tecnica come "proprietà intellettuale" e "danneggiamento", per suggerire al pubblico una certa linea

di pensiero, un'analogia semplicistica fra i programmi e gli oggetti fisici.

Le nostre idee e intuizioni a proposito della proprietà di oggetti materiali riguardano se sia giusto portar via un oggetto a qualcuno. Non si applicano direttamente al fatto di fare una copia di qualcosa. Ma i proprietari ci chiedono di applicarle lo stesso.

### **Esagerazioni**

I proprietari di software dicono che subiscono “danni” o “perdite economiche” quando gli utenti copiano i programmi per conto loro. Ma la copia non ha un effetto diretto sul proprietario e non danneggia nessuno. Il proprietario ha una perdita solo quando chi ha fatto la copia ne avrebbe acquistata una da lui se non l'avesse copiata.

Una piccola riflessione ci mostra che la maggior parte di queste persone non avrebbe comprato la copia. Tuttavia i proprietari calcolano le loro “perdite” come se invece tutti ne avrebbero comprato una. Questa è, a metterla gentilmente, esagerazione.

### **La legge**

I proprietari spesso descrivono la legislazione vigente e le dure sanzioni con cui possono minacciarci. Implicito in questo approccio c'è il suggerimento che la legge attuale riflette un'idea indiscutibile della moralità, e allo stesso tempo siamo invitati a vedere queste sanzioni come fatti di natura per i quali non si può biasimare nessuno. Questa linea argomentativa non è progettata per affrontare un pensiero critico; è intesa a rafforzare il modo di pensare comune.

È ovvio che non è la legge che decide cosa è giusto e cosa è sbagliato. Ogni americano dovrebbe sapere che, quaranta anni fa, era

contro la legge, in molti stati, che una persona di colore si sedesse in un autobus nei posti anteriori; ma solo i razzisti avrebbero detto che fosse sbagliato sedersi lì.

## **Diritti naturali**

Gli autori spesso rivendicano un legame speciale con i programmi che hanno scritto e affermano che, come conseguenza, i loro desideri e i loro interessi rispetto al programma superano quelli di chiunque altro, perfino quelli di tutto il resto del mondo. (In genere sono le aziende, non gli autori, che detengono il copyright sul software, ma ci si aspetta che non si faccia caso a questa differenza).

Per quelli che lo propongono come un assioma etico – l'autore è più importante di voi – posso solo dire che io stesso, noto autore di software, lo considero una fandonia.

Ma in generale è probabile che si provi simpatia solo per la rivendicazione dei diritti naturali, per due ragioni.

Una ragione è la forzata analogia con gli oggetti materiali. Quando mi cucino degli spaghetti reclamerò se a mangiarli è qualcun altro, perché non posso più mangiarmeli io. La sua azione mi danneggia esattamente nello stesso modo in cui favorisce chi li mangia; solo uno di noi può mangiare gli spaghetti, così la domanda è: chi? La più piccola differenza fra di noi è sufficiente a spostare l'ago della bilancia da un punto di vista etico.

Ma se viene eseguito o modificato un programma che ho scritto io, questo riguarda voi direttamente e me solo indirettamente. E se date una copia a un vostro amico, questo riguarda voi e il vostro amico molto di più di quanto riguardi me. Io non dovrei avere il potere di dirvi di non fare queste cose. Nessuno dovrebbe averlo.

La seconda ragione è che è stato detto che i diritti naturali dell'autore sono una tradizione accettata e indiscussa della nostra società.

Ma a guardare la storia, è vero l'opposto. L'idea dei diritti naturali degli autori è stata discussa e fermamente respinta quando venne stesa la Costituzione degli Stati Uniti. Ecco perché la Costituzione permette soltanto un sistema di diritto d'autore e non lo richiede; ecco perché dice che il diritto d'autore deve essere temporaneo. Stabilisce anche che lo scopo del diritto d'autore è di promuovere il progresso, non di premiare l'autore. Il copyright premia infatti in qualche modo l'autore e più ancora l'editore, ma è inteso come un mezzo per modificare il loro comportamento.

La tradizione radicata nella nostra società è che il diritto d'autore riduce i diritti naturali del pubblico, e questo può essere giustificato solo per il bene del pubblico.

## **Economia**

L'ultimo argomento usato per l'esistenza di proprietari del software è che questo porta alla produzione di più software.

Al contrario degli altri, questo argomento almeno usa un approccio legittimo al problema. È basato su un fine valido: soddisfare gli utenti del software. Ed empiricamente è chiaro che le persone producono di più se vengono pagate bene per farlo.

Ma l'argomento economico ha un difetto: è basato sull'assunto che la differenza è solo questione di quanti soldi dobbiamo pagare. Presuppone che la "produzione di software" sia ciò che vogliamo, sia che il software abbia proprietari sia che non li abbia.

Le persone accettano prontamente questo assunto perché si accorda con le nostre esperienze relative agli oggetti materiali. Si consideri un panino, per esempio. Si può avere uno stesso panino sia gratis che a pagamento. In questo caso la sola differenza è la cifra che si paga. Sia che lo si debba pagare o meno, il panino avrà lo stesso sapore, lo stesso valore nutritivo e in entrambi i casi lo si potrà man-

giare solo una volta. Che il panino sia stato acquistato da un proprietario o meno non ha conseguenze dirette su niente eccetto che sulla quantità di denaro che si avrà successivamente.

Ciò vale per qualunque oggetto materiale – il fatto che abbia o meno un proprietario non riguarda direttamente ciò che è o ciò che ci si può fare se lo si acquista.

Ma il fatto che un programma abbia un proprietario ha molte conseguenze su ciò che è e su ciò che si può fare con una copia, quando se ne compra una. La differenza non è solo una questione di denaro. Il sistema di proprietà del software incoraggia i proprietari del software a produrre qualcosa, ma non quello di cui la società ha realmente bisogno. E causa un intangibile inquinamento etico che ha conseguenze su tutti noi.

Di cosa ha bisogno la società? Ha bisogno di una informazione che sia realmente disponibile ai suoi cittadini - per esempio programmi che si possano leggere, correggere, adattare e migliorare, non soltanto usare. Ma quello che viene consegnato di solito dai proprietari del software è una scatola nera che non si può studiare o cambiare.

La società ha anche bisogno di libertà. Quando un programma ha un proprietario, gli utenti perdono la libertà di controllare parte della loro stessa vita.

Ma soprattutto la società ha bisogno di stimolare nei propri cittadini lo spirito di cooperazione volontaria. Quando i proprietari del software ci dicono che aiutare i nostri vicini in maniera naturale è “pirateria”, essi inquinano lo spirito civico della nostra società.

Questo è il motivo per cui diciamo che il software libero è una questione di libertà, non di prezzo.

L'argomento economico a favore dei proprietari di software è sbagliato, ma la questione economica è reale. Alcune persone scrivono software utile per il piacere di scriverlo o per ammirazione e amo-

re; ma se vogliamo più software di quanto già si scriva, bisogna raccogliere fondi.

Da dieci anni gli sviluppatori di software libero provano vari metodi per trovare fondi, con un certo successo. Non c'è bisogno di far diventare tutti ricchi, il reddito medio di una famiglia americana, circa 35.000 dollari annui, ha dimostrato di essere un incentivo sufficiente per molti lavori che sono meno soddisfacenti del programmare. Per anni, fin quando un'associazione lo ha reso non necessario, mi sono guadagnato da vivere con miglioramenti a richiesta del software libero che avevo scritto. Ciascun miglioramento è stato aggiunto alla versione standard rilasciata e reso così disponibile al pubblico. I clienti mi pagavano perché lavorassi sui miglioramenti che volevano loro, piuttosto che sulle funzionalità che altrimenti avrei considerato di più alta priorità.

La Free Software Foundation (FSF), una fondazione senza scopo di lucro per lo sviluppo del software libero, raccoglie fondi con la vendita di CD-ROM, magliette, manuali, e confezioni Deluxe di GNU (che gli utenti sono liberi di copiare e modificare), e anche con donazioni. Attualmente ha un organico di cinque programmatori, più tre impiegati che gestiscono gli ordini postali.

Alcuni sviluppatori di software libero guadagnano offrendo servizi di supporto. Cygnus Support, che ha circa 50 impiegati [quando questo articolo è stato scritto, nel 1994], stima che circa il 15 per cento delle attività del suo personale riguarda lo sviluppo del software libero – una percentuale rispettabile, per una società di software.

(Cygnus Support ha continuato ad avere successo, ma poi ha accettato investimenti esterni, è diventata avida, e ha iniziato a sviluppare software non-libero. Infine è stata acquistata da Red Hat, che ha ri-distribuito gran parte di quei programmi come software libero). Un gruppo di imprese che comprende Intel, Motorola, Texas Instru-

ments e Analog Devices si sono unite per finanziare il continuo sviluppo del compilatore libero GNU per il linguaggio C. Nel frattempo il compilatore GNU per il linguaggio Ada viene finanziato dalla US Air Force, che ritiene questa la modalità di spesa più efficace per ottenere un compilatore di alta qualità. [Il finanziamento della US Air Force è finito un po' di tempo fa; il compilatore GNU Ada è ora in servizio e la sua manutenzione è finanziata commercialmente]. Tutti questi sono piccoli esempi; il movimento del software libero è ancora piccolo e ancora giovane. Ma in questo paese [gli USA] l'esempio di radio sostenute dagli ascoltatori mostra che è possibile sostenere una grande attività senza costringere gli utenti a pagare. Come utenti di computer oggi ci si può trovare ad usare un programma proprietario. Se un amico ti chiede una copia sarebbe sbagliato rifiutare. La cooperazione è più importante del diritto d'autore. Ma una cooperazione nascosta e segreta non contribuisce a rendere giusta la società. Una persona dovrebbe aspirare a vivere una vita onesta, apertamente e con fierezza, e questo comporta dire "No" al software proprietario.

Meritate di poter cooperare apertamente e liberamente con altre persone che usano software. Meritate di poter imparare come funziona il software e con esso di insegnare ai vostri studenti. Meritate di poter assumere il vostro programmatore favorito per aggiustarlo quando non funziona.

Meritate il software libero.

Originariamente scritto nel 1994. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Cosa c'è in un nome?

I nomi trasmettono significati; la scelta che facciamo dei nomi determina il significato di ciò che diciamo. Un nome inappropriato comunica agli interlocutori un'idea sbagliata. Una rosa, qualsiasi nome abbia, avrebbe comunque un buon profumo, ma se la chiamiamo penna, gli interlocutori saranno piuttosto disorientati quando la utilizzeranno per scrivere. E se chiamiamo “rose” le penne, può darsi che gli interlocutori non capiscano a che cosa servano. Se chiamiamo “Linux” il nostro sistema operativo, verrà trasmessa un'idea sbagliata dell'origine, storia e scopo del sistema. Se lo chiamiamo GNU/Linux, verrà trasmessa (anche se non in dettaglio) un'idea accurata.

Questo è forse importante per la nostra comunità? È importante che si conoscano l'origine, la storia e lo scopo del sistema? Sì, perché chi dimentica la storia è spesso condannato a ripeterla. Il *Mondo Libero* che si è sviluppato intorno a GNU/Linux non è sicuro; i problemi che ci hanno portato a sviluppare GNU non sono stati completamente eliminati e minacciano di ripresentarsi.

Quando spiego perché è più appropriato chiamare il sistema operativo “GNU/Linux” invece che “Linux”, a volte ottengo questa risposta:

“Ammetto che il Progetto GNU meriti riconoscimento per questo lavoro, vale davvero la pena di preoccuparsi quando non gli viene dato credito? La cosa importante non è forse che il lavoro sia stato fatto, non chi l'abbia fatto? Dovete rilassarvi, essere orgogliosi del lavoro ben fatto e non preoccuparvi del riconoscimento”.

Questo sarebbe un saggio consiglio, se solo la situazione fosse questa – se il lavoro fosse terminato e fosse il momento di rilassarsi. Se soltanto questo fosse vero! Ma le sfide abbondano e questo non è il momento di ipotecare il futuro. La forza della nostra comunità sta nel dedicarsi alla libertà e alla cooperazione. Utilizzare il nome GNU/Linux è un modo perché le persone si ricordino e informino gli altri di questi obiettivi.

È possibile scrivere del buon software libero senza pensare a GNU; parecchio buon lavoro è stato fatto anche nel nome di Linux. Ma “Linux” è stato associato fin dalla sua creazione a una filosofia che non è vincolata alla libertà di cooperare. E avremo ancora più problemi a farlo associare allo spirito comunitario, dal momento che il nome viene utilizzato sempre di più dal mondo degli affari.

Una grande sfida al futuro del software libero viene dalla tendenza delle società che distribuiscono “Linux” ad aggiungere software non libero a GNU/Linux nel nome della convenienza e del potere. Lo fanno tutti gli sviluppatori delle maggiori distribuzioni commerciali: tra queste, solamente Red Hat offre un prodotto in CD completamente libero, ma non lo si trova in nessun negozio; le altre società non producono nemmeno qualcosa del genere. La maggior parte delle società non permette di identificare chiaramente i pacchetti non liberi delle loro distribuzioni; molte perfino sviluppano software non libero e lo aggiungono al sistema.

La gente giustifica l’inserimento di software non libero in nome della “popolarità di Linux”, dando in effetti maggior valore alla popolarità rispetto alla libertà. Talvolta viene ammesso apertamente. Per esempio la rivista *Wired*, dice Robert McMillan, editore di *Linux Magazine*, “percepisce che lo spostamento verso il software open

source dovrebbe essere alimentato da decisioni tecniche piuttosto che politiche”. E l’amministratore delegato (CEO) di Caldera ha apertamente esortato gli utenti ad abbandonare l’obiettivo della libertà e a lavorare invece per la “popolarità di Linux”.

Inserire software non libero nel sistema GNU/Linux può aumentarne la popolarità, se per popolarità intendiamo il numero di persone che utilizza alcuni GNU/Linux insieme a software non libero. Ma, allo stesso tempo, incoraggia implicitamente la comunità ad accettare software non libero come fatto positivo e a dimenticare l’obiettivo della libertà. Non ha senso guidare più velocemente per poi uscire di strada.

Quando l’“add-on” non libero è una libreria o uno strumento di programmazione, può diventare una trappola per gli sviluppatori di software libero. Quando scrivono del software che dipende dal pacchetto non libero, il loro software non può essere parte di un sistema completamente libero.

(In passato le librerie grafiche Motif e Qt hanno intrappolato in questo modo grandi quantità di software libero, creando problemi la cui soluzione ha richiesto anni. Il problema Qt è risolto perché oggi Qt è libero; il problema Motif non è ancora completamente risolto, dal momento che il suo sostituto libero, LessTif, ha bisogno di qualche rifinitura – offritevi volontari! L’implementazione Java non libera e le librerie Java non standard della Sun vanno ora causando un problema analogo, e la loro sostituzione con software libero è attualmente un importante sforzo di GNU).

Se la nostra comunità continua a muoversi in questa direzione, potrebbe mutare il futuro di GNU/Linux in un mosaico di componenti liberi e non liberi. Fra cinque anni avremo sicuramente ancora moltissimo software libero; ma se non faremo attenzione sarà a malapena utilizzabile senza il software non libero con cui gli

utenti si aspettano di trovarlo. Se succederà, la nostra campagna per la libertà sarà fallita.

Se rilasciare alternative libere fosse semplicemente un problema di programmazione, la soluzione di problemi futuri potrebbe diventare più facile dal momento che aumentano le risorse per lo sviluppo della nostra comunità. Ma dovremo affrontare ostacoli che minacciano di renderla più difficile: le leggi che proibiscono il software libero. Dato che i brevetti sul software sono in aumento e leggi come il DMCA (Digital Millennium Copyright Act) vengono utilizzate per proibire lo sviluppo del software libero per importanti compiti come guardare un DVD o ascoltare una trasmissione RealAudio, per combattere i formati di dati brevettati e segreti non avremo altro modo se non rifiutare i programmi non liberi che li utilizzano.

(Il Digital Millennium Copyright Act del 1998 punta ad aggiornare le leggi statunitensi in tema di copyright; tra le questioni ivi incluse troviamo norme relative alla circonvenzione della protezione di sistemi a tutela del copyright, uso legittimo, responsabilità dei fornitori di servizi online. Per ulteriori dettagli sul DMCA, si veda più avanti il testo *L'interpretazione sbagliata del copyright – una serie di errori*).

Affrontare queste sfide richiederà molti sforzi di diverso genere. Ma ciò di cui abbiamo soprattutto bisogno, per fronteggiare qualsiasi tipo di sfida, è ricordare l'obiettivo della libertà di cooperare. Non possiamo aspettarci che il solo desiderio di avere del software potente e affidabile motivi le persone a impegnarsi molto. Abbiamo bisogno del tipo di determinazione che si ha quando si combatte per la libertà e la comunità, la determinazione a continuare per anni e a non mollare.

Nella nostra comunità, questo obiettivo e questa determinazione

provengono principalmente dal Progetto GNU. Siamo noi quelli che parlano della libertà e della comunità come qualcosa su cui non cedere; le organizzazioni che parlano di “Linux” normalmente non lo dicono. Le riviste su “Linux” sono normalmente piene di annunci che pubblicizzano il software non libero; le società che mettono insieme dei pacchetti “Linux” aggiungono al sistema software non libero; altre società “supportano Linux” con applicazioni non libere; gli user group di “Linux” invitano normalmente i fornitori a presentare queste applicazioni. È probabile che anche persone di rilievo della nostra comunità si imbattano nell’idea di libertà e nella determinazione che c’è nel Progetto GNU.

Ma quando le persone vi si imbattono, sentono che riguarda anche loro?

Chi sa che sta utilizzando un sistema proveniente dal Progetto GNU riesce a vedere una relazione diretta tra se stesso e GNU. Non sarà automaticamente d’accordo con la nostra filosofia, ma vedrà almeno un motivo per pensarci seriamente. Al contrario, chi si considera un “utente Linux” e crede che il Progetto GNU “abbia sviluppato strumenti che si sono rivelati utili per Linux”, percepisce normalmente soltanto una relazione indiretta tra sé e GNU. Questo tipo di persona potrebbe semplicemente ignorare la filosofia GNU quando vi si imbatte.

Il Progetto GNU è idealistico e chiunque incoraggi l’idealismo oggi deve affrontare un grande ostacolo: l’ideologia prevalente incoraggia a rifiutare l’idealismo in quanto “irrealizzabile”. Il nostro idealismo è stato estremamente pratico: è il motivo per cui abbiamo un sistema operativo GNU/Linux libero. Chi ama questo sistema deve sapere che è il nostro idealismo divenuto reale.

Se “il lavoro” fosse davvero già terminato, se non ci fosse niente in gioco oltre al riconoscimento, forse sarebbe più saggio lasciar cade-

re la questione. Ma non siamo in questa posizione. Per stimolare le persone a fare il lavoro che deve essere fatto, abbiamo bisogno che ci venga riconosciuto il lavoro fatto finora. Per favore aiutateci, chiamando GNU/Linux il sistema operativo.

Originariamente scritto nel 2000. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Perché “software libero” è da preferire a “open source”

Mentre il software libero chiamato in qualunque altro modo offrirebbe le stesse libertà, fa una grande differenza quale nome utilizziamo: parole differenti hanno significati differenti.

Nel 1998, alcuni sviluppatori di software libero hanno iniziato a usare l'espressione “software open source” (<http://www.opensource.org>) invece di “software libero” per descrivere quello che fanno. Il termine “open source” è stato rapidamente associato a un approccio diverso, una filosofia diversa, valori diversi e perfino un criterio diverso in base al quale le licenze diventano accettabili. Oggi il movimento del Software Libero e il movimento dell'Open Source sono due movimenti diversi con diversi punti di vista e obiettivi, anche se possiamo lavorare, e in effetti lavoriamo, insieme su alcuni progetti concreti.

La differenza fondamentale tra i due movimenti sta nei loro valori, nel loro modo di guardare il mondo. Per il movimento Open Source, il fatto che il software debba essere Open Source o meno è un problema pratico, non un problema etico. Come si è espresso qualcuno, “l'Open Source è una metodologia di sviluppo; il Software Libero è un movimento di carattere sociale”. Per il movimento Open Source, il software non libero è una soluzione non ottimale. Per il movimento del Software Libero, il software non libero è un problema sociale e il software libero è la soluzione.

## La relazione tra il movimento del Software Libero e il movimento Open Source

Il movimento del Software Libero e quello Open Source sono come due partiti politici all'interno della comunità del Software Libero. Negli anni '60 i gruppi radicali si sono fatti la reputazione di essere faziosi: le organizzazioni si dividevano per disaccordi sui dettagli della strategia da utilizzare e poi si odiavano reciprocamente. O per lo meno questa è l'immagine che si ha di essi, vera o falsa che sia.

La relazione tra il movimento del Software Libero e quello Open Source è semplicemente l'opposto di questa situazione. Siamo in disaccordo sui principi di base, ma siamo più o meno d'accordo sugli aspetti pratici. Perciò possiamo lavorare e in effetti lavoriamo assieme su molti progetti specifici. Non vediamo il movimento Open Source come un nemico. Il nemico è il software proprietario.

Noi non siamo contro il movimento Open Source, ma non vogliamo essere confusi con loro. Riconosciamo che hanno contribuito alla nostra comunità, ma noi abbiamo creato questa comunità e vogliamo che si sappia. Vogliamo che quello che abbiamo realizzato sia associato con i nostri valori e la nostra filosofia, non con i loro. Vogliamo che ci sentano, non vogliamo sparire dietro a un gruppo con punti di vista diversi. Per evitare che si pensi che facciamo parte del movimento Open Source, ci preoccupiamo di evitare di utilizzare il termine "open" per descrivere il software libero, o il suo contrario, "closed", per parlare di software non libero.

Quindi, per favore, menzionate il movimento del Software Libero quando parlate del lavoro che abbiamo fatto e del software che abbiamo sviluppato – come il sistema operativo GNU/Linux.

## I due termini a confronto

Il resto di questo articolo confronta i due termini “software libero” e “open source”. Spiega perché il termine “open source” non risolve i problemi, anzi di fatto ne crea alcuni.

### Ambiguità

L'espressione “software libero” ha un problema di ambiguità [il termine “free” in inglese può significare sia “libero” sia “gratis”, in italiano non succede]: un significato non previsto, “software che si può avere senza spendere niente” corrisponde a quell'espressione altrettanto bene del significato previsto, cioè software che dà all'utente certe libertà. Abbiamo risolto questo problema pubblicando una definizione più precisa di software libero, ma questa non è la soluzione perfetta. Non può eliminare completamente il problema. Sarebbe meglio un termine corretto e non ambiguo, presupponendo che non ci siano altri problemi.

Sfortunatamente, tutte le alternative in inglese presentano problemi. Abbiamo considerato molte alternative che ci sono state suggerite, ma nessuna è così completamente “corretta” che sia una buona idea sceglierla. Tutte le soluzioni proposte per “software libero” hanno un qualche tipo simile di problema semantico, se non peggio, incluso “software open source”.

La definizione ufficiale di “software open source”, come pubblicata dalla Open Source Initiative, si avvicina molto alla nostra definizione di software libero; tuttavia, per certi aspetti è un po' più ampia, ed essi hanno accettato alcune licenze che noi consideriamo inaccettabilmente restrittive per gli utenti. Tuttavia, il significato ovvio di “software open source” è «puoi guardare il codice sorgente». Questa è una espressione meno vigorosa di “software libero”;

include il software libero, ma include anche software semi-libero come ad esempio Xv e perfino qualche software proprietario, inclusa Qt nella sua licenza originale (prima della QPL).

Questo significato ovvio di “open source” non è quello inteso dai suoi sostenitori. Il risultato è che la maggior parte delle persone fraintende quello che quei sostenitori sostengono. Ecco come lo scrittore Neal Stephenson ha definito “open source”:

Linux è software “open source” e questo significa, semplicemente, che chiunque può ottenere le copie del suo codice sorgente.

Non penso che abbia cercato deliberatamente di rifiutare o contrastare la definizione “ufficiale”. Penso che abbia semplicemente applicato le convenzioni della lingua inglese per arrivare al significato. Lo stato del Kansas ha pubblicato una definizione simile: «Utilizzare software open source (SOS). SOS è software il cui codice sorgente è disponibile liberamente e pubblicamente, anche se gli specifici accordi di licenza variano relativamente a quanto sia permesso fare con quel codice».

Ovviamente, chi si occupa di open source ha cercato di affrontare questo problema pubblicando una definizione precisa del termine, proprio come abbiamo fatto noi per “software libero”.

Ma la spiegazione di “software libero” è semplice: chi ha capito il concetto di “libertà di parola, non birra gratis” non sbaglierà più [in inglese, l’espressione “free speech, not free beer” mette sinteticamente in contrasto i due significati della parola “free”]. Non c’è un modo più breve per spiegare il significato di “open source” e indicare chiaramente perché la definizione ovvia è quella sbagliata.

## **La paura della libertà**

Il principale argomento a favore dell’espressione “software open source” è che “software libero” può far sentire a disagio. Ed è vero:

parlare di libertà, di problemi etici, di responsabilità, così come di convenienza, è chiedere di pensare a cose che potrebbero essere ignorate. Questo può causare imbarazzo e alcune persone possono rifiutare l'idea di farlo. Questo non vuol dire che la società starebbe meglio se smettessimo di parlare di questi argomenti.

Anni fa, gli sviluppatori di software libero si accorsero di queste reazioni di disagio e iniziarono a cercare una soluzione a questo problema. Pensarono che mettendo in secondo piano l'etica e la libertà e parlando piuttosto dei benefici pratici immediati di qualche software libero, sarebbero stati in grado di "vendere" il software più efficacemente a una determinata utenza, in particolar modo alle aziende. Il termine "open source" viene offerto come un modo per venderne di più – un modo per essere "più accettabili alle aziende". Il punto di vista e i valori del movimento Open Source derivano da questa decisione.

Questo approccio al problema ha dimostrato di funzionare, alle sue condizioni. Oggi molte persone passano al software libero per ragioni puramente pratiche. Questa è una buona cosa, di per sé, ma non è tutto quello che dobbiamo fare! Non basta attirare gli utenti verso il software libero: questo è solo il primo passo.

Prima o poi questi utenti saranno invitati a utilizzare nuovamente software proprietario per alcuni vantaggi pratici. Un enorme numero di aziende cerca di offrire questa tentazione, e perché gli utenti dovrebbero rifiutare? Solo se hanno imparato a valorizzare la libertà che viene offerta loro dal software libero di per sé. Tocca a noi diffondere quest'idea – e per farlo, dobbiamo parlare di libertà. Una parte dell'approccio "teniamole tranquille" nei confronti delle aziende può essere utile per la comunità, ma dobbiamo comunque parlare molto di libertà.

Attualmente, è molto diffuso l'approccio "teniamole tranquille",

ma non si parla abbastanza della libertà. La maggior parte delle persone coinvolte nel software libero parla molto poco della libertà – di solito perché cerca di essere “più accettabile per le aziende”. I distributori di software sono quelli che più seguono questa regola. Alcune distribuzioni del sistema operativo GNU/Linux aggiungono pacchetti di software proprietario al sistema libero di base e invitano gli utenti a considerarlo un vantaggio, invece che un passo indietro rispetto alla libertà.

Non riusciamo a rimanere alla pari rispetto all’afflusso di utenti di software libero, non riusciamo a insegnare alle persone cosa siano queste libertà e cosa sia la nostra comunità man mano che vi entra. Questo è il motivo per cui software non libero (come lo era Qt la prima volta che divenne popolare) e le distribuzioni di sistemi operativi parzialmente non liberi, trovano un terreno così fertile. Smettere di utilizzare la parola “libero” adesso sarebbe un errore. Abbiamo bisogno che si parli di più, e non di meno, di libertà. Che coloro che usano il termine “open source” portino più utenti alla nostra comunità è senz’altro un contributo, ma significa che dobbiamo impegnarci ancora di più per portare il problema della libertà all’attenzione di quegli utenti. Dobbiamo dire “è software libero e ti dà libertà!” sempre di più e più forte che mai.

### **Un marchio registrato può aiutare?**

I sostenitori del “software open source” hanno tentato di rendere questo un marchio registrato, pensando di poter così prevenire utilizzi scorretti. Il tentativo è fallito quando, nel 1999, la richiesta è stata fatta decadere. Per cui lo status legale di “open source” è lo stesso di quello del “software libero”: non esiste nessuna restrizione legale per il suo utilizzo. Ho sentito, talvolta di persona, molte aziende chiamare “open source” i loro pacchetti software anche se

questi non rientravano, per le loro caratteristiche, nella definizione ufficiale.

Ma avrebbe davvero fatto questa grande differenza usare un termine che fosse un marchio registrato? Non necessariamente.

Le aziende inoltre hanno fatto annunci che danno l'impressione che un programma sia "software open source" senza dirlo esplicitamente. Ad esempio, un annuncio di IBM riguardo a un programma che non rientrava nella definizione ufficiale diceva questo:

«Come è comune fare nella comunità open source, gli utenti della ... tecnologia saranno inoltre in grado di collaborare con IBM ...»

Questa frase non dice che il programma è "open source", ma molti lettori non hanno notato quel dettaglio. (Devo comunque far notare che IBM era sinceramente interessata a rendere questo programma software libero e ha successivamente adottato una nuova licenza che lo rendeva tale e "open source". Ma quando questo annuncio è stato fatto, il programma non si qualificava come nessuno dei due).

Ed ecco come Cygnus Solutions, che fu creata come azienda di software libero e successivamente estese la sua attività (per così dire) al software proprietario, pubblicizzava alcuni prodotti software proprietari:

«Cygnus Solution è una azienda leader nel mercato open source e ha appena lanciato due prodotti sul mercato [GNU/]Linux».

Diversamente da IBM, Cygnus non stava tentando di rendere questi pacchetti software libero e questi pacchetti non si avvicinavano minimamente a poter essere definiti tali. Ma Cygnus non ha in realtà detto che questo è "software open source", ha soltanto utilizzato questo termine per dare quest'impressione a un lettore poco attento.

Queste osservazioni suggeriscono che un marchio registrato non avrebbe risolto sul serio i problemi legati al termine "open source".

## Le errate interpretazioni di “open source”

La definizione di open source è abbastanza chiara ed è abbastanza chiaro che il tipico programma non libero non rientra in questa definizione. Quindi penserete che una “azienda Open Source” produca software libero (o qualcosa del genere), giusto? Non sempre è vero, molte aziende stanno anche cercando di dargli un differente significato.

All’incontro “Open Source Developers Day” svoltosi nell’agosto 1998, molti degli sviluppatori commerciali invitati dissero che erano intenzionati a creare come software libero (o “open source”) solo una parte del loro lavoro. Il fulcro del loro business è lo sviluppo di aggiunte proprietarie (software o documentazione) da vendere agli utenti di questo software libero. Ci chiedono di considerarlo come legittimo, come parte della nostra comunità, poiché parte del denaro viene donato per lo sviluppo di software libero.

In effetti, queste aziende tentano di guadagnare una favorevole immagine “open source” per i loro prodotti software proprietari – anche se questi non sono software “open source” – poiché hanno una qualche relazione con il software libero o perché la stessa azienda mantiene anche un qualche software libero. (Il fondatore di una azienda ha esplicitamente detto che avrebbero messo, nei pacchetti di software libero da loro supportati, un po’ del loro lavoro per poter far parte della comunità).

Negli anni, molte aziende hanno contribuito allo sviluppo del software libero. Alcune di queste aziende sviluppavano principalmente software non libero, ma le due attività erano separate. Per questo potevamo ignorare i loro prodotti non liberi e lavorare con loro sui progetti di software libero. Quindi potevamo poi onestamente ringraziarli per i loro contributi al software libero, senza parlare degli altri prodotti che portavano avanti.

Non possiamo fare altrettanto con queste nuove aziende, poiché loro non lo accetterebbero. Queste aziende cercano attivamente di portare il pubblico a considerare senza distinzione tutte le loro attività. Vogliono che noi consideriamo il loro software non libero come se fosse un vero contributo, anche se non lo è. Si presentano come “aziende open source” sperando che la cosa ci interessi, che le renda attraenti ai nostri occhi e che ci porti ad accettarle.

Questa pratica di manipolazione non sarebbe meno pericolosa se fatta utilizzando il termine “software libero”. Ma le aziende non sembrano utilizzare il termine “software libero” in questo modo. Probabilmente la sua associazione con l’idealismo lo rende non adatto allo scopo. Il termine “open source” ha così aperto tutte le porte.

In una mostra specializzata di fine 1998, dedicata al sistema operativo spesso chiamato “Linux”, il relatore di turno era un alto dirigente di una importante azienda di software. Era stato probabilmente invitato poiché la sua azienda aveva deciso di “supportare” questo sistema. Sfortunatamente, la forma di “supporto” consisteva nel rilasciare software non libero che funziona con il sistema – in altre parole, utilizzava la nostra comunità come un mercato ma non vi contribuiva affatto.

Disse: “Non renderemo mai il nostro prodotto open source, ma forse lo renderemo tale ‘internamente’. Se permetteremo al nostro staff di supporto ai clienti di avere accesso al codice sorgente, potrà risolvere gli errori per i clienti e potremo quindi fornire un prodotto e un servizio migliori”. (Questa non è la trascrizione esatta del discorso, poiché non avevo preso nota delle parole, ma rende comunque l’idea). Alcune persone tra il pubblico mi dissero successivamente “non ha capito il senso del nostro lavoro”. Era forse vero? Quale senso non aveva colto?

In realtà aveva colto il significato del movimento Open Source. Questo movimento non dice che gli utenti dovrebbero avere libertà, dice solo che permettendo a più persone di guardare il codice sorgente e di aiutare a migliorarlo, consentirà uno sviluppo più veloce e migliore. Il dirigente ha colto perfettamente quel significato: non ha voluto utilizzare questo approccio nella sua interezza, utenti inclusi, pensando di utilizzarlo parzialmente all'interno della sua azienda.

Il significato che non ha colto è quello che l'“open source” ha progettato di non sollevare: cioè che l'utente merita la libertà.

Diffondere l'idea della libertà è un lavoro difficile: ha bisogno del vostro aiuto. Per questo il progetto GNU rimarrà legato al significato di “software libero”, per aiutare a diffondere l'idea di libertà. Se sentite che libertà e comunità sono importanti in quanto tali – non soltanto per la convenienza implicita in esse – unitevi a noi nell'utilizzare il termine “software libero”.

Joe Barr ha scritto un articolo intitolato “Live and let license” dove illustra il proprio punto di vista su questo argomento: <http://www.itworld.com/App-Dev/350/LWD010523vcontrol4/>

Originariamente scritto nel 1998. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Rilasciare software libero se lavorate all'università

All'interno del movimento del software libero, crediamo che gli utenti informatici debbano godere della libertà di modificare e distribuire il software che usano. Il termine "free", riferito al software libero, indica la libertà: in altre parole, gli utenti hanno la libertà di eseguire, modificare e ridistribuire il software. Il software libero contribuisce alla conoscenza umana, al contrario di quanto fa il software non libero. Le università dovrebbero perciò incoraggiare il software libero per l'avanzamento della conoscenza umana, così come dovrebbero incoraggiare ricercatori e studenti a pubblicare i propri lavori.

Ahimè, molti amministratori universitari dimostrano una tendenza caratterizzata dall'avidità verso il software (e verso la scienza); vedono nei programmi l'opportunità per trarne dei profitti, non per contribuire alla conoscenza umana. Gli sviluppatori di software libero hanno dovuto far fronte a questa tendenza per almeno vent'anni. Quando iniziai a sviluppare il sistema operativo GNU, il primo passo fu quello di lasciare il mio posto al MIT. Lo feci proprio per impedire all'ufficio licenze del MIT di interferire con il rilascio di GNU come software libero. Avevo pianificato un approccio preciso per licenziare programmi GNU in modo che fosse assicurato il mantenimento delle versioni modificate come software libero, un approccio concretizzatosi nella GNU General Public License (GNU GPL), e non volevo supplicare l'amministrazione del MIT perché me lo lasciasse fare.

Nel corso degli anni, spesso esponenti universitari hanno contattato la Free Software Foundation per chiedere consiglio su come convincere gli amministratori che considerano il software soltanto come qualcosa da vendere. Un buon metodo, applicabile anche a progetti finanziati ad hoc, è basare il vostro lavoro su un programma già esistente rilasciato sotto la licenza GNU GPL. A quel punto potete dire agli amministratori: “Non possiamo rilasciare la versione modificata con una licenza che non sia la GNU GPL, qualsiasi altro modo violerebbe il diritto d’autore”. Quando l’immagine del dollaro sfumerà davanti ai loro occhi, generalmente acconsentiranno a rilasciarlo come software libero.

Potete anche chiedere aiuto allo sponsor che finanzia. Quando un gruppo della NYU [New York University] sviluppò il compilatore GNU Ada con i fondi della US Air Force, il contratto prevedeva esplicitamente la donazione del codice risultante alla Free Software Foundation. Contrattate prima lo sponsor, poi chiarite gentilmente all’amministrazione dell’università che non è possibile rinegoziare l’accordo preso. Preferiranno avere un contratto per sviluppare software libero piuttosto che non averne affatto, così molto probabilmente acconsentiranno.

Per tutto ciò che fate, sollevate presto la questione – sicuramente prima che il programma sia stato sviluppato per metà. A questo punto, l’università avrà ancora bisogno di voi e potrete giocare le vostre carte: dite all’amministrazione che finirete il programma, lo renderete utilizzabile, se accetterà per iscritto che sia software libero (e accoglierà la vostra scelta di licenziarlo come software libero). In caso contrario, ci lavorerete sopra quel tanto che basta per scriverne una ricerca, e senza mai creare una versione sufficientemente evoluta da poter essere distribuita. Quando gli amministratori si renderanno conto che la scelta è tra avere pacchetti di software libe-

ro che porteranno credito all'università o non avere proprio niente, generalmente sceglieranno la prima opzione.

Non tutte le università seguono politiche basate sull'avidità. La politica comunemente seguita alla University of Texas prevede il rilascio come software libero sotto GNU General Public License di tutto il software sviluppato al suo interno. La Unives in Brasile e l'International Institute of Information Technology di Hyderabad (India) seguono entrambe una politica favorevole al rilascio di software sotto GPL. Sviluppando prima il supporto per la facoltà, potrete riuscire a instaurare una politica analoga nella vostra università. Presentatela come una questione di principio: l'università ha la missione di stimolare l'avanzamento della conoscenza umana, o il suo unico scopo è quello di perpetuare se stessa?

Qualunque approccio usiate, aiuta mostrarsi determinati e adottare una prospettiva etica, come facciamo nel movimento del software libero. Per trattare il pubblico in modo eticamente corretto, il software dovrebbe essere libero – nel senso della libertà – per chiunque.

Molti sviluppatori di software libero professano ragioni strettamente pratiche per farlo: sostengono di voler consentire ad altri di condividere e modificare il software come espediente per renderlo potente e affidabile. Se questi valori vi spingono a sviluppare software libero, funzionante e utile, vi ringraziamo per il contributo. Ma tali valori non vi offrono una forte presa per resistere quando gli amministratori universitari tentano di convincervi a scrivere software non-libero.

Possono, ad esempio, sostenere che: “Potremmo renderlo ancora più potente e affidabile con tutto il denaro che potremmo farci”. Questa pretesa può o meno rivelarsi valida alla fine, ma è dura da confutare a priori. Possono suggerire una licenza che offra copie

“gratuite, esclusivamente a uso accademico”, sottintendendo così che il pubblico generico non meriti la libertà e che ciò solleciterà la cooperazione dei ricercatori, che è tutto quello di cui (dicono) avete bisogno.

Se partite da valori “pragmatici”, è difficile trovare una buona ragione per rifiutare queste proposte senza via d’uscita, ma potete riuscirci facilmente se basate la vostra fermezza su valori etici e politici. Cosa c’è di positivo nel creare un programma potente e affidabile a spese della libertà degli utenti? Non si dovrebbe applicare la libertà sia all’interno che all’esterno delle istituzioni accademiche? Le risposte sono ovvie se la libertà e la comunità rientrano tra i vostri obiettivi. Il software libero rispetta la libertà degli utenti, mentre il software non libero la nega.

Non c’è nulla che rafforzi la vostra risolutezza come sapere che la libertà della comunità dipende, in primo luogo, da voi stessi.

Originariamente scritto nel 2002. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Vendere software libero

Molta gente crede che lo spirito del progetto GNU sia che non si debba far pagare per distribuire copie del software, o che si debba far pagare il meno possibile – solo il minimo per coprire le spese. In realtà noi incoraggiamo chi ridistribuisce il software libero a far pagare quanto vuole o può. Se vi sembra sorprendente, per favore continuate a leggere.

Il termine “free” ha due legittimi significati comuni: può riferirsi sia alla libertà che al prezzo. Quando parliamo di “free software”, parliamo di libertà, non di prezzo. Ci si rammenti di considerare “free” come in “free speech” (libertà di parola) anziché in “free beer” (birra gratis). In particolare, significa che l’utente è libero di eseguire il programma, modificarlo, e ridistribuirlo con o senza modifiche.

I programmi liberi sono talvolta distribuiti gratuitamente, e talvolta a un prezzo consistente. Spesso lo stesso programma è disponibile in entrambe le modalità in posti diversi. Il programma è libero indipendentemente dal prezzo, perché gli utenti sono liberi di utilizzarlo.

Programmi non-liberi vengono di solito venduti a un alto prezzo, ma talvolta un negozio vi darà una copia senza farvela pagare. Questo non rende comunque il software libero. Prezzo o non prezzo, il programma non è libero perché gli utenti non hanno libertà.

Dal momento che il software libero non è una questione di prezzo, un basso prezzo non vuol dire che il programma sia più libero o più vicino a esserlo. Perciò, se state ridistribuendo copie di software libe-

ro, potreste anche venderle a un prezzo consistente e guadagnarci. Ridistribuire il software libero è una attività buona e legale; se la fate, potete anche trarne profitto.

Il software libero è un progetto comunitario, e chiunque vi dipenda dovrebbe cercare modalità per contribuire a costruire la comunità. Per un distributore il modo di farlo è dare parte del profitto alla Free Software Foundation o a qualche altro progetto di sviluppo di software libero. Finanziando lo sviluppo, potete far progredire il mondo del software libero.

Distribuire software libero è un'opportunità per raccogliere fondi per lo sviluppo. Non sprecatela!

Per contribuire ai fondi, avete bisogno di avere un sovrappiù. Se fate pagare un prezzo troppo basso, non vi avvanzerà niente per sostenere lo sviluppo.

### **Può un prezzo della distribuzione più alto danneggiare alcuni utenti?**

La gente talvolta si preoccupa del fatto che un alto compenso per la distribuzione possa mettere il software libero fuori dalla portata degli utenti che non hanno molto denaro. Con il software proprietario, un alto compenso fa esattamente questo – ma il software libero è diverso.

La differenza è che il software libero tende naturalmente a diffondersi, e ci sono molti modi per procurarselo.

Coloro che fanno incetta di software cercheranno in tutti i modi di impedirvi di eseguire un programma proprietario senza pagare il prezzo stabilito. Se questo prezzo è alto, sarà difficile per alcuni utenti utilizzare il programma.

Con il software libero, gli utenti non devono pagare il costo della distribuzione per utilizzare il software. Possono copiare il pro-

gramma, da un amico che ne abbia una copia o con l'aiuto di un amico che abbia accesso alla rete. Oppure diversi utenti possono unirsi, dividere il prezzo di un CD-ROM e a turno installare il software. Un alto prezzo del CD-ROM non è un grosso ostacolo quando il software è libero.

### **Può un prezzo della distribuzione più alto scoraggiare l'uso del software libero?**

Un altro problema comune è la popolarità del software libero. La gente pensa che un prezzo alto per la distribuzione riduca il numero di utenti o che un prezzo basso è probabile che li incoraggi.

Questo è vero per il software proprietario – ma il software libero è diverso. Con così tanti modi di procurarsi le copie, il prezzo del servizio di distribuzione ha meno effetto sulla sua popolarità.

Alla fine, il numero di persone che utilizza il software libero è determinato principalmente da quanto il software può fare, e dalla facilità di utilizzo. Molti utenti continueranno a utilizzare software proprietario se il software libero non può fare tutto ciò che essi vogliono. Perciò, se vogliamo aumentare il numero di utenti a lungo andare, dobbiamo soprattutto sviluppare più software libero.

Il modo più diretto per farlo è scrivere da sé il software libero o i manuali necessari. Ma se voi li distribuite piuttosto che scriverli, il miglior modo di aiutare è raccogliere i fondi perché altri li scrivano.

### **Anche l'espressione “vendere software” può confondere**

A rigor di termini, “vendere” significa commerciare prodotti per denaro. Vendere una copia di un programma libero è legale, e noi lo incoraggiamo.

Tuttavia, quando la gente pensa di “vendere software”, di solito

immagina di farlo nel modo in cui lo fa la maggior parte delle società: facendo software proprietario piuttosto che libero.

Così, a meno che non vogliate fare precise distinzioni – come le fa questo articolo – noi suggeriamo sia meglio evitare di utilizzare l'espressione “vendere software” e scegliere invece qualche altra espressione. Per esempio, potreste dire “distribuire software libero dietro compenso” – che non è ambiguo.

## **Compensi alti o bassi, e la GPL GNU**

Tranne che per una situazione particolare, la General Public License GNU (GPL GNU) non detta condizioni su quanto potete chiedere per distribuire una copia di software libero. Potete non chiedere niente, chiedere dieci lire, mille lire, o un miliardo di lire. Decidete voi, e il mercato, perciò non lamentatevi con noi se nessuno vuole pagare un miliardo di lire per una copia.

L'unica eccezione si ha nel caso in cui i binari vengono distribuiti senza il corrispondente codice sorgente completo. A coloro che lo fanno la GPL GNU impone di fornire il codice sorgente a una successiva richiesta. Senza un limite al compenso per il codice sorgente, loro potrebbero stabilire un compenso troppo alto da pagare per chiunque – per esempio, un miliardo – e così fingere di rilasciare il codice sorgente che in realtà continuano a mantenere segreto. Perciò, in questo caso, dobbiamo mettere un limite al compenso del sorgente, per assicurare la libertà dell'utente. In situazioni normali, tuttavia, non c'è nessuna giustificazione simile per limitare i compensi per le distribuzioni, perciò non li limitiamo.

Qualche volta le aziende, le cui attività oltrepassano il limite di quello che la GPL GNU permette, richiedono l'autorizzazione, dicendo di “non chiedere nessun pagamento per il software GNU” o simili. In questo modo non vanno da nessuna parte. Il software libe-

ro riguarda la libertà, e far rispettare la GPL vuol dire difendere la libertà. Quando difendiamo la libertà dell'utente, non siamo sviati da questioni secondarie come per esempio quanto compenso venga richiesto per una distribuzione. La libertà è il problema, l'intero e solo problema.

Originariamente scritto nel 1996. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# **Il software libero ha bisogno di documentazione libera**

Il più grande difetto nei sistemi operativi liberi non sta nel software – è la mancanza di buoni manuali liberi da poter includere in questi sistemi. Molti dei programmi più importanti non hanno un manuale completo. La documentazione è una parte essenziale di qualunque pacchetto di software; quando un pacchetto importante di software libero è fornito senza un manuale libero, si ha una grossa lacuna. A tutt'oggi abbiamo molte di queste lacune.

Una volta, molti anni fa, pensai di imparare il Perl. Presi una copia di un manuale libero, ma lo trovai difficile da leggere. Quando chiesi alternative agli utilizzatori del Perl mi dissero che c'erano manuali introduttivi migliori – ma non erano liberi.

Come mai? Gli autori dei buoni manuali li avevano scritti per la O'Reilly Associates che li pubblicava con termini restrittivi – divieto di copia, divieto di modificazione, sorgenti non disponibili – il che li escludeva dalla comunità del software libero.

Non era la prima volta che accadeva questo tipo di cose, e (con grande perdita per la nostra comunità) non era neanche l'ultima. Gli editori di manuali proprietari da allora hanno indotto molti degli autori a porre limitazioni ai loro manuali. Molte volte ho sentito un utente di software GNU parlarmi entusiasticamente di un manuale che stava scrivendo, che si aspettava avrebbe aiutato il progetto GNU – ma poi le mie speranze si spezzavano, quando procedeva a spiegarmi che aveva firmato un contratto con un editore che ne avrebbe ristretto l'uso cosicché non avremmo potuto usarlo.

Dato che scrivere in un buon inglese è un'abilità rara fra i programmatori, possiamo permetterci a malapena di perdere manuali in questo modo.

La documentazione libera, come il software libero, è una questione di libertà, non di prezzo. Il problema con questi manuali non era che la O'Reilly Associates imponesse un prezzo per le copie stampate – che di per sé va bene (anche la Free Software Foundation vende copie dei manuali GNU liberi). Ma i manuali GNU sono disponibili in forma sorgente, mentre questi manuali sono disponibili solo su carta. I manuali GNU vengono forniti con il permesso di copiarli e modificarli; i manuali del Perl no. Il problema sono queste restrizioni.

I criteri per un manuale libero sono sostanzialmente gli stessi del software libero: è questione di dare a tutti gli utenti certe libertà. La redistribuzione (compresa quella commerciale) deve essere permessa, così il manuale potrà accompagnare ogni copia del programma, sia online che su carta. Anche il permesso di fare modifiche è cruciale.

Come regola generale non credo che sia essenziale per le persone avere il permesso di modificare ogni sorta di articoli e libri. I problemi relativi agli scritti non sono necessariamente identici a quelli del software. Per esempio, non penso che io o voi siamo obbligati a dare il permesso di modificare articoli come questo in cui descriviamo le nostre azioni e i nostri punti di vista.

Ma c'è una ragione particolare per cui la libertà di effettuare modifiche è cruciale per la documentazione del software libero. Quando le persone esercitano il loro diritto di modificare il software, e aggiungono o cambiano funzionalità, se coscienziosamente cambiassero anche il manuale, potrebbero fornire documentazione accurata e utilizzabile per il programma modificato. Un manuale

che proibisce ai programmatori di essere coscienti e completare il lavoro, o che più precisamente richiede loro di scrivere da capo un nuovo manuale se cambiano il programma, non risponde alle necessità della nostra comunità.

Mentre una proibizione generale sulle modifiche è inaccettabile, alcuni tipi di limitazione sui metodi delle modifiche non pongono problemi. Ad esempio, vanno bene quelle di mantenere la nota di copyright dell'autore originale, i termini di distribuzione, o la lista degli autori. Non c'è problema anche nel richiedere che versioni modificate diano nota del loro essere tali, e anche che abbiano intere sezioni che non possono essere tolte o cambiate, fintanto che hanno a che fare con argomenti non tecnici (alcuni manuali GNU le hanno).

Questo tipo di restrizioni non sono un problema perché, dal punto di vista pratico, non impediscono al programmatore cosciente di adattare il manuale per corrispondere alle modifiche del programma. In altre parole, non impediscono alla comunità del software libero di fare pieno uso del manuale.

Tuttavia deve essere possibile modificare tutti i contenuti tecnici del manuale, e distribuire il risultato attraverso tutti i mezzi consueti, attraverso tutti i canali usuali; altrimenti le restrizioni bloccherebbero la comunità, il manuale non sarebbe libero e così ci servirebbe un altro manuale.

Sfortunatamente, è spesso difficile trovare qualcuno che scriva un altro manuale quando esiste un manuale proprietario. L'ostacolo è che molti utenti pensano che un manuale proprietario è sufficiente – così non vedono la necessità di scrivere un manuale libero. Non vedono che i sistemi operativi liberi hanno una lacuna che deve essere riempita.

Perché gli utenti pensano che i manuali proprietari siano suffi-

cienti? Alcuni non hanno considerato il problema. Spero che questo articolo faccia qualcosa per cambiare tutto ciò.

Altri utenti considerano i manuali proprietari accettabili per le stesse ragioni per cui molte persone considerano accettabile il software proprietario: giudicano soltanto in termini pratici e non usano la libertà come criterio. Queste persone hanno diritto alle loro opinioni, ma poiché queste opinioni derivano da valori che non includono la libertà, essi non sono di esempio per quelli di noi che danno importanza alla libertà.

Per favore, spargete la voce riguardo a questo problema. Continuiamo a perdere manuali a favore di pubblicazioni proprietarie. Se spargiamo la voce che i manuali proprietari non sono sufficienti, forse la prossima persona che vuole aiutare il progetto GNU scrivendo documentazione si renderà conto, prima che sia troppo tardi, che deve anzitutto renderla libera.

Incoraggiamo inoltre gli editori commerciali a vendere manuali liberi con permesso d'autore invece di manuali proprietari. Una maniera di far questo è di controllare i termini di distribuzione di un manuale prima di comprarlo, e preferire manuali con permesso d'autore [copyleft] a quelli senza permesso d'autore.

[Nota: La Free Software Foundation mantiene una pagina web che elenca libri di documentazione libera pubblicati da altri editori, <http://www.gnu.org/doc/other-free-books.html>]


Originariamente scritto nel 2000. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# La canzone del software libero


Sulla melodia della canzone folk bulgara “Sodi Moma”.

whistle



Join us now and share the soft-ware You'll be  
Hoard-ers may get piles of mo-ney, That is  
When we have e-nough free soft-ware At our,

orchestral strings




4



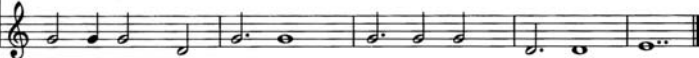
free hack - ers, you'll be free. Join us now and  
true, hack - ers, that is true. But they can - not  
call, hack - ers, at our call, We'll kick out those



8



share the soft-ware You'll be — free hack - ers, you'll be free.  
help their neigh-bors That's not — good, hack - ers, that's not good.  
dir - ty li-cens-es Ev - er more, hack - ers, Ev - er more.



*Le liriche in italiano:*

Unitevi a noi e condividete il software,  
Sarete liberi, hacker, sarete liberi

Qualche avido potrà fare mucchi di soldi,  
È vero, hacker, è vero  
Ma non potrà aiutare i vicini  
Questo non va bene, hacker, non va bene

Quando avremo abbastanza software libero  
A disposizione, hacker, a disposizione  
Getteremo via quelle sporche licenze  
Sempre più, hacker, sempre di più

Unitevi a noi e condividete il software,  
Sarete liberi, hacker, sarete liberi

Originariamente scritto nel 1993. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

**Parte seconda**

**Copyright,  
copyleft e brevetti**



# Il diritto di leggere

*Tratto da “La strada verso Tycho”, raccolta di articoli sugli eventi precedenti la Rivoluzione Lunaria, pubblicata a Luna City nel 2096.* Per Dan Halbert, la strada verso Tycho si rivelò all'epoca del college – quando Lissa Lenz gli chiese in prestito il computer. Il suo si era rotto e, a meno di non poterne usare un altro, avrebbe mancato la scadenza per la presentazione del progetto di metà corso. Non osava chiederlo a nessun altro tranne Dan, ponendolo così di fronte a un grave dilemma. Dan aveva il dovere di aiutarla – ma una volta prestatole il computer, Lissa avrebbe potuto leggerne ogni libro. A parte il rischio di finire in carcere per molti anni per aver consentito ad altri l'accesso a tali libri, inizialmente Dan rimase assai colpito dall'idea stessa di una simile eventualità. Come chiunque altro, fin dalle elementari gli era stato insegnato quanto fosse malvagio e sbagliato condividere i libri – qualcosa che soltanto i pirati si azzardavano a fare.

Ed era impossibile che la SPA – la Software Protection Agency, l'Agenzia per la tutela del software – avesse mancato di smascherarlo. Nel corso sul software, Dan aveva imparato che ogni libro era dotato di un apposito sistema di monitoraggio sul copyright in grado di riportare all'Agenzia centrale per le licenze quando e dove ne fosse avvenuta la lettura, e da parte di chi. (Questi dati venivano poi utilizzati nelle indagini per la cattura dei pirati della lettura, ma anche per vendere ai grossisti i profili sugli interessi personali dei singoli). La prossima volta che il suo computer fosse stato collegato al network centrale, l'Agenzia l'avrebbe scoperto. In quanto pro-

prietario del computer, sarebbe stato lui a subire la punizione più pesante – per non aver fatto abbastanza nella prevenzione di quel crimine.

Naturalmente non era affatto scontato che Lissa avesse intenzione di leggere i libri presenti sul computer. Forse lo avrebbe usato soltanto per finire la relazione di metà corso. Ma Dan sapeva che la sua condizione sociale non elevata le consentiva di pagare a malapena le tasse scolastiche, meno che mai le tariffe per l'accesso alla lettura dei testi. Una situazione che comprendeva bene; lui stesso era stato costretto a chiedere in prestito dei soldi per pagare le quote necessarie alla consultazione di tutte le ricerche disponibili. (Il dieci per cento di tali quote andava direttamente agli autori delle ricerche; poichè Dan puntava alla carriera accademica, poteva sperare di ripagare il prestito con la percentuale sulle proprie ricerche, nel caso venissero consultate con una certa frequenza).

Solo più tardi Dan avrebbe appreso dell'esistenza di un'epoca passata in cui chiunque poteva recarsi in biblioteca a leggere articoli e ricerche senza dover pagare nulla. E i ricercatori indipendenti avevano accesso a migliaia di pagine, pur in assenza di contributi governativi alle biblioteche. Ma negli anni '90 sia gli editori nonprofit sia quelli commerciali iniziarono a imporre delle tariffe per la consultazione di quei materiali. A partire dal 2047, le biblioteche che offrivano accesso pubblico e gratuito alle opere dei ricercatori non erano altro che una memoria del passato.

Naturalmente esistevano vari modi per ingannare la SPA e l'Agenzia centrale per le licenze. Modalità del tutto illegali. Uno degli studenti che aveva seguito il corso sul software con Dan, Frank Marucci, era entrato in possesso di un programma illecito per il debugging [l'attività di collaudo del software], e lo aveva utilizzato per disattivare il codice di monitoraggio del copyright per la lettura dei

libri. Purtroppo era andato in giro a raccontarlo a troppi amici e uno di loro l'aveva denunciato alla SPA in cambio di una ricompensa in denaro (gli studenti fortemente indebitati erano assai pronti al tradimento). Nel 2047 Frank era in prigione, non per lettura illegale, bensì per il possesso di un debugger.

In seguito Dan avrebbe saputo che tempo addietro a chiunque era consentito il possesso di simili programmi. Circolavano liberamente persino su CD o tramite download via internet. Ma i comuni utenti presero a usarli per superare le restrizioni sul monitoraggio del copyright, e alla fine una sentenza giudiziaria stabilì come questa fosse divenuta pratica comune nell'impiego di tali programmi. Di conseguenza, questi vennero dichiarati illegali e gli sviluppatori di debugger [programma per l'attività di collaudo del software] condannati al carcere.

Pur se i programmatori avevano comunque bisogno di programmi per il debugging, nel 2047 i produttori ne distribuivano soltanto copie numerate, e unicamente a programmatori provvisti di licenza e assicurazione ufficiali. Il debugger a disposizione di Dan nel corso sul software era dotato di uno speciale firewall [sistema a protezione di accessi non autorizzati], in modo da poter essere utilizzato soltanto per gli esercizi in classe.

Onde superare le restrizioni sul monitoraggio del copyright, era altresì possibile installare una versione modificata del kernel di sistema. Dan avrebbe poi scoperto l'esistenza di kernel liberi, perfino di interi sistemi operativi liberamente disponibili, negli anni a cavallo del secolo. Ma non soltanto questi erano illegali, al pari dei debugger – non era comunque possibile installarli senza conoscere la password centrale del computer. Qualcosa che né l'FBI né il servizio-assistenza di Microsoft ti avrebbero mai rivelato.

Dan concluse che non avrebbe potuto semplicemente prestare il

computer a Lissa. Ma nemmeno poteva rifiutarsi di aiutarla, perché l'amava. Qualsiasi opportunità di parlare con lei lo riempiva di gioia. E il fatto che avesse chiesto aiuto proprio a lui poteva significare che anche lei gli voleva bene.

Dan risolse il dilemma con una decisione perfino più impensabile – le prestò il computer rivelandole la propria password. In tal modo, se Lissa avesse letto i libri ivi contenuti, l'Agenzia centrale avrebbe ritenuto che fosse Dan a leggerli. Si trattava pur sempre di un crimine, ma la SPA non avrebbe potuto scoprirlo in maniera automatica. Ciò avrebbe potuto avvenire soltanto dietro un'esplicita denuncia di Lissa.

Naturalmente, se la scuola avesse scoperto che aveva rivelato la password personale a Lissa, entrambi avrebbero chiuso con la carriera scolastica, a prescindere dall'utilizzazione o meno di tale password. Qualsiasi interferenza con i dispositivi predisposti da un istituto accademico sul monitoraggio nell'impiego dei computer da parte degli studenti provocava delle sanzioni disciplinari. Non importava se si fossero arrecati o meno danni materiali – il crimine consisteva nel rendere difficile il controllo sui singoli da parte degli amministratori locali. I quali potevano cioè presumere che tale comportamento nascondesse ulteriori attività illegali, e non avevano bisogno di sapere quali fossero.

In circostanze simili generalmente gli studenti non venivano espulsi – almeno non in maniera diretta. Se ne impediva piuttosto l'accesso ai sistemi informatici dell'istituto, provocandone così l'inevitabile voto insufficiente in ogni corso.

Più tardi Dan avrebbe scoperto come una siffatta procedura fosse stata implementata nelle università a partire dagli anni '80, quando gli studenti iniziarono a fare ampio uso dei computer accademici. In precedenza, le università seguivano una strategia diversa

per le questioni disciplinari, punendo soltanto le attività che provocavano danni materiali, non quelle che potevano suscitare appena dei sospetti.

Lissa non denunciò Dan alla SPA. La decisione di aiutarla condusse al loro matrimonio, e li spinse anzi a mettere in discussione quel che era stato insegnato loro fin da piccoli riguardo la pirateria. I due presero a documentarsi sulla storia del copyright, sulle restrizioni sulla copia in vigore in Unione Sovietica e perfino sul testo originale della Costituzione degli Stati Uniti. Decisero poi di trasferirsi su Luna, per unirsi agli altri che in maniera analoga gravitavano lontano dalla lunga mano della SPA. Quando nel 2062 scoppiò la rivolta di Tycho, il diritto universale alla lettura ne costituì subito uno degli obiettivi prioritari.

## Nota dell'autore

Il diritto di leggere è una battaglia che si va combattendo ai giorni nostri. Pur se potrebbero passare 50 anni prima dell'oscuramento dell'attuale stile di vita, gran parte delle procedure e delle norme specifiche descritte sopra sono state già proposte; parecchie fanno parte integrante del corpo legislativo negli Stati Uniti e altrove. Nel 1998 il Digital Millennium Copyright Act statunitense ha stabilito le basi legali per limitare la lettura e il prestito di libri computerizzati (e anche altri materiali). Una direttiva sul copyright emanata nel 2001 dall'Unione Europea ha imposto analoghe restrizioni.

Esiste però un'eccezione: l'idea che l'FBI e Microsoft possano tenere segreta la password centrale di ogni personal computer, senza informarne l'utente, non ha trovato spazio in alcun disegno di legge. In questo caso si tratta di una estrapolazione di quanto contenuto nel testo sul chip Clipper e in analoghe proposte sulle chiavi di decifrazione avanzate dal governo statunitense. Ciò in aggiunta a una tendenza in atto da tempo: con sempre maggior frequenza i sistemi informatici vengono progettati per fornire agli operatori in remoto il controllo proprio su quegli utenti che utilizzano tali sistemi.

È tuttavia evidente come ci si stia avviando verso un simile scenario. Nel 2001 il senatore Hollings, con il sostegno economico di Walt Disney, ha presentato una proposta di legge denominata Security Systems Standards and Certification Act (ora sotto il nuovo titolo di Consumer Broadband and Digital Television Promotion Act) che prevede l'introduzione obbligatoria in ogni nuovo computer di apposite tecnologie atte a impedire ogni funzione di copia e impossibili da superare o disattivare da parte dell'utente.

Nel 2001 gli Stati Uniti hanno avviato il tentativo di utilizzare il

trattato denominato Free Trade Area of the Americas per imporre le medesime norme a tutti i paesi dell'emisfero occidentale. Questo è uno dei cosiddetti trattati a favore del "libero commercio", in realtà progettati per garantire all'imprenditoria maggior potere nei confronti delle strutture democratiche; l'imposizione di legislazioni quali il Digital Millenium Copyright Act è tipico dello spirito che li pervade. La Electronic Frontier Foundation sta chiedendo a tutti di spiegare ai propri governi i motivi per cui occorre opporsi a questo progetto.

La SPA, che in realtà sta per Software Publishers Association, l'Associazione degli editori di software statunitensi, è stata sostituita in questo ruolo simil-repressivo dalla BSA, Business Software Alliance, l'alleanza per il software commerciale. Attualmente questa non ricopre alcuna funzione ufficiale in quanto organo repressivo; ufficiosamente però agisce in quanto tale. Ricorrendo a metodi che ricordano i tempi dell'ex-Unione Sovietica, la Business Software Alliance invita gli utenti a denunciare amici e colleghi di lavoro. Una campagna terroristica lanciata in Argentina nel 2001 minacciava velatamente quanti dividevano il software di possibili stupri una volta incarcerati.

Quando venne scritto il racconto di cui sopra, la Software Publishers Association stava minacciando i piccoli fornitori di accesso a internet, chiedendo loro di consentire alla stessa associazione il monitoraggio dei propri utenti. Sotto il peso delle minacce, molti fornitori d'accesso tendono ad arrendersi perchè impossibilitati ad affrontare le conseguenti spese legali (come riporta il quotidiano *Atlanta Journal-Constitution*, 1 ottobre 1996, pag. D3). Dopo essersi rifiutato di aderire a tale richiesta, almeno uno di questi fornitori, Community ConneXion di Oakland, California, ha subito formale denuncia. L'istanza è stata successivamente ritirata dalla

Software Publishers Association, ottenendo però l'approvazione di quel Digital Millenium Copyright Act che le fornisce quel potere che andava cercando.

Le procedure di sicurezza in ambito accademico sopra descritte non sono frutto dell'immaginazione. Ad esempio, quando si inizia a usare un computer di un'università nell'area di Chicago, questo è il messaggio che viene stampato automaticamente:

“Questo sistema può essere utilizzato soltanto dagli utenti autorizzati. Coloro che ne fanno uso privi di apposita autorizzazione, oppure in maniera a questa non conforme, possono subire il controllo e la registrazione, da parte del personale addetto, di ogni attività svolta sul sistema. Nel corso dell'attività di monitoraggio su usi impropri degli utenti oppure durante la manutenzione del sistema, possono essere monitorate anche le attività di utenti autorizzati. Chiunque utilizzi questo sistema fornisce il proprio consenso esplicito al monitoraggio e viene avvisato che, nel caso ciò dovesse rivelare attività illegali o violazioni alle norme universitarie, il personale addetto potrà fornire le prove di tali attività alle autorità universitarie e/o agli ufficiali di polizia”.

Ci troviamo così di fronte a un interessante approccio al Quarto Emendamento della Costituzione statunitense: forti pressioni contro chiunque per costringerlo a dichiararsi d'accordo, in anticipo, sulla rinuncia a ogni diritto previsto da tale emendamento.

Questo il testo del Quarto Emendamento:

“Il diritto degli individui alla tutela della propria persona, abitazione, documenti ed effetti personali contro ogni perquisizione e sequestro immotivato, non potrà essere violato e nessun mandato verrà emesso se non nel caso di causa probabile, sostenuta da giu-

ramento o solenne dichiarazione, riguardanti in particolare la descrizione del luogo soggetto a perquisizione, e gli individui o gli effetti da sequestrare”.

### *Riferimenti:*

- La White Paper dell'amministrazione USA: “Information Infrastructure Task Force, Intellectual property and the National Information Infrastructure: The Report of the Working Group on Intellectual Property Rights” (1995).
- Una spiegazione della suddetta White Paper: “The Copyright Grab”, Pamela Samuelson, *Wired*, gennaio 1996 ([http://www.wired.com/wired/archive/4.01/white\\_paper\\_pr.htm](http://www.wired.com/wired/archive/4.01/white_paper_pr.htm)).
- “Sold Out”, James Boyle, *The New York Times*, 31 marzo 1996
- “Public Data or Private Data”, *The Washington Post*, 4 novembre 1996.
- Union for the Public Domain, organizzazione mirata alla resistenza e al ribaltamento degli eccessivi ampliamenti di potere assegnato al copyright e ai brevetti (<http://www.public-domain.org>).

Questo saggio è stato pubblicato per la prima volta (nell'originale inglese) sul numero di febbraio 1997 della rivista *Communications of the ACM* (volume 40, numero 2). La Nota dell'autore è stata aggiornata nel 2002. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# L'interpretazione sbagliata del copyright una serie di errori

Qualcosa di strano e pericoloso sta accadendo alle legislazioni in materia di copyright [diritto d'autore]. Come stabilito dalla Costituzione degli Stati Uniti, il copyright esiste a beneficio degli utenti – chiunque legga dei libri, ascolti della musica, guardi dei film o utilizzi del software – non nell'interesse degli editori o degli autori. Tuttavia, anche quando la gente tende sempre più a rifiutare e disubbidire alle restrizioni sul copyright imposte “a loro beneficio”, il governo statunitense vi aggiunge ulteriori restrizioni, nel tentativo di intimorire il pubblico e costringerlo a ubbidire sotto la pressione di nuove e pesanti sanzioni.

In che modo le procedure sul copyright sono divenute diametralmente opposte agli obiettivi dichiarati? E come possiamo fare in modo che tornino ad allinearsi con tali obiettivi? Per comprendere la situazione, è bene partire dando un'occhiata alle radici delle leggi sul copyright degli Stati Uniti, il testo della stessa Costituzione.

## Il copyright nella Costituzione statunitense

Nella stesura del testo della Costituzione, l'idea che agli autori potesse essere riconosciuto il diritto al monopolio sul copyright venne proposta – e rifiutata. I padri fondatori degli Stati Uniti partirono da una premessa diversa, secondo cui il copyright non è un diritto naturale degli autori, quanto piuttosto una condizione artificiale concessa loro per il bene del progresso. La Costitu-

zione permette l'esistenza di un sistema sul copyright tramite il seguente paragrafo (articolo I, sezione 8):

«Il Congresso avrà il potere di promuovere il progresso della scienza e delle arti utili, garantendo per periodi di tempo limitati ad autori e inventori il diritto esclusivo ai rispettivi testi scritti e invenzioni».

La Corte Suprema ha ripetutamente affermato che promozione del progresso significa apportare dei benefici agli utenti delle opere sotto copyright. Ad esempio, nella causa *Fox Film v. Doyal*, la Corte ha sostenuto:

«L'unico interesse degli Stati Uniti e l'obiettivo primario nell'assegnazione del monopolio [sul copyright] va cercato nei benefici generali derivanti al pubblico dai lavori degli autori».

Questa decisione fondamentale illustra il motivo per cui nella Costituzione statunitense il copyright non venga *imposto*, bensì soltanto *consentito* in quanto opzione possibile – e perché se ne ipotizza la durata per “periodi di tempo limitati”. Se si trattasse di un diritto naturale, qualcosa assegnato agli autori perché lo meritano, nulla potrebbe giustificare la cessazione dopo un determinato periodo, al pari dell'abitazione di qualcuno che dovesse divenire di proprietà pubblica trascorso un certo tempo dalla sua costruzione.

## Il “contratto sul copyright”

Il sistema del copyright funziona tramite l'assegnazione di privilegi e relativi benefici per editori e autori. Ma non lo fa nell'interesse di costoro, quanto piuttosto per modificarne il comportamento: per fornire un incentivo agli autori a scrivere di più e agli editori a pubblicare di più. In effetti, il governo utilizza i diritti naturali del pubblico, a nome di quest'ultimo, come parte di una trattativa con-

trattuale finalizzata a offrire allo stesso pubblico un maggior numero di opere. Gli esperti legali definiscono questo concetto “contratto sul copyright”. Qualcosa di analogo all’acquisto da parte del governo di un’autostrada o di un aeroplano usando i soldi dei contribuenti, con la differenza che qui il governo spende la nostra libertà anziché il nostro denaro.

Ma l’esistenza di un tale contratto può davvero considerarsi un buon affare per il pubblico? È possibile considerare molti altri accordi alternativi; qual’è il migliore? Ogni singola questione inerente le procedure sul copyright rientra nel contesto di una simile domanda. Se non si comprende pienamente la natura di tale domanda, tenderemo a prendere decisioni errate sulle varie questioni coinvolte.

La Costituzione autorizza l’assegnazione dei poteri del copyright agli autori. In pratica, costoro tipicamente li cedono agli editori; generalmente spetta a questi ultimi, non agli autori, l’esercizio di tali poteri onde trarne la maggior parte dei benefici, pur se agli autori ne viene riservata una piccola porzione. Ne consegue che normalmente sono gli editori a spingere per l’incremento dei poteri conferiti dal copyright. Onde offrire una riflessione più attenta sulla realtà del copyright, piuttosto che sui suoi miti, il presente saggio cita gli editori, anziché gli autori, come detentori dei poteri del copyright. Ci si riferisce inoltre agli utenti delle opere sotto copyright con il termine di “lettori”, pur se non sempre s’intende l’azione di leggere, perché “utenti” è troppo astratto e lontano.

### **Primo errore: “il raggiungimento di un equilibrio”**

Il contratto sul copyright pone il pubblico al primo posto: il beneficio per il lettore è un fine in quanto tale; i benefici (nel caso esistano) per gli editori non rappresentano altro che un mezzo per il

raggiungimento di quel fine. Gli interessi dei lettori e quelli degli editori sono qualitativamente diseguali nelle rispettive priorità. Il primo passo verso un'errata interpretazione sugli obiettivi del copyright consiste nell'elevare gli interessi degli editori al medesimo livello d'importanza di quelli dei lettori.

Si dice spesso che la legislazione statunitense sul copyright mira al "raggiungimento di un equilibrio" tra gli interessi degli editori e quelli dei lettori. I sostenitori di questa interpretazione la presentano come una riproposizione delle posizioni di partenza affermate nella Costituzione; in altri termini, ciò viene ritenuto l'equivalente del contratto sul copyright.

Ma le due interpretazioni sono tutt'altro che equivalenti: sono differenti a livello concettuale, come pure nelle implicazioni annesse. L'idea di equilibrio dà per scontato che gli interessi di editori e lettori differiscano per importanza soltanto a livello quantitativo, rispetto a "quanto peso" va assegnato a tali interessi e in quali circostanze questi vadano applicati. Allo scopo di inquadrare la questione in un simile contesto, spesso si ricorre al concetto di "partecipazione equa"; in tal modo si assegna il medesimo livello d'importanza a ciascun tipo d'interesse per quanto concerne le decisioni sulle procedure applicative. Questo scenario ripudia la distinzione qualitativa tra gli interessi degli editori e quelli dei lettori che è alla radice della partecipazione del governo nelle trattative contrattuali sul copyright.

Le conseguenze di una simile alterazione della situazione appaiono di ampia portata, perché la grande protezione del pubblico inclusa nel contratto sul copyright – l'idea secondo cui i privilegi del copyright possano trovare giustificazione soltanto in nome dei lettori, mai in nome degli editori – viene ripudiata dall'interpretazione del "raggiungimento di un equilibrio". Poichè l'interesse degli editori

è considerato un fine in se stesso, può motivarne i privilegi sul copyright; in altre parole, il concetto di “equilibrio” sostiene che i privilegi possano trovare giustificazione in nome di qualche soggetto che non sia il pubblico.

A livello pratico, la conseguenza di tale concetto di “equilibrio” consiste nel ribaltare l’onere di motivare i cambiamenti da apportare alle legislazioni in materia. Il contratto sul copyright impegna gli editori a convincere i lettori nel cedere loro determinate libertà. Praticamente l’idea di equilibrio capovolge quest’onere, perché in genere non esiste alcun dubbio che gli editori trarranno beneficio dai privilegi aggiuntivi. Così, a meno di non comprovare un danno arrecato ai lettori, sufficiente da “pesare di più” di tale beneficio, siamo inclini a concludere che agli editori vada garantito pressoché qualsiasi privilegio richiesto.

L’idea del “raggiungimento di un equilibrio” tra editori e lettori va respinta, in quanto nega a questi ultimi la priorità cui hanno diritto.

### **Raggiungere un equilibrio con cosa?**

Quando il governo acquista qualcosa per il pubblico, agisce in nome di quest’ultimo; è sua responsabilità ottenere l’accordo più vantaggioso possibile – per il pubblico, non per gli altri soggetti coinvolti nella trattativa.

Ad esempio, quando firma un contratto con degli imprenditori edili per la costruzione di autostrade, il governo tende a spendere la minima quantità possibile di denaro pubblico. Le agenzie statali ricorrono a gare d’appalto competitive per spingere i prezzi al ribasso.

A livello pratico, il prezzo non può risultare pari a zero, perché gli imprenditori non accettano contratti così bassi. Pur in assenza di condizioni particolari, costoro hanno i medesimi diritti di ogni cittadino in una società libera, compreso quello di rifiutare contratti

svantaggiosi; per un imprenditore anche l'offerta più bassa potrebbe rivelarsi sufficiente onde guadagnare qualcosa. Esiste quindi una sorta di equilibrio. Ma non si tratta di un equilibrio deliberatamente cercato tra due interessi che esigono considerazioni particolari. È un equilibrio tra un obiettivo pubblico e le dinamiche del mercato. Il governo tenta di ottenere per i contribuenti motorizzati il miglior contratto possibile nel contesto di una società libera e di un libero mercato.

Nella trattativa contrattuale sul copyright, il governo spende la nostra libertà anziché il nostro denaro. La prima è più preziosa del secondo, motivo per cui la responsabilità del governo nello spenderla in maniera saggia e parsimoniosa è decisamente maggiore di quella relativa alle spese economiche. Lo stato non deve mai porre gli interessi degli editori sullo stesso piano della libertà del pubblico.

### Non “equilibrio” ma “scambio”

L'idea di raggiungere un equilibrio tra gli interessi dei lettori e quelli degli editori è la maniera sbagliata di giudicare le procedure sul copyright, ma in realtà esistono due interessi da soppesare: entrambi riguardano *i lettori*. Questi hanno interesse nella propria libertà per l'utilizzo delle opere pubblicate; a seconda delle circostanze, possono inoltre avere interesse nell'incoraggiare la pubblicazione tramite qualche sistema d'incentivazione.

Il termine “equilibrio”, nelle discussioni in tema di copyright, è divenuto sinonimo di scorciatoia per l'idea di “raggiungere l'equilibrio” tra lettori ed editori. Di conseguenza, l'uso di tale termine per indicare questi due interessi dei lettori provocherebbe confusione – c'è bisogno di un altro termine.

In generale, quando un'entità presenta due obiettivi in parziale conflitto tra loro e non è in grado di raggiungerli entrambi in maniera

completa, la situazione viene definita “scambio”. Pertanto, anziché riferirci al “raggiungimento del giusto equilibrio” tra entità diverse, dovremmo parlare di “trovare il giusto scambio tra il consumo e la conservazione della libertà”.

### **Secondo errore: privilegiare un unico aspetto**

Il secondo errore delle politiche sul copyright consiste nell’adottare l’obiettivo di massimizzare la quantità di opere pubblicate, non soltanto di incrementarle. L’erroneo concetto del “raggiungimento del giusto equilibrio” aveva posto gli editori al medesimo livello dei lettori; questo secondo errore li eleva molto al di sopra.

Quando compriamo qualcosa, generalmente non acquistiamo l’intera quantità di articoli disponibili in magazzino o il modello più costoso. Preferiamo piuttosto risparmiare per ulteriori compere, acquistando soltanto quanto ci occorre di una determinata merce, e scegliendo un modello di buon livello anziché della qualità migliore in assoluto. Sulla base del principio della diminuzione del profitto, spendere tutti i soldi per un unico articolo si rivela con tutta probabilità una gestione inefficiente delle risorse disponibili.

La diminuzione del profitto si applica al copyright come a qualsiasi acquisto. Le prime libertà che dovremmo scambiare sono quelle di cui potremo fare più facilmente a meno, pur offrendo il maggiore incoraggiamento possibile alla pubblicazione. Mentre barattiamo le libertà aggiuntive via via più familiari, ci rendiamo conto come ogni scambio comporti un sacrificio maggiore del precedente, portando al contempo un minore incremento all’attività letteraria. Assai prima che tale incremento raggiunga quota zero, possiamo ben dire che ciò non giustifica ulteriori aumenti di prezzo; dovremmo quindi raggiungere un accordo che preveda l’aumento

del numero delle pubblicazioni in circolazione, senza tuttavia arrivare al massimo possibile.

L'accettazione dell'obiettivo di massimizzare la quantità delle pubblicazioni comporta il rifiuto aprioristico di tutti questi accordi più saggi e vantaggiosi – tale posizione impone al pubblico di cedere quasi tutta la propria libertà di utilizzo delle opere pubblicate, in cambio di un incremento modesto delle pubblicazioni.

### **La retorica della massimizzazione**

In pratica, l'obiettivo di massimizzare le pubblicazioni prescindendo dal prezzo imposto alla libertà si fonda sulla diffusa retorica secondo cui la copia pubblica sia qualcosa di illegale, ingiusto e intrinsecamente sbagliato. Ad esempio, gli editori definiscono “pirati” coloro che copiano, termine dispregiativo mirato ad equiparare l'assalto a una nave e la condivisione delle informazioni con il vicino di casa. (Quel termine dispregiativo era già stato impiegato dagli autori per descrivere quegli editori che avevano scovato dei modi legali per pubblicare edizioni non autorizzate; il suo utilizzo attuale da parte degli editori riveste un significato pressoché opposto). Questa retorica ripudia direttamente le basi costituzionali a supporto del copyright, ma si presenta come rappresentativa dell'inequivocabile tradizione del sistema legale americano.

In genere la retorica del “pirata” viene accettata perché inonda a tal punto tutti i media che pochi riescono ad afferrarne la radicalità. Si dimostra efficace perché, se la copia a livello pubblico è fondamentalmente qualcosa di illegittimo, non potremmo mai obiettare alla richiesta degli editori di cedere quella libertà che ci appartiene. In altre parole, quando il pubblico viene sfidato a spiegare perché gli editori non dovrebbero ottenere ulteriori poteri, il moti-

vo più importante di tutti – “vogliamo copiare” – subisce una degradazione aprioristica.

Ciò non lascia spazio per controbattere l'incremento di potere assegnato al copyright se non ricorrendo a questioni collaterali. Di conseguenza oggi l'opposizione al maggior potere del copyright poggia quasi esclusivamente su tali questioni collaterali, e non osa mai citare la libertà di distribuire delle copie in quanto legittimo valore pubblico.

A livello pratico, l'obiettivo della massimizzazione consente agli editori di sostenere che “una determinata pratica sta portando alla riduzione delle vendite - o crediamo possa farlo – così riteniamo che ciò sia causa della diminuzione di una quantità imprecisata di pubblicazioni, e di conseguenza occorre proibirla”. Siamo portati a credere all'oltraggiosa conclusione secondo cui il bene pubblico vada misurato dalle vendite degli editori. Quello che va bene per i Grandi Media va bene per gli Stati Uniti.

### **Terzo errore: massimizzare il potere degli editori**

Una volta riconosciuto agli editori l'assenso a una politica mirata alla massimizzazione della quantità di pubblicazioni in circolazione, costi quel che costi, il passo successivo è quello di ritenere che ciò significhi assegnare loro i massimi poteri possibili – ricorrendo al copyright per regolamentare ogni impiego immaginabile di un'opera, oppure applicando altri strumenti legali dall'effetto analogo, tipo le licenze accettate automaticamente dall'utente nel momento in cui apre la confezione originale di un prodotto. Quest'obiettivo, che implica l'abolizione di ogni uso legittimo e del diritto alla prima vendita, viene perseguito con forza a ogni livello governativo, dai singoli stati USA alle organizzazioni internazionali.

Si tratta una procedura errata perché norme sul copyright eccessi-

vamente rigide impediscono la creazione di opere nuove e utili. Ad esempio, Shakespeare prese in prestito la trama di alcuni suoi testi teatrali da altri lavori in circolazione già da alcuni decenni; applicando a quell'epoca le odierne norme sul copyright, le sue opere avrebbero dovuto considerarsi illegali.

Pur mirando alla maggiore quantità possibile di pubblicazioni, volendo ignorarne il prezzo ai danni del pubblico, è sbagliato arrivarci massimizzando i poteri degli editori. Come mezzo per la promozione del progresso, ciò si rivela controproducente.

### **I risultati dei tre errori**

L'attuale tendenza delle legislazioni sul copyright è quella di concedere agli editori maggiori poteri per periodi di tempo più lunghi. Il principio concettuale del copyright, che emerge distorto a seguito della serie di errori sopra illustrati, raramente offre la base per poter dire no a tale tendenza. A parole i legislatori sostengono l'idea del copyright al servizio del pubblico, mentre in realtà cedono a qualunque richiesta degli editori.

Ad esempio, così si è espresso il senatore statunitense Hatch nel 1995, durante la presentazione del disegno di legge S. 483 finalizzato all'estensione dei termini del copyright di ulteriori 20 anni: «Credo che oggi il punto sia quello di dare una risposta alla domanda se gli odierni termini del copyright possano tutelare adeguatamente gli interessi degli autori e alla questione connessa se quei termini possano continuare a fornire un sufficiente incentivo per la creazione di nuove opere».

Questa legge ha esteso il copyright su opere già pubblicate, scritte a partire dal 1920. La modifica è stata un regalo agli editori senza alcun possibile beneficio per il pubblico, poichè è impossibile

aumentare in maniera retroattiva il numero di libri pubblicati allora. Tuttavia, ciò costa al pubblico una libertà oggi significativa – la redistribuzione dei libri del passato.

La normativa estende inoltre il copyright di opere che devono essere ancora scritte. Per i lavori su commissione, il copyright durerà 95 anni invece degli attuali 75. In teoria ciò dovrebbe rivelarsi un maggiore incentivo per la creazione di nuove opere; ma qualunque editore che sostenga la necessità di un simile incentivo dovrebbe motivarlo con le previsioni di bilancio fino all'anno 2075.

Inutile aggiungere che il Congresso non ha posto in dubbio gli argomenti degli editori: la legislazione per l'estensione del copyright è stata approvata nel 1998. È stata chiamata Sonny Bono Copyright Term Extension Act, riprendendo il nome di uno dei proponenti poi scomparso in quell'anno. La vedova, che ne ha proseguito il mandato parlamentare, ha rilasciato la seguente dichiarazione:

«In realtà, Sonny voleva far durare il copyright all'infinito. Qualcuno dello staff mi ha informato che ciò violerebbe la Costituzione. Vi invito tutti a lavorare con me per rafforzare le norme sul copyright in ogni modo possibile. Come sapete, esiste anche una proposta di Jack Valenti per farlo durare indefinitamente meno un giorno. Forse la commissione potrebbe prenderla in esame nel corso della prossima sessione congressuale».

La Corte Suprema ha accettato di esaminare la richiesta dell'annullamento di tali norme sulla base del fatto che un'estensione retroattiva sia contraria all'obiettivo costituzionale della promozione del progresso.

Un'altra legge, approvata nel 1996, ha trasformato in reato grave la copia, in quantità sufficientemente elevate, di qualsiasi lavoro pubblicato, anche nel caso di successiva distribuzione agli amici per

pura gentilezza. In precedenza ciò non veniva affatto considerato reato negli Stati Uniti.

Una legislazione finanche peggiore, il Digital Millennium Copyright Act (DMCA), è stata progettata per imporre nuovamente protezioni anti-copia (detestate dagli utenti informatici), rendendo reato ogni infrazione a tali protezioni, o perfino la pubblicazione di informazioni sul modo di superarle. Questa legge dovrebbe essere chiamata “Domination by Media Corporations Act” (legge per la dominazione delle corporation dei media) perché consente di fatto agli editori la possibilità di scrivere leggi sul copyright a proprio vantaggio. Queste norme permettono loro l'imposizione di qualsiasi tipo di restrizioni sull'utilizzo di un'opera, con le annesse sanzioni repressive, purché le opere siano dotate di qualche tipo di critica o di licenza onde poterle applicare.

Una delle tesi a sostegno di questa legge era che sarebbe servita all'implementazione di un recente trattato mirato all'espansione dei poteri del copyright. Il trattato è stato promulgato dalla World Intellectual Property Organization, entità in cui dominano gli interessi dei detentori di copyright e di brevetti, con l'aiuto della pressione esercitata dall'amministrazione Clinton; poiché il trattato non fa altro che ampliare il potere del copyright, è assai dubbio che possa servire gli interessi del pubblico in altri paesi. In ogni caso, la normativa andò ben oltre quanto richiesto dal trattato stesso.

Le biblioteche costituirono un elemento chiave nell'opposizione a quella proposta, particolarmente riguardo alle norme che impedivano le varie forme di copia considerate “uso legittimo”. Come hanno risposto gli editori? L'ex deputato Pat Schroeder, attualmente impegnato in azioni di lobby per conto della Association of American Publisher, l'Associazione degli editori statunitensi, ha sostenuto che “gli editori non possono aderire alle richieste [delle biblio-

teche]”. Poiché queste ultime chiedevano semplicemente di mantenere parte dello status quo, si potrebbe replicare chiedendosi come abbiano fatto gli editori a sopravvivere fino a oggi.

Il parlamentare Barney Frank, nel corso di una riunione con il sottoscritto e altri oppositori della legge, mostrò fino a che punto sia stato travisato il concetto di copyright incluso nella costituzione. Secondo il deputato statunitense, occorre stabilire urgentemente nuovi poteri, sostenuti da pene severe, perché “l’industria cinematografica è preoccupata”, come pure “il settore discografico” e “altre industrie”. Allora gli ho chiesto: «Ma ciò sarebbe forse a favore dell’interesse pubblico?». La sua replica è stata: «Perché mai tiri fuori l’interesse pubblico? Queste persone creative non devono cedere i propri diritti a favore dell’interesse pubblico!». Così “l’industria” viene identificata con le “persone creative” cui dà lavoro, il copyright è trattato come un diritto che le appartiene e la costituzione viene completamente ribaltata. IL DMCA è stato approvato nel 1998. Nella stesura finale si legge che l’uso legittimo rimane formalmente tale, ma gli editori hanno la facoltà di vietare tutto il software o l’hardware necessario per poterlo mettere in pratica. Di fatto, l’uso legittimo viene proibito. Sulla base di questa legge, l’industria cinematografica ha imposto la censura sul software libero per la lettura e la visione dei DVD, e perfino sulle relative informazioni. Nell’aprile 2001 il professor Edward Felten della Princeton University, minacciato di denuncia dalla Recording Industry Association of America (RIAA), ha ritirato una ricerca scientifica in cui illustrava quanto aveva imparato sul sistema cifrato proposto per impedire l’accesso alla musica registrata.

Stiamo inoltre assistendo all’avvento di libri elettronici (e-book) che cancellano molte delle libertà tipiche del lettore tradizionale – ad esempio, quella di prestare il libro a un amico, di rivenderlo a un libreria

dell'usato, di prenderlo in prestito da una biblioteca, di acquistarlo senza dover fornire le proprie generalità al database aziendale, perfino la libertà di poterlo rileggere. Generalmente i libri elettronici cifrati impediscono tutte queste libertà – è possibile leggerli soltanto grazie a un particolare software segreto, progettato per imporre simili restrizioni al lettore.

Non acquisterò mai uno di questi e-book crittati e protetti, e spero che anche voi li rifiuterete. Se un libro elettronico non offre le medesime libertà di un tradizionale volume cartaceo, non accettatelo!

Chiunque diffonda in modo indipendente un software in grado di leggere gli e-book cifrati rischia di andare in galera. Nel 2001 un programmatore russo, Dmitry Sklyarov, venne arrestato mentre si trovava negli Stati Uniti per intervenire a una conferenza, perché aveva scritto un tale programma in Russia, dove ciò era pienamente legale. Ora anche la Russia sta varando una legge per vietare simili attività, e recentemente l'Unione Europea ne ha adottata una analoga.

Finora il mercato di massa dei libri elettronici si è dimostrato un fallimento commerciale, ma non perché i lettori abbiano deciso di difendere le proprie libertà; gli e-book sono poco interessanti per altri motivi, tra cui la difficile lettura dei testi sul monitor del computer. A tempi lunghi non possiamo affidare la nostra tutela a questo felice incidente di percorso; il prossimo tentativo di promuovere gli e-book prevede l'utilizzo di "carta elettronica" – oggetti somiglianti ai comuni volumi all'interno dei quali scaricare libri elettronici crittati e protetti. Se questa superficie simile alla carta dovesse risultare più leggibile degli odierni monitor, saremo chiamati a tutelare la nostra libertà onde poterla conservare. Nel frattempo gli e-book vanno aprendosi un mercato di nicchia: la New York Uni-

versity e altri istituti richiedono agli studenti di acquistare i libri di testo nel formato elettronico protetto.

L'industria dei media non è ancora soddisfatta. Nel 2001 il senatore Hollings, sovvenzionato dalla Disney, ha presentato una proposta di legge chiamata "Security Systems Standards and Certification Act" (SSSCA), in seguito rinominata Consumer Broadband and Digital Television Promotion Act, la quale prevede la presenza in tutti i computer (e altri apparecchi digitali per la registrazione e la lettura) di sistemi anti-copia imposti dal governo. Ciò rappresenta l'obiettivo finale dell'industria, ma il primo punto all'ordine del giorno mira a vietare qualunque dispositivo in grado di intervenire sulla sintonia della HDTV (High Definition TV, la TV digitale ad alta definizione), a meno che non sia progettato in modo tale da impedire all'utente di "manometterla" (ovvero, di modificarla a scopo personale). Poiché il software libero è tale proprio perché gli utenti possano modificarlo, qui ci troviamo di fronte per la prima volta a una proposta di legge che vieta esplicitamente il software libero per determinate funzioni. Certamente seguiranno analoghi divieti per ulteriori funzioni. Nel caso la Federal Communications Commission statunitense dovesse adottare simili proposte, programmi di software libero già esistenti quali GNU Radio verrebbero censurati.

Occorre mobilitarsi a livello politico per bloccare queste normative (a partire dai seguenti siti web: <http://www.digitalspeech.org> e <http://www.eff.org>).

### **Come arrivare a un contratto equo**

Qual'è la maniera adeguata per stabilire una corretta politica del copyright? Se quest'ultimo è un patto raggiunto a nome del pubblico, dovrebbe innanzitutto servire l'interesse pubblico. Il dovere

del governo, quando si appresta a smerciare la libertà pubblica, è quello di vendere soltanto quanto necessario e al prezzo più caro possibile. Come minimo dovremmo controbilanciare al massimo l'estensione del copyright pur conservando un'analogia quantità di pubblicazioni disponibili.

Poiché è impossibile raggiungere questo livello minimo di libertà tramite gare d'appalto competitive, come nel caso dei progetti edilizi, quale strada conviene seguire?

Un metodo possibile consiste nel ridurre i privilegi del copyright in maniera graduale e osservarne i risultati. Verificando se e quando si raggiunge un livello misurabile nella diminuzione delle pubblicazioni, potremo capire quanto sia il potere del copyright effettivamente necessario per il raggiungimento degli obiettivi del pubblico. Ciò va giudicato tramite l'osservazione diretta, non sulla base di quanto gli editori ritengano debba accadere, perché questi hanno tutto l'interesse a esagerare le previsioni negative in caso ne venga ridotto in qualche modo il potere.

Le politiche sul copyright comprendono svariate dimensioni tra loro indipendenti, le quali possono essere organizzate in maniera separata. Dopo aver raggiunto il livello minimo relativo a una di tali dimensioni, è sempre possibile ridurre altre dimensioni del copyright pur mantenendo la voluta quantità di pubblicazioni. Una dimensione importante del copyright riguarda la sua durata, che tipicamente oggi è dell'ordine di un secolo. La limitazione del monopolio sulla copia a dieci anni, a partire dalla data di pubblicazione di un'opera, potrebbe rivelarsi un buon passo iniziale. Un altro aspetto del copyright – quello concernente la realizzazione di lavori derivati – potrebbe invece continuare a esistere per un periodo più lungo.

Perché si parte dalla data di pubblicazione? Perché il copyright su

lavori inediti non limita direttamente la libertà dei lettori; avere la libertà di copiare un'opera è qualcosa di fittizio quando non ne circolano degli esemplari. Consentire perciò maggior tempo per pubblicare qualcosa non procura alcun danno. Raramente gli autori (che in genere prima della pubblicazione sono titolari del copyright) sceglieranno di ritardare la pubblicazione soltanto per estendere all'indietro l'esaurimento dei termini del copyright.

Perché dieci anni? Perché è una proposta adeguata; a livello pratico possiamo ritenere che questa riduzione produrrà scarso impatto sulle odierne attività editoriali in generale. Per la maggior parte dei settori e dei generi, le opere di successo sono molto remunerative nel giro di qualche anno, e perfino tali opere di successo generalmente vanno fuori catalogo assai prima dei dieci anni. Anche per i testi di consultazione generale, la cui vita d'utilità può estendersi fino a parecchi decenni, un copyright di dieci anni dovrebbe risultare sufficiente: se ne pubblicano regolarmente nuove stesure aggiornate, e gran parte dei lettori preferiranno acquistare l'ultima edizione sotto copyright, anziché una versione di dominio pubblico del decennio precedente.

Dieci anni potrebbe comunque essere un periodo più lungo del necessario: una volta sistemate le cose, potremmo provare un'ulteriore riduzione per meglio rifinire il sistema. Nel corso di una discussione sul copyright durante una manifestazione letteraria, dove proponevo il termine dei dieci anni, un noto autore di testi fantastici che mi sedeva accanto protestò con veemenza, sostenendo che qualunque termine superiore ai cinque anni sarebbe stato intollerabile.

Ma non c'è motivo di applicare la medesima durata a tutti i tipi di lavori. Il mantenimento di una stretta uniformità per le politiche sul copyright non è cruciale all'interesse pubblico, e già le legisla-

zioni correnti prevedono numerose eccezioni per impieghi e ambiti particolari. Sarebbe folle pagare per ogni progetto autostradale la stessa somma necessaria per i progetti più difficili realizzati nelle aree più costose del paese; parimenti folle sarebbe “pagare” ogni tipo di produzione artistica al prezzo più caro in termini di libertà ritenuto necessario per un’opera specifica.

Così forse i romanzi, i dizionari, i programmi informatici, le canzoni, le sinfonie e i film dovrebbero seguire una durata diversa per il copyright, in modo da poterla ridurre per ciascun genere al termine necessario a garantire la pubblicazione di un certo numero di lavori. Forse i film che durano più di un’ora potrebbero avere un copyright di vent’anni, considerandone le spese di produzione. Nel mio settore, la programmazione informatica, tre anni dovrebbero bastare, perché i cicli di produzione sono anche più brevi di un tale periodo.

Un’altra dimensione delle politiche sul copyright riguarda l’estensione dell’uso legittimo: quelle modalità di riproduzione totale o parziale di un lavoro, legalmente consentite anche quando l’opera pubblicata è coperta da copyright. Il primo passo naturale nella riduzione di questa dimensione del potere del copyright consiste nel permettere la copia e la distribuzione tra i singoli individui a livello occasionale, privato e in piccole quantità. In tal modo si eviterebbe l’intrusione della polizia nella vita privata della gente, pur avendo probabilmente scarso effetto sulle vendite dei lavori pubblicati. (Potrebbe rivelarsi necessario intraprendere ulteriori passi legali onde assicurarsi che le licenze incluse automaticamente nelle confezioni originali dei prodotti non possano essere utilizzate in sostituzione del copyright per limitare tali attività di copia). L’esperienza di Napster dimostra che dovremmo altresì consentire la redistribuzione integrale non-commerciale a una comunità più

vasta – quando una parte così ampia del pubblico decide di copiare e condividere qualcosa, considerando assai utili simili pratiche, ciò potrà essere bloccato soltanto ricorrendo a misure draconiane, e il pubblico merita di avere quanto chiede.

Per i romanzi, e in generale per le opere d'intrattenimento, la redistribuzione integrale non-commerciale potrebbe dimostrarsi una libertà sufficiente per i lettori. I programmi informatici, essendo utilizzati per scopi funzionali (portare a termine determinati compiti), richiedono ulteriori libertà aggiuntive, compresa la pubblicazione di versioni migliorate. A motivazione delle libertà che dovrebbero avere gli utenti di software si veda il testo incluso in questo stesso volume “La definizione di software libero”. Tuttavia, un compromesso accettabile potrebbe rivelarsi quello di rendere tali libertà universalmente disponibili soltanto dopo un ritardo di due o tre anni dalla data di pubblicazione del programma.

Questa serie di modifiche finirebbero per allineare il copyright con la volontà del pubblico di usare le tecnologie digitali per copiare. Senza dubbio gli editori considereranno “sbilanciate” simili proposte; potrebbero minacciare di prendere le proprie biglie e andarsene via, ma non lo faranno sul serio, perché il gioco rimarrà comunque redditizio e sarà l'unico possibile.

Mentre si vanno considerando le possibili riduzioni ai poteri del copyright, dobbiamo accertarci che le varie aziende del settore non lo sostituiscano semplicemente con apposite licenze relative all'utente finale. Sarà necessario vietare l'uso di contratti mirati a imporre restrizioni sulla copia che vadano oltre quelle già previste dal copyright. Nel sistema legale statunitense è pratica comune stabilire simili disposizioni su quanto previsto dai contratti non-negoziabili per settori di grande consumo.

## Una nota personale

La mia attività riguarda la programmazione informatica, non l'ambito giuridico. Mi sono interessato alle questioni legate al copyright perché è impossibile evitarle nel mondo delle reti informatiche (essendo internet quella più vasta al mondo). In quanto utente di computer e di reti informatiche per trent'anni, attribuisco molto valore alle libertà che abbiamo abdicato, e a quelle che potremmo perdere in futuro. In quanto autore, rifiuto la mistica romantica che ci considera alla stregua di creature semidivine, immagine spesso citata dagli editori a giustificare l'incremento di poteri sul copyright agli autori, i quali poi li trasferiscono agli stessi editori.

Per la gran parte questo saggio presenta fatti e ragionamenti facilmente verificabili, oltre a una serie di proposte su cui ciascuno di noi può farsi una propria opinione. Chiedo tuttavia al lettore di accettare un solo elemento basato sulla mia parola: autori come il sottoscritto non meritano di avere poteri speciali sugli altri. Se qualcuno vuole ricompensarmi ulteriormente per il software o i libri che ho scritto, accetto volentieri un assegno – ma vi invito a non rinunciare alla vostra libertà a nome mio.

Questa è la prima versione mai pubblicata di questo saggio, e fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# La scienza deve mettere da parte il copyright

Dovrebbe essere evidente che lo scopo dell'editoria scientifica è la diffusione delle conoscenze scientifiche, e che le relative pubblicazioni esistono per facilitare un simile processo. Di conseguenza le norme che regolamentano tale attività editoriale dovrebbero assecondare il raggiungimento di quest'obiettivo.

Le regole attualmente in vigore, note come copyright, vennero stabilite all'epoca dell'invenzione della stampa, metodo intrinsecamente centralizzato per la copia a livello di massa. Nel settore della stampa, il copyright sugli articoli di queste pubblicazioni riguardava soltanto gli editori, imponendo loro l'ottenimento del permesso per la pubblicazione dei materiali, e i potenziali plagiaristi. Ciò consentì a quell'attività editoriale di operare e diffondere conoscenza, senza interferire con l'utile attività di ricercatori e studenti, sia in quanto autori o lettori dei testi. Si trattava di norme adeguate a quel sistema.

Tuttavia, la tecnologia moderna per l'editoria scientifica è il World Wide Web. Quali le norme che possono garantire al meglio la massima diffusione di materiale e conoscenze scientifiche sul Web? Gli articoli andrebbero distribuiti in formati non-proprietari, garantendone il libero accesso a tutti. E chiunque dovrebbe avere il diritto a crearne dei mirror, ovvero a ripubblicarli altrove in versione integrale con gli adeguati riconoscimenti.

Regole queste che andrebbero applicate sia a testi passati che futuri, quando venga distribuito in formato elettronico. Ma non esiste alcun bisogno reale di modificare l'attuale sistema di copyright relativo alle

pubblicazioni cartacee, poichè il problema non riguarda quel settore. Sembra purtroppo che non tutti siano d'accordo con l'evidente verità che ha aperto questo saggio. Numerosi editori di pubblicazioni scientifiche sembrano ritenere che lo scopo dell'editoria specializzata sia quello di consentire loro quell'attività in modo da incassare le quote di abbonamento da ricercatori e studenti. Un ragionamento meglio noto come "confondere il fine con il mezzo".

L'approccio di costoro è stato quello di impedire l'accesso perfino alla lettura del materiale scientifico a quanti possono e sono disposti a pagare per farlo. Si è ricorso alle leggi sul copyright, che rimangono in vigore nonostante l'inadeguatezza rispetto alle reti informatiche, come scusa per impedire ai ricercatori di scegliere nuove regole.

Nell'interesse della cooperazione scientifica e del futuro dell'umanità, dobbiamo rifiutare alla radice un simile approccio – non soltanto i sistemi di blocco realizzati su queste basi, ma anche le errate priorità a cui sono ispirati.

Talvolta questi editori sostengono che l'accesso online richiede l'impiego di costosi server di alta potenza, e che devono imporre delle tariffe onde pagare le relative spese. Questo "problema" è una conseguenza dell'analogia "soluzione". Offriamo a tutti la libertà di creare dei mirror, e saranno le biblioteche di ogni parte del mondo a occuparsi di tali mirror per far fronte alle richieste. Una soluzione decentralizzata che ridurrà le necessità dell'ampiezza di banda e garantirà la rapidità d'accesso, tutelando al contempo i materiali di ricerca contro perdite accidentali.

Secondo gli editori, inoltre, lo stipendio dei redattori interni richiede l'imposizione di tariffe per l'accesso ai materiali. Diamo per scontato il fatto che i redattori vadano remunerati. La spesa per la revisione di una comune ricerca varia tra l'uno e il tre per cento del costo necessario alla sua realizzazione. Una percentuale talmente ridotta non può giustificare l'ostruzione nell'utilizzo dei risultati delle ricerche.

Al contrario, le spese di revisione potrebbero essere recuperate, ad esempio, imponendo una tariffa per pagina a carico degli autori, i quali a loro volta verrebbero rimborsati dagli sponsor della ricerca. È probabile che costoro non sollevino obiezioni, visto che attualmente sostengono spese ben più sostanziose per via delle tariffe a copertura degli abbonamenti delle biblioteche universitarie alle varie pubblicazioni. Modificando il modello economico in modo che le spese di revisione siano a carico degli sponsor della ricerca, è possibile eliminare l'apparente bisogno di limitare la visione dei materiali on-line. L'autore occasionale non affiliato con alcuna istituzione o azienda, e privo del sostegno di uno sponsor, potrebbe essere esente dalle spese di revisione, i cui costi andrebbero aggiunti a quegli autori che operano all'interno delle istituzioni.

Un'ulteriore giustificazione per l'imposizione di quote per accedere alle pubblicazioni on-line concerne la conversione degli archivi cartacei in formato digitale. Occorre certamente portare a termine simili progetti, ma dovremmo trovare modalità alternative per sostenerne le spese, modalità che non prevedano simili restrizioni d'accesso. Il lavoro in se stesso non risulterà più difficoltoso, né produrrà la maggioranza delle spese. È controproducente riversare gli archivi in formato digitale per poi sprecarne i risultati limitandone l'accesso.

La Costituzione statunitense sostiene che il copyright esiste per “promuovere il progresso della scienza”. Quando è il copyright a impedire tale progresso, la scienza deve metterlo da parte.

Questo saggio è apparso per la prima volta nel 1991 sul sito <http://www.nature.com> nella sezione *Web Debates*. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Cos'è il copyleft?

Il copyleft [permesso d'autore] è un metodo generale per realizzare un programma di software libero e richiedere che anche tutte le versioni modificate e ampliate dello stesso rientrino sotto il software libero.

La maniera più semplice per rendere libero un programma è quella di farlo diventare di pubblico dominio, senza copyright [diritto d'autore]. Ciò consente a chiunque di condividere tale programma e i relativi perfezionamenti, se questa è l'intenzione dell'autore. Ma così facendo, qualcuno poco incline alla cooperazione potrebbe trasformarlo in software proprietario. Potrebbe apportarvi delle modifiche, poche o tante che siano, e distribuirne il risultato come software proprietario. Coloro che lo ricevono in questa versione modificata non hanno la stessa libertà riconosciuta loro dall'autore originale; è stato l'intermediario a strappargliela.

L'obiettivo del progetto GNU è quello di offrire a tutti gli utenti la libertà di ridistribuire e modificare il software GNU. Se l'intermediario potesse strappar via la libertà, potremmo vantare un gran numero di utenti, ma privati della libertà. Di conseguenza, anziché rendere il software GNU di pubblico dominio, lo trasformiamo in "copyleft".

Questo significa che chiunque ridistribuisca il software, con o senza modifiche, debba passare oltre anche la libertà di poterlo copiare e modificare ulteriormente. Il copyleft garantisce che ogni utente conservi queste libertà.

Il copyleft fornisce inoltre ad altri programmatori l'incentivo ad

aggiungere propri contributi al software libero. Importanti programmi liberi, quali il compilatore GNU C++, esistono soltanto grazie a tali incentivi.

Il copyleft aiuta altresì quei programmatori disposti a offrire contributi per migliorare il software libero a ottenerne il permesso. Spesso costoro lavorano per aziende o università che sarebbero disposte a quasi tutto pur di guadagnare qualcosa. Un programmatore potrebbe voler offrire alla comunità le proprie modifiche, ma il datore di lavoro vorrebbe invece inserirle all'interno di un prodotto di software proprietario.

Quando gli spieghiamo che è illegale distribuirne versioni migliorate se non come software libero, generalmente il datore di lavoro decide di diffonderle in quanto tali piuttosto che buttarle via.

Per trasformare un programma in copyleft, prima lo dichiariamo sotto copyright; poi aggiungiamo i termini di distribuzione, strumento legale onde garantire a chiunque il diritto all'utilizzo, alla modifica e alla redistribuzione del codice di quel programma o di qualsiasi altro da esso derivato, ma soltanto nel caso in cui i termini della distribuzione rimangano inalterati. Così il codice e le libertà diventano inseparabili a livello legale.

Gli sviluppatori di software proprietario ricorrono al copyright per rubare agli utenti la propria libertà; noi usiamo il copyright per tutelare quella libertà. Ecco perché abbiamo scelto il nome opposto, modificando "copyright" in "copyleft".

Il copyleft è un concetto generale; esistono svariate modalità per definirne i dettagli. Nel progetto GNU, i termini specifici della nostra distribuzione vengono indicati nella GNU General Public License (Licenza Pubblica Generica GNU), spesso abbreviata in GNU GPL. Al riguardo esiste l'apposita pagina che risponde alle domande più frequenti (FAQ, Frequently Asked Questions:

<http://www.gnu.org/licenses/gpl-faq.html>). È inoltre possibile informarsi sul perché la Free Software Foundation riceva dei progetti sotto copyright da vari collaboratori (<http://www.gnu.org/copyleft/why-assign.html>).

Una forma alternativa di copyleft, la GNU Lesser General Public License, nota con l'acronimo LGPL, viene applicata ad alcune librerie GNU, ma non a tutte. Inizialmente questa licenza era chiamata GNU Library GPL, ma ne abbiamo modificato il nome perché quello precedente incoraggiava gli sviluppatori a usarla con maggior frequenza di quanto avessero dovuto. La GNU Library GPL, è tuttora disponibile in formato HTML e testo, pur essendo stata superata dalla LGPL.

La GNU Free Documentation License, abbreviata in FDL (Licenza per Documentazione Libera GNU) è una forma di copyleft stilata per l'utilizzo in manuali, libri di testo o altri documenti onde garantire a chiunque l'effettiva libertà di copiare e ridistribuire tali materiali, con o senza modifiche, sia a livello commerciale che non-commerciale. La licenza appropriata è inclusa in numerosi manuali e in ogni distribuzione del codice sorgente GNU.

La GNU GPL è progettata in modo da poter essere facilmente applicata a ogni programma, qualora l'autore ne detenga il copyright. Per farlo non è necessario apportare modifiche a tale licenza, basta aggiungere al programma una nota che faccia corretto riferimento al testo della GNU GPL.

Per rendere copyleft un programma usando la GNU GPL oppure la GNU LGPL, occorre riferirsi alla pagina con le apposite istruzioni (<http://www.gnu.org/copyleft/gpl-howto.html>). È importante notare che, qualora si decida di fare uso della GPL, bisogna riportarne il testo per intero. Si tratta di un insieme integrale, di cui non è consentita la copia parziale. (Analogo discorso per la LGPL).

Il ricorso agli stessi termini di distribuzione per programmi diversi tra loro ne facilita la copia del codice. Poichè tutti i programmi seguono i medesimi termini di distribuzione, non occorre preoccuparsi se questi siano o meno compatibili. La LGPL comprende una nota che consente la modifica dei termini di distribuzione per aderire alla GPL normale, in modo da renderne possibile la copia del codice in un altro programma già coperto dalla GPL.

Per rendere copyleft un manuale tramite la GNU FDL si consulti la pagina delle relative istruzioni (<http://www.gnu.org/copyleft/fdl-howto.html>). Come nel caso della GNU GPL, occorre usare la licenza per intero; non sono ammesse copie parziali.

Originariamente scritto nel 1996. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Copyleft: idealismo pragmatico

Ogni decisione presa nella vita emerge dai valori e dagli obiettivi personali. Questi possono variare da individuo a individuo; la fama, il denaro, l'amore, la sopravvivenza, il divertimento e la libertà, sono soltanto alcuni degli obiettivi perseguiti da una brava persona. Quando l'obiettivo è quello di aiutare tanto gli altri quanto se stessi, lo si definisce *idealismo*.

La mia attività nel campo del software libero è motivata da uno scopo idealistico: diffondere libertà e collaborazione. Voglio stimolare la diffusione del software libero, sostituendo il software proprietario che vieta la cooperazione, per contribuire così al miglioramento della società.

Questa la motivazione centrale per cui la GNU General Public License – il copyleft. (Quest'ultimo è anche definito permesso d'autore, mentre il copyright è il diritto d'autore). Tutto il codice aggiunto a un programma coperto dalla GPL deve essere software libero, anche se incluso in un file a parte. Rendo disponibile il mio codice affinché venga utilizzato nel software libero, e non nel software proprietario, in modo da incoraggiare altri programmatori a fare altrettanto.

La mia posizione è che, se gli sviluppatori di software proprietario ricorrono al copyright per impedirci di condividere i programmi, noi che preferiamo cooperare possiamo usare il copyright per offrire a ulteriori collaboratori un vantaggio particolare: diamo loro il permesso di utilizzare il nostro codice.

Non tutti coloro che usano la GNU GPL puntano a un simile obiet-

tivo. Molti anni fa, a un amico venne chiesto di ridistribuire un programma già coperto da copyleft sotto termini non-copyleft, e la sua risposta fu più o meno questa:

«Talvolta mi occupo di software libero, altre volte di software proprietario – ma in quest’ultimo caso, mi aspetto di essere retribuito». Era disposto a spartire il proprio lavoro con una comunità che condivide il software, ma non vedeva alcun motivo di fare lo stesso con un’azienda i cui prodotti avrebbero escluso tale comunità. Pur perseguendo uno scopo diverso dal mio, riconobbe l’utilità della GNU GPL per il raggiungimento dei suoi obiettivi.

Per ottenere qualcosa al mondo, l’idealismo da solo non è sufficiente – occorre scegliere un metodo che ci consenta di raggiungere lo scopo prefisso. In altri termini, bisogna essere “pragmatici”. La GPL è pragmatica? Diamo un’occhiata ai suoi risultati:

Prendiamo il compilatore GNU C++. Perché esiste un compilatore C++ libero? Soltanto perché ciò viene stabilito dalla GNU GPL. GNU C++ è stato sviluppato da un consorzio industriale, la MCC, partendo dal compilatore GNU C.

Normalmente la MCC realizza prodotti quanto più proprietari possibile. Ma hanno distribuito il front end C++ come software libero, poichè secondo la GNU GPL questo era l’unico modo per poterlo distribuire. Il front end C++ comprendeva parecchi nuovi file, ma poichè erano stati progettati per essere collegati con GCC, anch’essi dovevano aderire alla GPL. Il beneficio per la nostra comunità è evidente.

Passiamo a GNU Objective C. Inizialmente NeXT (sistema operativo creato da Steve Jobs, successivamente acquistato dalla Apple) voleva farne un front end proprietario; proposero di distribuirlo come file .o, lasciando agli utenti la possibilità di collegarlo con il resto di GCC, ritenendo così di poter aggirare i requisiti della GPL.

Ma secondo il nostro avvocato, ciò non avrebbe potuto eludere tali requisiti e non era consentito farlo. E così distribuirono il front end Objective C come software libero.

Questi esempi si riferiscono a diversi anni fa, ma la GNU GPL continua a portarci sempre più software libero.

Molte delle librerie GNU rientrano sotto la GNU Library General Public License, ma non per tutte è così. Una di queste librerie coperta dalla GNU GPL ordinaria è Redline, la quale implementa l'editing a linea di comando. Una volta ho scoperto un programma non libero che prevedeva l'utilizzo di Redline, e dissi all'autore che si trattava di un uso non consentito. Egli avrebbe potuto eliminare dal programma soltanto le funzionalità dell'editing a linea di comando, ma in realtà decise di ridistribuirlo sotto la GPL. Ora è un programma di software libero.

Non di rado i programmatori che mettono a punto dei miglioramenti a GCC (oppure a Emacs, Bash, Linux, o qualsiasi altro programma coperto dalla GPL) lavorano presso qualche azienda o università. Quando costoro vogliono ridistribuire quelle migliorie alla comunità e vedere il proprio codice incluso nella versione del programma, il datore di lavoro potrebbe dire:

«Fermo lì – quel codice ci appartiene! Non vogliamo dividerlo con altri; abbiamo deciso di trasformare la tua versione migliorata in un prodotto di software proprietario».

È qui che arriva in soccorso la GNU GPL. Il programmatore chiarisce al datore di lavoro che un simile prodotto di software proprietario costituirebbe una violazione del copyright, e costui comprende di trovarsi davanti a due sole possibilità: distribuire il nuovo codice come software libero oppure non distribuirlo affatto. Quasi sempre il programmatore ottiene carta bianca, e il codice viene inserito nella versione successiva del programma.

La GNU GPL non è sempre accondiscendente. Dice “no” ad alcune delle cose che talvolta si vogliono fare. Secondo alcuni utenti, ciò sarebbe un elemento negativo – la GPL “esclude” degli sviluppatori di software proprietario che invece “occorre portare nella comunità del software libero”.

Ma non siamo noi a escluderli dalla nostra comunità; sono loro che scelgono di non entrarvi. La decisione di produrre software proprietario significa scegliere di starne fuori. Farne parte vuol dire unirsi e contribuire al lavoro collettivo; non possiamo “portarli nella comunità” se non vogliono unirsi a noi.

Quel che possiamo fare è offrire loro un incentivo a farne parte. La GNU GPL è progettata in modo da fornire loro un incentivo sulla base del software preesistente: «Se rendete libero il vostro software, potrete usare questo codice». Naturalmente ciò non basta per convincere tutti, ma talvolta funziona.

Lo sviluppo di software proprietario non porta benefici alla nostra comunità, ma non di rado quei programmatori ci chiedono di passar loro qualcosa. Gli utenti di software libero possono dare qualche soddisfazione all’ego personale di quanti sviluppano software libero – riconoscenza e gratitudine – ma la tentazione è molto forte quando un’azienda ti dice:

«Basta che tu ci consenta di includere il tuo pacchetto nel nostro programma di software proprietario, e questo verrà utilizzato da migliaia e migliaia di persone!».

La tentazione potrebbe essere davvero forte, ma a lungo termine è meglio per tutti riuscire a resistere. È più difficile riconoscere le lusinghe e le pressioni quando queste arrivano in maniera indiretta, tramite organizzazioni di software libero che hanno adottato politiche favorevoli al software proprietario. Ne offrono un esempio l’X Consortium (e il suo successore, l’Open Group): finanziati

da produttori di software proprietario, per un decennio hanno cercato di convincere i programmatori a non usare il copyleft. Ora che l'Open Group ha distribuito X11R6.4 come software non-libero, quelli tra noi che hanno resistito sono contenti di averlo fatto.

(Nel settembre 1998, diversi mesi dopo il rilascio di X11R6.4 con termini di distribuzione non liberi, l'Open Group ha fatto marcia indietro, decidendo di ri-rilasciarlo sotto la medesima licenza di software libero, priva del copyleft, usata per il precedente X11R6.3. Ringrazio l'Open Group, ma il tardivo ripensamento non invalida le nostre conclusioni sul fatto che fosse effettivamente possibile aggiungere quelle restrizioni).

A livello pragmatico, pensare agli obiettivi a più lungo termine rafforzerà la capacità di resistenza contro simili pressioni. Concentrando l'attenzione sulla libertà e sulla comunità che si può costruire rimanendo fermi sulle proprie posizioni, si rinsalda la volontà di farcela. “Battiti per qualcosa o soccomberai per un nonnulla”.

E se i cinici mettono in ridicolo la libertà e la comunità... se i “realisti più intransigenti” sostengono che l'unico ideale possibile è il profitto... basta ignorarli, e continuare a usare il copyleft.

Originariamente scritto nel 1998. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Il pericolo dei brevetti sul software

Credo siate a conoscenza del mio lavoro a sostegno del software libero. L'intervento odierno non riguarda questo tema, ma affronta la questione degli abusi legislativi tesi a trasformare lo sviluppo del software in un'attività pericolosa. Questo è ciò che accade quando le norme sui brevetti vengono applicate al campo del software. Il punto non è la brevettabilità del software. Una simile descrizione sarebbe decisamente errata ed equivoca, perché non si tratta di brevettare dei programmi singoli. Se così fosse, non farebbe alcuna differenza, sarebbe qualcosa di fundamentalmente innocuo. La questione riguarda invece la brevettabilità delle idee. Ciascun brevetto copre qualche idea. I brevetti sul software sono brevetti che coprono qualche idea sul software, idee che prevediamo di usare nello sviluppo del software. In tal senso ciò rappresenta un ostacolo pericoloso per lo sviluppo del software nella sua interezza.

Forse avrete sentito qualcuno usare un termine ingannevole, “proprietà intellettuale”. Come potete notare, questa definizione è basata su un pregiudizio: dà per scontato che, qualunque sia il tema in discussione, il modo di trattarlo è considerarlo una sorta di proprietà, mentre in realtà è soltanto una delle molte alternative disponibili. Il termine “proprietà intellettuale” si pone come pregiudiziale sulle questioni fondamentali di qualsiasi tematica ci si stia occupando. Ciò non porta a considerazioni chiare e aperte.

Esiste un ulteriore problema in quel termine, il quale non ha nulla a che fare con la promozione delle opinioni personali: è d'intralcio nella comprensione perfino dei fatti. L'espressione “proprietà intel-

lettuale” viene usata come una sorta di panacea generale: raggruppa insieme aree del tutto disparate del corpo legislativo quali il copyright (il diritto d’autore) e i brevetti, ambiti completamente diversi tra loro che differiscono in ogni dettaglio. Nel mucchio finiscono anche i marchi registrati, qualcosa di ulteriormente diverso, e altri elementi in cui ci s’imbatte più di rado. Nessuno di questi settori ha nulla in comune con gli altri. Storicamente hanno origini completamente distinte; le rispettive legislazioni furono progettate in maniera indipendente; interessano ambiti diversi della vita e delle comuni attività. Le questioni di politica pubblica che sollevano non presentano alcuna relazione tra loro, di modo che cercando di affrontarli come un unico insieme è garantito il raggiungimento di conclusioni folli. È letteralmente impossibile avere un’opinione motivata e intelligente sulla “proprietà intellettuale”. Perciò, se si vuole considerare la questione con chiarezza, evitiamo di fare d’ogni erba un fascio. Meglio affrontare il copyright di per sé, e poi occuparsi dei brevetti. Impariamo a conoscere le norme sul copyright, e separatamente quelle sui brevetti.

Queste alcune delle maggiori differenze esistenti tra copyright e brevetti:

- Il copyright concerne i dettagli dell’espressione di un’opera, ma non copre alcuna idea. I brevetti riguardano soltanto le idee e il loro utilizzo.
- Il copyright è automatico. I brevetti vengono concessi dal relativo ufficio in risposta a un’apposita richiesta.
- I brevetti sono molto onerosi. In realtà le spese degli avvocati per la stesura della richiesta sono perfino più esose della domanda stessa. Normalmente occorrono alcuni anni prima che la richiesta venga presa in considerazione, anche se i vari uffici brevetti lavorano in maniera estremamente lenta nell’esame delle domande.

- La durata del copyright è tremendamente lunga. In alcuni casi si arriva anche a 150 anni. I brevetti durano 20 anni, periodo breve rispetto alla vita umana ma comunque eccessivo per i ritmi di un settore come quello del software. Basti pensare a 20 anni fa, quando il PC era qualcosa di nuovo. Immaginiamo di dover essere costretti a sviluppare software utilizzando soltanto i concetti conosciuti nel 1982.
- Il copyright copre unicamente la copia. Se qualcuno scrive un romanzo che si scopre essere identico parola per parola a *Via col vento*, potendo al contempo dimostrare di non averlo mai visto, ciò rappresenterebbe un'ottima difesa contro ogni accusa di infrazione al copyright.
- Un brevetto è un monopolio assoluto sull'utilizzo di un'idea. Anche potendo dimostrare di aver avuto quell'idea per conto proprio, ciò sarebbe del tutto irrilevante se quell'idea è stata già brevettata da qualcun altro.

Spero possiate dimenticarvi del copyright per il resto del mio intervento, perché parlerò invece dei brevetti, e le due questioni non dovrebbero mai essere messe insieme - unica possibilità per comprendere con chiarezza le rispettive questioni legali. Pensiamo a cosa potrebbe accadere nella comprensione della chimica pratica (o dell'arte culinaria) se dovessimo confondere l'acqua con l'etanolo. Quando si sente qualcuno parlare del sistema dei brevetti, normalmente questo viene descritto dal punto di vista di chi spera di ottenere un brevetto - le procedure che bisognerebbe eventualmente seguire per richiederlo, la sensazione che si proverebbe nel camminare per strada avendone uno in tasca, in modo da tirarlo fuori ogni tanto e sbatterlo in faccia a qualcuno dicendo: «Dammi tutti i soldi che hai!».

C'è una ragione dietro questi pregiudizi, perché la maggior parte di

quanti ci descrivono il sistema dei brevetti vanta qualche tipo di interesse in tale sistema, perciò ce ne dipingono i tratti piacevoli. Esiste un'ulteriore motivazione: il sistema dei brevetti somiglia parecchio a una lotteria, perché soltanto una minima parte dei brevetti porta effettivamente qualche beneficio ai rispettivi possessori. Non casualmente una volta la rivista *The Economist* lo ha paragonato a una "lotteria spreca-tempo". Se avete dato un'occhiata alle inserzioni pubblicitarie delle lotterie, avrete notato come queste ci spingano sempre a pensare alla possibilità di vincere. Non invitano certo a considerare l'eventualità di perdere, pur essendo questa la probabilità più concreta. Lo stesso vale per il sistema dei brevetti: vengono sempre presentati come un invito a considerarci tra i vincitori.

Per controbilanciare questa serie di pregiudizi, mi accingo a descrivere il sistema dei brevetti dal punto di vista delle vittime – ovvero, dal punto di vista di qualcuno che vuole sviluppare del software, ma è costretto a convivere con un sistema di brevetti sul software che potrebbe risultare in una denuncia.

Qual'è dunque la prima cosa da fare dopo aver avuto un'idea sul tipo di programma che ci si appresta a scrivere? Tanto per cominciare, dovendo aver a che fare con il sistema dei brevetti, si potrebbe cercare di scoprire quali siano i brevetti che coprono il futuro programma. Compito impossibile. Il motivo è che alcune delle domande pendenti sono segrete. Possono essere pubblicate soltanto dopo un certo tempo dalla presentazione, qualcosa tipo 18 mesi. Si tratta tuttavia di un periodo più che sufficiente per scrivere un programma, e finanche per distribuirlo, senza poter conoscere se sia già coperto da brevetto o meno, e rischiare di conseguenza la denuncia.

Non è una faccenda puramente accademica. Nel 1984 venne realiz-

zato compress, un programma per la compressione dei dati. All'epoca non esistevano brevetti sull'algoritmo di compressione LZW usato in quel programma. Più tardi, nel 1985, l'ufficio statunitense diffuse un brevetto su tale algoritmo, e nel corso degli anni successivi coloro che ne curavano la distribuzione iniziarono a ricevere minacce legali. Era impossibile per l'autore dell'algoritmo di compressione prevedere che potesse subire una denuncia. Non aveva fatto altro che usare un'idea trovata in una pubblicazione, proprio come era sempre successo tra i programmatori. Non aveva capito che non era più possibile usare liberamente un'idea trovata in qualche pubblicazione. Dimentichiamo questo problema. I brevetti approvati vengono pubblicati dall'apposito ufficio, così da poterne reperire il lungo elenco e vedere con esattezza cosa dicono. Ovviamente, in realtà è impossibile leggere la lista per intero, perché ne comprende una quantità enorme. Negli Stati Uniti esistono centinaia di migliaia di brevetti sul software. Non c'è alcun modo di tener traccia di tutte le aree coperte. L'unica possibilità è provare a cercare quelli più rilevanti.

Qualcuno sostiene che dovrebbe trattarsi di un compito semplice nell'attuale epoca informatica. Si può ricorrere a ricerche per parole chiave e così via, ma ciò funziona soltanto fino a un certo punto. Si troveranno alcuni brevetti riguardanti l'area d'interesse, ma non necessariamente tutti.

Ad esempio, c'era un brevetto sul software (che oggi potrebbe essere estinto) relativo alle operazioni di ricalcolo secondo l'ordine naturale per i fogli di calcolo elettronici. In pratica ciò significa che, facendo dipendere determinate celle da altre precedenti, si procede sempre al ricalcolo di tutti gli elementi inseriti dopo quelli da cui dipendevano, in modo che dopo ogni operazione il totale risulta sempre aggiornato. Nei primi fogli di calcolo elettronici si proce-

deva dall'alto verso il basso, e facendo dipendere una cella da quella sottostante, impostando al contempo una serie di simili passaggi, occorreva ricalcolare il totale svariate volte per far propagare il nuovo valore verso l'alto. (Si presupponeva che ogni elemento dipendesse dalla cella sovrastante).

Allora qualcuno ha pensato, perché non rifare i calcoli in modo che ogni valore venga ricalcolato immediatamente dopo quello da cui dipende? Questo algoritmo è conosciuto col nome di 'classificazione topologica'. Il primo riferimento che sono riuscito a trovare è datato 1963. Il brevetto copriva diverse dozzine di modalità in cui si poteva implementare la classificazione topologica.

Non era tuttavia possibile reperire tale brevetto cercando con "fogli di calcolo elettronici". Né lo si trovava provando con "ordine naturale" oppure "classificazione topologica". La spiegazione non comprendeva nessuno di questi termini. Veniva in realtà descritto come un metodo per "compilare formule in codici oggetto". Quando lo vidi per la prima volta non credevo fosse quello giusto.

Supponiamo di trovarci davanti a un elenco di brevetti e di volerli rendere conto di quel che sia consentito fare. Quando si prova a studiarne le descrizioni, si scopre che sono molto difficili da comprendere, poiché sono scritte in un tortuoso linguaggio legale il cui significato è tutt'altro che facile da capire. Spesso quanto dice l'ufficio brevetti ha un significato diverso da quello apparente.

Negli anni '80 il governo australiano ha condotto una ricerca sul sistema dei brevetti. La conclusione fu che, al di là delle pressioni internazionali, non c'era alcun motivo per l'esistenza di un tale sistema – non portava alcun giovamento al pubblico – e ne raccomandava l'abolizione, se non fosse per le pressioni internazionali. Lo studio riportava che gli ingegneri non cercavano neppure di leggere i brevetti per imparare qualcosa, consideratane la difficoltà di

comprensione. Veniva citata l'opinione di un ingegnere: "Non riesco a riconoscere le mie stesse invenzioni in linguaggio brevettese". Questo non è uno scenario teorico. Verso il 1990 un programmatore di nome Paul Heckel denunciò la Apple, sostenendo che Hypercard infrangeva un paio di suoi brevetti. Quando lo vide per la prima volta, gli sembrò che Hypercard non avesse nulla a che fare con quei brevetti, con le sue "invenzioni". Non pareva simile a queste. Ma quando l'avvocato gli fece notare che quei brevetti potevano essere interpretati a coprire una parte di Hypercard, decise di attaccare la Apple. Nel corso di un mio successivo intervento a Stanford menzionai quella circostanza, con Heckel presente tra il pubblico. Mi interruppe per ribattere: "Non è vero, è solo che non compresi la portata della tutela legale!". Io replicai: "Sì, proprio quello che intendevo dire".

Così, in pratica occorre spendere un sacco di tempo a parlare con gli avvocati per capire quel che i brevetti proibiscono di fare. Alla fine se ne usciranno con qualcosa tipo: "Se fai qualcosa in quest'area, sei sicuro di perdere; se intervieni in quest'altra area (Stallman fa dei gesti circolari con le mani), in sostanza si rischia di perdere; e se vuoi essere davvero al sicuro, meglio star lontano da quest'area (facendo gesti circolari più ampi). E tieni comunque conto che qualsiasi denuncia comporta qualche elemento di rischio sostanziale".

Ora che avete un terreno prevedibile su cui basare i vostri affari(!), cosa pensate di fare? Bé, si può scegliere fra tre diverse eventualità, ciascuna delle quali è applicabile in determinate circostanze. Ecco:

- 1) evitare il brevetto;
- 2) ottenere la licenza per il brevetto;
- 3) ribaltare il brevetto in tribunale.

Consentitemi di illustrare questi tre scenari per capire cosa li renda praticabili o meno.

## Evitare il brevetto

“Evitare il brevetto” vuol dire non usare l’idea già coperta da qualche brevetto. Posizione semplice o difficile da seguire, dipende dal tipo di idea in oggetto.

In alcuni casi, soltanto una funzione risulta brevettata. In tal caso si può eludere il brevetto evitando di implementarla. Il punto sta nell’importanza di tale funzione. In alcune situazioni, se ne può fare a meno. Tempo fa gli utenti dell’elaboratore testi XyWrite vennero notificati di un downgrade (declassamento) del programma. Ne fu rimossa una funzione che consentiva la predefinita delle abbreviazioni. Ovvero, quando si digitava un’abbreviazione seguita da un carattere d’interpunzione, questa sarebbe stata immediatamente sostituita dalla relativa espansione. In tal modo per alcune frasi lunghe era possibile definire l’abbreviazione, digitare soltanto quest’ultima e l’intera frase sarebbe apparsa nel documento. Gli sviluppatori mi scrissero riguardo questa opzione perché sapevano che l’editor Emacs presentava una funzione analoga. Infatti ne faceva parte fin dagli anni ‘70. Fu interessante perché ciò dimostrò che in vita mia avevo avuto almeno un’idea meritevole di essere brevettata. Posso affermarlo con certezza, perché poco tempo dopo è proprio quel che fece qualcun altro!

In realtà gli sviluppatori considerarono tutte e tre le strategie. Prima tentarono di negoziare con il detentore del brevetto, il quale si rivelò trattare in cattiva fede. Poi analizzarono le probabilità concrete di poter ribaltare il brevetto in tribunale. Alla fine decisero che la cosa da fare fosse l’eliminazione di quella funzione. È possibile farne a meno. Se l’elaboratore testi difetta di quest’unica opzione, forse gli utenti continueranno a utilizzarlo ugualmente. Ma se le mancanze cominciano ad accumularsi, alla fine si avrà un programma non troppo soddisfacente, ed è probabile venga ignorato.

In questo caso si trattava di un brevetto limitato su una funzione assai specifica. Come la mettiamo con il brevetto in possesso di British Telecom sugli hyperlink [i link del world wide web] accoppiato con l'accesso tramite la comune chiamata telefonica? Il ricorso agli hyperlink è assolutamente essenziale nell'odierno uso del computer, al pari della connessione via telefono. Come faremmo senza tale opzione? Per chiarezza, questa non può neppure considerarsi una singola funzione, bensì la combinazione di due opzioni arbitrariamente interconnesse tra loro. Qualcosa di analogo al possesso di un brevetto su divano e televisione in una stessa stanza.

Talvolta l'idea brevettata appare talmente vasta e basilare che praticamente finisce per coprire un intero settore. È ad esempio il caso della crittazione a chiave pubblica, il cui brevetto statunitense è scaduto nel 1997. Fino ad allora fu quel brevetto a impedire per la maggior parte il ricorso alla crittazione a chiave pubblica negli Stati Uniti. Un certo numero di programmi in corso di lavorazione vennero bloccati – non furono mai realmente disponibili perché i detentori del brevetto presero a minacciarne gli autori. Poi ne uscì fuori uno, PGP, che inizialmente venne distribuito come software libero. In questo caso sembra che i possessori del brevetto lasciarono passare del tempo prima di minacciare la denuncia, e a quel punto si resero conto della cattiva pubblicità che avrebbero potuto subire. Così imposero delle restrizioni, rendendolo disponibile soltanto per usi non-commerciali, onde impedirne l'eccessiva diffusione. In tal modo, per un decennio e oltre l'uso della crittazione a chiave pubblica venne fortemente limitato. Non c'era modo di superare quel brevetto. Era impossibile inventarsi qualcos'altro per scrivere programmi di crittazione a chiave pubblica.

Altre volte viene brevettato un algoritmo specifico. Esiste ad esempio un brevetto su una versione ottimizzata del Fast Fourier Tran-

sform, grazie al quale quest'ultimo gira a velocità doppia. Per evitarlo basta usarne una comune versione, anche se questa parte del programma impiegherà il doppio del tempo. Forse non è poi una funzione così importante, forse ciò riguarda una parte minima del tempo totale in cui gira il programma. Anche se è due volte più lento, può darsi che nessuno se ne accorga. Oppure, al contrario, il programma non si lancerebbe affatto perché richiede il doppio del tempo reale per operare come previsto. Gli effetti possono variare. In qualche caso si può cercare un algoritmo migliore. Ciò può tornare utile o meno. Non potendo usare 'compress' all'interno del progetto GNU, iniziammo a cercare un algoritmo alternativo adatto alla compressione dati [compress è una utility per i sistemi Unix con algoritmo brevettato, per cui non poteva essere utilizzata nel progetto GNU: il brevetto avrebbe posto una limitazione alla redistribuzione, rendendo impossibile distribuire il software con licenza GPL].

Qualcuno ci informò di averne uno disponibile; aveva scritto un programma e decise di offrircelo come contributo. Eravamo sul punto di distribuirlo. Per pura coincidenza mi capitò di vedere una copia del *New York Times* che casualmente aveva la rubrica settimanale dedicata ai brevetti. (Non sfogliavo quel quotidiano più di una volta ogni paio di mesi). Così iniziai a dargli un'occhiata e leggo che qualcuno aveva ottenuto un brevetto per "aver inventato un nuovo metodo per la compressione dei dati". Decido che è meglio capire come stanno le cose. Ne prendo una copia e scopro che tale brevetto copriva proprio il programma che ci apprestavamo a distribuire nel giro di una settimana. Il programma morì prima ancora di esser nato.

In seguito trovammo un altro algoritmo non coperto da brevetto. Divenne il programma gzip, oggi l'efficace standard de facto per la

compressione dati. Come algoritmo per essere usato in simili programmi, risultò perfetto. Chiunque volesse ricorrere alla compressione dei dati poteva usare gzip invece di compress.

Lo stesso algoritmo di compressione già brevettato LZW veniva impiegato anche in formati per immagini, tra cui GIF. Ma in questo caso, poichè la gente non voleva semplicemente comprimere dei dati bensì avere un'immagine che fosse possibile visualizzare con il proprio software, si rivelò estremamente difficile trovare un algoritmo diverso. Non ci siamo ancora riusciti dopo 10 anni! Sì, gli utenti ricorrevano all'algoritmo 'gzip' con cui definire un altro formato per l'immagine, una volta piovute minacce di possibili denunce per l'uso di file GIF. Quando iniziammo a dire in giro di smetterla di usare quei file GIF per passare all'altro formato, la replica fu: "Non possiamo cambiare, il browser non supporta ancora il nuovo formato". Gli sviluppatori di browser ribatterono: "Non abbiamo alcuna fretta, dopo tutto nessuno usa quel formato".

In effetti la società ha dimostrato così tanta inerzia nell'utilizzo del formato GIF che non siamo riusciti a convincere la gente a cambiare. In pratica il continuo ricorso della comunità a tale formato forza tuttora i vari siti a farne uso, con il risultato che si rivelano vulnerabili a possibili minacce legali.

Anzi, la situazione è ben più strana. In realtà sono due i brevetti che coprono l'algoritmo LZW. L'ufficio brevetti non si è reso conto che stava assegnando due brevetti su una medesima idea; non erano riusciti a esaminare adeguatamente le richieste. Ciò però poggia su una buona ragione: occorre parecchio tempo per studiare con attenzione i due brevetti prima di rendersi conto che coprono veramente la stessa cosa.

Se si fosse trattato di due brevetti su dei processi chimici, sarebbe stato assai più semplice rendersene conto. Bastava identificare le

sostanze impiegate, gli ingredienti e i risultati, quali le azioni concrete intraprese. Prescindendo dal modo in cui ciò veniva descritto, si sarebbe visto di cosa si trattava e ci si sarebbe accorti della somiglianza. Se un'azione è puramente matematica, è possibile descriverla in molti modi anche assai diversi tra loro. Non appaiono simili neppure a livello superficiale. Bisogna comprenderli fino in fondo prima di accorgersi che stanno descrivendo qualcosa d'identico. L'ufficio brevetti non ha abbastanza tempo. Alcuni anni fa l'ufficio brevetti degli Stati Uniti dedicava mediamente 17 ore a ciascuna richiesta presentata. Un periodo di tempo insufficiente per analizzarle con attenzione, ovvio quindi che si possano commettere errori come questo. Lo stesso è accaduto al programma menzionato sopra, quello morto prima ancora di nascere. Anche quell'algoritmo è coperto da due brevetti, entrambi rilasciati negli Stati Uniti; sembra si tratti di una circostanza nient'affatto insolita.

Evitare i brevetti può quindi essere semplice, oppure impossibile. Se è semplice, può darsi che renda inutile il programma – dipende dalla situazione specifica.

Vorrei puntualizzare un'altra questione. Talvolta un'azienda o un consorzio riesce a imporre un formato o un protocollo come standard de facto. Nel caso tale formato o protocollo venga poi brevettato, è un vero e proprio disastro. Esistono perfino degli standard ufficiali soggetti alle limitazioni dei brevetti. Nel settembre del 2001 ci fu una grande sollevazione politica quando il World Wide Web Consortium propose di iniziare ad adottare degli standard coperti da brevetti. La comunità si oppose, costringendoli a ripensarci. Il consorzio fece marcia indietro, ribadendo che qualsiasi brevetto doveva essere liberamente applicabile da chiunque e che gli standard dovevano essere liberi in modo che tutti potessero implementarli. Si trattò di una vittoria interessante. Credo che quella fu la

prima volta in cui un'organizzazione sugli standard prese quel tipo di decisione. È normale per tali organizzazioni voler inserire all'interno degli standard qualche elemento coperto da brevetto, impedendo agli utenti di procedere liberamente all'implementazione. Bisogna far pressione su entità organizzative analoghe per costringerle a modificare quelle norme.

### **Ottenere la licenza per il brevetto**

La seconda possibilità, oltre quella di evitare il brevetto, consiste nell'ottenerne la licenza. Non è detto ciò possa essere necessariamente un'opzione fattibile. Chi detiene il brevetto non deve offrirvi alcuna licenza; non è obbligato a farlo. Dieci anni fa, la League for Programming Freedom ricevette una richiesta d'aiuto da parte di qualcuno la cui attività familiare riguardava la costruzione di macchine per i giochi d'azzardo nei casinò, e già allora usavano i computer. Aveva ricevuto la lettera di un'altra azienda che minacciava: "Quel brevetto l'abbiamo noi. Non ti è consentito fare quelle cose. Smettila!". Decisi di dare un'occhiata a quel brevetto. Copriva una situazione in cui un certo numero di computer venivano collegati in rete in modo che ciascuna macchina fosse in grado di gestire più giochi diversi e l'utente potesse giocare simultaneamente.

Si scopre così che l'ufficio brevetti ritiene davvero brillante la capacità di fare più di una cosa in contemporanea. Non si rendono conto che in campo informatico ciò rappresenta la maniera più ovvia per generalizzare qualsiasi cosa. Lo hai fatto una volta, perciò adesso puoi rifarlo quante volte vuoi, si può creare una subroutine [una funzione del programma]. Credono che se riesci a fare qualcosa più di una volta, ciò in qualche modo significa che sei brillante e che nessuno è in grado di tenerti testa, hai il diritto di dare ordini agli altri come ti pare e piace.

Comunque sia, al tipo in questione non venne offerta alcuna licenza. Fu costretto a smettere. Non poteva neppure permettersi le spese legali per il tribunale. Direi che quel particolare brevetto copri-va un'idea piuttosto ovvia. È possibile che il giudice si fosse dichiarato d'accordo, ma non potremo mai saperlo perché non c'erano i soldi per avviare il procedimento.

Tuttavia, parecchi possessori di brevetti sono soliti offrire le relative licenze. Ma spesso chiedono cifre salate. L'azienda in possesso del brevetto per il ricalcolo secondo l'ordine naturale pretendeva il 5 per cento delle entrate lorde di ogni foglio di calcolo elettronico venduto negli Stati Uniti. Qualcuno mi riferì che si trattava del prezzo ridotto precedente l'eventuale denuncia – se fossero stati costretti ad andare veramente in tribunale e avessero vinto, avrebbero preteso di più.

Può darsi che su uno specifico brevetto ci si possa permettere quel 5 per cento per la licenza, ma cosa succede quando occorrono le licenze su 20 brevetti diversi per realizzare un certo programma? A quel punto tutti i ricavi servono a pagare le licenze. Cosa succederebbe se fossero necessarie le licenze su 21 brevetti? Persone che operano nel settore mi hanno spiegato che a livello pratico due o tre di tali licenze porterebbero al fallimento di qualsiasi attività.

Esiste una situazione in cui ottenere la licenza è un'ottima soluzione. Ovvero nel caso di una mega-corporation multinazionale. Dato che queste aziende possiedono una gran quantità di brevetti, possono offrirsi le licenze a vicenda. In tal modo evitano gran parte del danno insito nel sistema dei brevetti e usufruiscono soltanto degli aspetti positivi.

Tempo fa l'IBM pubblicò sulla rivista *Think* – credo fosse il numero 5 del 1990 – un articolo sul pacchetto di brevetti aziendale, dove si illustravano i due tipi di vantaggi derivanti alla stessa IBM dal

possesto di 9.000 brevetti statunitensi. (Credo che oggi tale cifra sia più ampia). Si trattava, primo, di incassarne le quote sui diritti, e, secondo, di ottenere “accesso ai brevetti altrui”. Spiegavano come quest’ultimo beneficio fosse notevolmente maggiore del primo. I vantaggi derivanti all’IBM dalla possibilità di utilizzare le idee brevettate da altri equivaleva a dieci volte il beneficio diretto ottenuto dall’offerta di licenze sui propri brevetti.

Cosa significa veramente tutto ciò? Quali vantaggi ricava l’IBM da un simile “accesso ai brevetti altrui”? Si tratta in pratica del beneficio di essere esenti dai problemi provocati dal sistema dei brevetti. Un sistema analogo a una lotteria: qualsiasi brevetto può finire in un nonnulla, o rivelarsi una fortuna per alcuni detentori, oppure un disastro per chiunque altro. Ma consideratene le ampie proporzioni, per l’IBM lo scenario risulta vantaggioso. Un’azienda simile è in grado di valutare sia i danni sia i benefici di quel sistema. In questo caso, i guai causati da un tale sistema avrebbero superato di dieci volte gli aspetti positivi.

Ho detto “avrebbero” perché, grazie allo scambio vicendevole delle licenze, l’IBM può evitare ogni problema. Questi esistono soltanto a livello potenziale, non si concretizzeranno mai per una tale azienda. Eppure quest’ultima, calcolando i benefici derivanti dalla possibilità di evitarli, ne stima in dieci volte il valore del denaro raccolto dai brevetti di cui è titolare.

Questo fenomeno del trasferimento reciproco delle licenze serve a confutare un mito comune, quello del “genio morto di fame”, il mito secondo cui i brevetti servono a “tutelare” il “piccolo inventore”. (Si tratta di termini di propaganda. Non andrebbero usati). Questo lo scenario che ci troviamo di fronte: supponiamo che qualcuno abbia in mente un progetto “brillante”. Supponiamo che abbia trascorso “anni in soffitta morendo di fame” nella stesura di

un progetto nuovo e meraviglioso di qualcosa, e ora vuole passare a produrlo. Non è forse un'ingiustizia che qualche grande azienda decida di fargli concorrenza, di rubargli il mercato e farlo "morire di fame"?

Devo sottolineare che quanti lavorano nel settore dell'alta tecnologia generalmente non operano in proprio, che le idee non prendono forma nel vuoto – sono basate su quelle altrui – e che oggi giorno vantano ottime probabilità di ottenere un buon impiego qualora ne avessero bisogno. Perciò un tale scenario – il fatto che un'idea brillante sia scaturita da qualcuno che lavora in solitudine – è inverosimile, così come impensabile è l'eventualità che sia in pericolo di morire di fame.

È invece plausibile che qualcuno possa avere una buona idea che, insieme ad altre 100 o 200, sia alla base della realizzazione di un qualche tipo di prodotto, e che le grandi aziende vogliano fargli concorrenza. Vediamo perciò cosa potrebbe accadere nel caso costui tenti di usare un brevetto per bloccarle. Eccolo dire all'IBM: "No, non puoi competere con me, quel brevetto è mio". Al che l'IBM replica: "Vediamo un po' il tuo prodotto. Noi abbiamo questo brevetto, e quest'altro, quest'altro, quest'altro, quest'altro, e quest'altro ancora, rispetto ai quali il tuo prodotto commette delle infrazioni. Se credi di poter controbattere a tutti questi brevetti in tribunale, sicuramente ne troveremo degli altri. Perché invece non ci cediamo reciprocamente le licenze?". E allora al brillante inventore non resta che cedere: "Va bene, scambiamoci pure le licenze". Così può rimettersi al lavoro e portare a termine quel meraviglioso progetto. Ma lo stesso fa l'IBM, la quale ottiene "l'accesso" al suo brevetto e anche il diritto a fargli concorrenza, il che significa che tale brevetto non lo ha "tutelato" affatto. Non è questo l'obiettivo del sistema dei brevetti. Per la gran parte, le mega-corporation evitano i pericoli di tale siste-

ma; ne sperimentano principalmente i lati positivi. Questo il motivo per cui vogliono avere i brevetti sul software: saranno loro a trarne vantaggio. Ma ciò non funziona per il piccolo inventore o per chi lavora in una piccola struttura. Possono provarci, ma il problema è che un'azienda di proporzioni limitate non arriverà mai a possedere una quantità adeguata di brevetti per riuscirci (cioè, costringere gli altri allo scambio reciproco delle licenze).

Ciascun brevetto punta in una certa area. Così se una piccola azienda possiede dei brevetti relativi a determinati settori, e laggiù (Stallman indica in una direzione diversa) c'è qualcuno che gliene punta uno contro e vuole tutti i soldi, alla piccola azienda non resta che arrendersi. L'IBM può permetterselo, perché con 9000 brevetti copre ogni settore; a prescindere dall'area in cui si operi, è probabile esista già un brevetto dell'IBM. Questa può così costringere quasi sempre gli altri allo scambio reciproco delle licenze. Le piccole aziende possono riuscirci invece solo occasionalmente. Sostengono di volere i brevetti a scopo difensivo, ma non riescono mai ad accumularne abbastanza da poterlo fare realmente.

Esistono tuttavia dei casi in cui neppure l'IBM può costringere qualcuno a scambiare delle licenze a vicenda. Ovvero quando l'unica attività di un'impresa è quella di prendere un brevetto e spremere soldi dagli altri. Esattamente ciò che faceva l'azienda detentrica del brevetto per il ricalcolo secondo l'ordine naturale. L'unica sua pratica consisteva nel minacciare gli altri di denuncia e incassare le quote di chi stava effettivamente sviluppando qualche prodotto.

Non esistono brevetti sulle procedure legali. Credo che gli avvocati comprendano bene gli affanni di avere personalmente a che fare con il sistema dei brevetti. Il risultato è che diventa impossibile ottenere brevetti che possano costringere questo tipo di aziende allo

scambio reciproco. Così vanno in giro a spremere soldi agli altri. Ma credo che entità quali l'IBM considerino ciò parte del prezzo insito nell'attività commerciale, e si siano adattate di conseguenza. Questo dunque il quadro sulle possibilità di ottenere la licenza di un brevetto, cosa realizzabile o meno, e di cui è possibile o meno sostenere il peso economico – il che ci porta alla terza strategia.

### **Ribaltare il brevetto in tribunale**

Normalmente, qualsiasi cosa venga brevettata deve risultare nuova, utile e non ovvia. (Questa la terminologia usata negli Stati Uniti; credo che in altri paesi il linguaggio sia pressoché analogo). Naturalmente, quando entra in ballo l'ufficio brevetti inizia a dare la propria interpretazione di “nuovo” e “originale”. Si scopre così che “nuovo” equivale a “non presente nel nostro archivio” e “non ovvia” tende a significare “non ovvia per qualcuno con un quoziente d'intelligenza di 50” [per una persona comune è intorno ai 140].

Secondo qualcuno che studia la maggior parte dei brevetti sul software assegnati negli Stati Uniti – almeno, una volta era solito farlo, non so se riesce ancora a starci dietro – il 90 per cento non avrebbe superato “l'esame di Crystal City”, per intendere che nel caso gli addetti all'ufficio brevetti avessero deciso di passare in edicola e procurarsi qualche rivista d'informatica, avrebbero scoperto come quelle idee fossero già note.

L'ufficio brevetti prende decisioni così chiaramente folli che non occorre neppure essere aggiornati sulle ultima novità per rendersi conto della loro assurdità. Ciò non si limita soltanto al software. Una volta ho visto il famoso brevetto sul topo di Harvard, ottenuto dopo che i ricercatori locali avevano modificato geneticamente il topo iniettandogli il gene portatore del cancro. Tale gene era già conosciuto, e venne inserito ricorrendo a tecniche note all'interno

di una catena preesistente di cellule di topo. Il brevetto concesso loro copriva l'inserimento di qualsiasi gene portatore di cancro in qualunque mammifero usando un metodo qualsiasi. Non occorre saper nulla di ingegneria genetica per rendersi conto di quanto ciò sia ridicolo. Mi si dice che questo "eccesso di rivendicazione" sia pratica comune, e che talvolta l'ufficio brevetti statunitense invita i richiedenti a estendere ulteriormente il campo coperto dal brevetto. Praticamente si finisce per coprire il massimo possibile fino a quando non ci si accorge di essere vicini a un'area certamente già occupata da opere precedenti. Si cerca di arraffare quanto più territorio possibile dello spazio mentale a disposizione.

Quando i programmatori considerano molti brevetti sul software, non possono far a meno di osservare: "quest'idea è ridicolmente ovvia!". I burocrati dei brevetti tirano fuori ogni tipo di scuse pur di giustificare la loro ignoranza del pensiero dei programmatori. Replicano così: "Bisogna però considerarla rispetto a come stavano le cose dieci o venti anni fa". Per poi scoprire che se portate alle estreme conseguenze, simili posizioni diventano controproducenti. Qualsiasi cosa può apparire originale quando se ne scompongono i pezzi, quando la si analizza abbastanza a fondo. Semplicemente svanisce ogni standard di ovvietà, o quantomeno si perde la capacità di giustificare qualunque standard di ovvietà o non ovvietà. A quel punto, naturalmente, si finisce per descrivere tutti coloro che possiedono un brevetto come dei brillanti inventori; di conseguenza, non possiamo mettere in discussione il loro diritto a imporci cosa fare.

Se si decide di andare in tribunale, è probabile che i giudici mostrino maggiore attenzione alla questione della ovvietà o meno. Ma il problema è che per arrivarci bisogna spendere milioni di dollari. Ho sentito parlare di un caso, l'accusato ricordo era la Qualcomm,

in cui credo la sentenza finale fu di 13 milioni di dollari, la maggior parte dei quali servì a coprire l'onorario degli avvocati di entrambe le parti. Rimasero un paio di milioni di dollari per il querelante (fu la Qualcomm a perdere la causa).

In un contesto più ampio, la questione della validità o meno di un brevetto dipende dalle circostanze storiche. Meglio, da una gran quantità di indizi storici, tipo cosa e quando venne pubblicato, il materiale che si riesce a recuperare, quello non andato perduto, le date precise e così via. È la presenza di un certo numero di prove storiche a determinare la validità di un brevetto.

In realtà, è alquanto strano che British Telecom presentò domanda nel 1975 per il brevetto sugli “hyperlink accoppiato alla connessione telefonica”. Credo fosse nel 1974 che il sottoscritto sviluppò per la prima volta il pacchetto Info, grazie al quale è possibile collegare tra loro gli hyperlink, mentre gli utenti usavano il telefono per accedere al sistema. Di fatto avevo realizzato un'invenzione precedente a quel brevetto. Questa è la seconda idea brevettabile che so di aver avuto in vita mia.

Ma non credo di avere alcuna prova al riguardo. Non l'avevo considerata sufficientemente importante da pubblicarla. Dopo tutto, l'idea di seguire gli hyperlink mi venne dalla dimostrazione dell'elaboratore creato da Doug Engelbart. Fu lui ad avere un'idea interessante da pubblicare. Quel che feci io, lo definii “ipertesto del pover'uomo”, poiché dovetti implementarlo nel contesto del TECO [acronimo per Text Editor and COrrector, era l'aggiornamento di un elaboratore testi per telescriventi, adattato da Stallman alla macchina PDP-6 operante nel Laboratorio di Intelligenza Artificiale del MIT, con innovazioni importanti per quei tempi, primi anni '70, come i testi a tutto schermo]. Non risultò altrettanto potente del suo ipertesto, ma almeno si dimostrò utile per naviga-

re nella documentazione, che era poi l'obiettivo finale. E per quanto concerne l'accesso via telefono al sistema, bé, funzionava così, non mi venne in mente che esistesse una relazione particolare tra le due cose. Non pensai di dover pubblicare una ricerca per dire: "Ho realizzato l'ipertesto del pover'uomo, e indovinate un po', c'è la linea telefonica anche nel computer!".

Sospetto non esista alcun modo per stabilire con esattezza la data in cui riuscii a implementare tutto ciò. Venne forse pubblicato in qualche modo? Bé, invitammo alcuni ospiti dal giro di ARPANET a collegarsi online dalla nostra macchina – può darsi che navigando nella documentazione usando il pacchetto Info si siano accorti della cosa. Se ce l'avessero chiesto, avrebbero scoperto l'esistenza dell'accesso tramite la chiamata telefonica. Come è possibile notare, quindi, sono le circostanze storiche a determinare l'esistenza o meno di un'opera precedente. Naturalmente esiste una pubblicazione sull'ipertesto curata da Engelbart che loro, gli imputati, si apprestano a mostrare. Non credo tuttavia dica nulla sul fatto dell'accesso telefonico presente nel computer, per cui non è chiaro se ciò potrà risultare sufficiente.

La possibilità di andare in tribunale per ribaltare il brevetto rappresenta un'opzione possibile. A causa delle spese necessarie, però, viene considerata di rado pur potendo provare l'esistenza certa di un'opera precedente che sembri sufficiente a ribaltare il brevetto. Come risultato, un brevetto non valido, un brevetto che a livello nominale non avrebbe dovuto esistere (come infatti dovrebbe essere per moltissimi brevetti), rappresenta un'arma pericolosa. Se qualcuno vi attacca con un brevetto non valido potrebbe davvero procurarvi grossi guai. Potreste bluffare tirando fuori un'opera precedente. Dipende dal fatto se ciò possa essere sufficiente per spaventarli. Potrebbero invece pensare, "Bé, stai soltanto bluffando, non

ce la farai ad andare in tribunale, non puoi permettertelo, per cui ti denunciamo lo stesso”.

Tutte e tre questi scenari costituiscono altrettante opzioni a disposizione, ma spesso è impossibile usarle. In pratica occorre affrontare un brevetto dopo l'altro. Ogni volta può darsi sia possibile ricorrere a una di tali opzioni, ma subito dopo c'è un altro brevetto e poi un altro ancora. È come attraversare un campo minato. È difficile che a ogni passo, a ogni decisione progettuale, si possa cadere su un brevetto esistente, e per un raggio limitato è probabile non ci sia alcuna esplosione. Ma le probabilità di riuscire ad attraversare indenni il campo minato e sviluppare il programma che si ha in mente senza mai inciampare in un brevetto, diminuiscono in maniera direttamente proporzionale all'ampiezza del programma. A questo punto, qualcuno è solito chiedermi: “Anche in altri settori esistono i brevetti, perché mai il software dovrebbe esserne esente?”. Notiamo la stranezza di questa supposizione, per cui in qualche modo saremmo tutti costretti a soffrire passando attraverso il sistema dei brevetti. È come dire: “C'è gente che si prende il cancro, perché non dovresti averlo anche tu?”. Per come la vedo io, è un bene che non tutti siano malati di cancro.

Ma dietro quest'aspetto si nasconde una domanda meno pregiudiziale, una buona domanda: il software è forse diverso da altri settori? Le politiche sui brevetti dovrebbero forse essere diverse per ciascun ambito? Se sì, perché mai?

Consideriamo l'intera questione: i brevetti hanno funzionalità diverse a seconda dei settori, perché si comportano altrettanto diversamente con i rispettivi prodotti.

A un estremo abbiamo l'industria farmaceutica, dove una determinata formula chimica ottiene il brevetto in modo tale che questo copra un unico e singolo prodotto. Una nuova medicina non può

essere coperta da un brevetto preesistente. Se dev'esserci un brevetto per questo nuovo prodotto, verrà assegnato a chiunque lo abbia sviluppato.

Ciò è coerente con l'idea infantile del sistema dei brevetti che abbiamo oggi: se hai realizzato qualcosa di nuovo, te ne spetta "il brevetto". L'idea è che a ciascun prodotto corrisponda un brevetto in grado di coprire l'idea alla base di quel prodotto. In alcuni settori tale scenario è vicino alla realtà; in altri assai lontano.

Il software rientra all'estremo opposto di questa seconda categoria: ciascun programma interseca numerosi brevetti. Ciò per via del fatto che normalmente i pacchetti software sono di ampie dimensioni. Fanno uso di molte idee diverse in combinazione tra loro. Se il programma è nuovo e non soltanto copiato, allora è probabile ricorra a una differente combinazione di idee – inserite, ovviamente, all'interno di codice sorgente interamente riscritto, perché è impossibile limitarsi a enunciare tali idee e farle funzionare come per magia. Bisogna implementarle una dopo l'altra all'interno di quella combinazione.

Ne risulta che anche nella stesura di un programma si fa uso di molte idee differenti, ciascuna delle quali potrebbe essere stata brevettata da persone diverse. In ogni programma esistono perciò migliaia di elementi, migliaia di punti vulnerabili potenzialmente già coperti dal brevetto di qualcuno.

Ecco perché i brevetti sul software tendono a ostacolare il progresso del software – il lavoro di sviluppo di un programma. Se fosse "un brevetto, un prodotto", allora i brevetti non impedirebbero lo sviluppo di nuovi prodotti perché è impossibile che ciascuno di questi sia stato già brevettato da qualcuno. Ma quando un programma è il risultato della combinazione di parecchie idee diverse, è assai probabile che il nuovo prodotto (in parte o per intero) sia già coperto da qualche brevetto precedente.

Non a caso una recente indagine economica rileva proprio come l'imposizione del sistema dei brevetti in un settore basato sull'innovazione per incrementi possa rallentarne il progresso. I sostenitori del sistema dei brevetti dicono: "Sì, è vero, possono nascere dei problemi, ma ancora più importante è il fatto che i brevetti debbano promuovere l'innovazione, e ciò è talmente importante che non importa quanti problemi possano provocare". Naturalmente si guardano bene dal dirlo ad alta voce perché è un'affermazione ridicola, ma implicitamente vogliono farci credere che fino a quando il sistema dei brevetti riesce a stimolare il progresso, ciò supera qualsiasi costo possibile. Ma in realtà non esiste motivo per ritenere che ciò sia effettivamente in grado di stimolare il progresso. Oggi esiste un modello preciso a dimostrazione delle modalità con cui i brevetti possono rallentare il progresso. Il caso in cui applicare tale modello descrive abbastanza bene il campo del software, un campo/sistema a innovazione incrementale.

Perché il software si trova all'estremità opposta dello spettro? Il motivo è che nel software sviluppiamo oggetti matematici astratti. Si può costruire un castello complicato e poggiarlo su una linea sottile, si reggerà perché non pesa nulla. In altri settori, si ha a che fare con la perversità della materia, degli oggetti fisici. La materia è qualcosa di ben preciso. Possiamo tentare di modellarla, ma se il comportamento reale non corrisponde al modello predisposto allora sono guai, perché la sfida consiste nel costruire oggetti materiali capaci di funzionare sul serio.

Se voglio inserire un costrutto "if" all'interno di un "while" non devo preoccuparmi se il costrutto "if" possa oscillare a una determinata frequenza e collida con il ciclo "while" provocando la rottura delle due strutture. [L'intero esempio è basato su "if" e "while", due costrutti usati nella programmazione]. Non ho bisogno di

preoccuparmi se ciò possa oscillare a una frequenza così alta da indurre una iniezione di segnale che provochi un cambiamento di valore di qualche altra variabile. Né devo preoccuparmi di quanta corrente attraversi il costrutto “if” e se questo possa dissiparla in calore all’interno del ciclo “while”, o se possa verificarsi un calo di voltaggio all’interno del ciclo “while” tale da impedire il funzionamento del costrutto “if”. Neppure devo preoccuparmi del fatto che, nel caso faccia girare il programma in un ambiente con acqua salata, il sale possa infiltrarsi tra il costrutto “if” e il ciclo “while” e causare corrosione. [Il pubblico ride durante tutto il corso della descrizione].

Non devo preoccuparmi, quando utilizzo il valore di una variabile, se stia superando il limite di fan-out utilizzandola 20 volte. Né devo preoccuparmi della sua capacità massima, e se esista tempo sufficiente per caricarla alla giusta tensione.

Quando scrivo un programma, non ho bisogno di preoccuparmi di come in seguito dovrò assemblare materialmente ogni copia del programma, e se possa riuscire ad avere spazio sufficiente per infilare quel costrutto “if” all’interno del ciclo “while”. Né devo preoccuparmi di come aprire l’apparato nell’eventualità di una rottura del costrutto “if” per rimuoverlo e sostituirlo con uno nuovo. Ci sono così tanti problemi di cui non dobbiamo preoccuparci con il software; ciò rende sostanzialmente più semplice scrivere un programma anziché progettare un oggetto materiale capace di funzionare.

Ciò potrà apparire strano, perché probabilmente avrete sentito dire in giro quanto sia difficile progettare del software, e quanto sia complicato trovare soluzioni adeguate ai vari problemi. Non si tratta della medesima questione che sto illustrando ora. Il confronto cui mi riferivo riguarda i sistemi di software e quelli materiali aventi una complessità analoga, un identico numero di componenti.

Ritengo che un sistema di software sia molto più facile da progettare di un sistema fisico. Ma l'intelligenza usata in questi campi diversi è la stessa, e allora cosa facciamo quando ci troviamo a operare in un contesto semplice? Decidiamo di andare più avanti! Spingiamo al limite massimo le nostre capacità. Di fronte alla semplicità dei sistemi di dimensioni analoghe, ne aumentiamo la grandezza di dieci volte – allora sì che diventeranno difficili! Ecco cosa facciamo: costruiamo sistemi di software molto più estesi, in termini di numero dei componenti, dei sistemi fisici.

Un sistema fisico il cui progetto preveda un milione di pezzi diversi diventa un megaprogetto. Un programma informatico che include un milione di pezzi raggiunge forse le 300.000 righe di codice; un pugno di persone impiegheranno un paio d'anni per scriverlo. Non si tratta di un programma particolarmente gigantesco. Oggi GNU Emacs conta svariati milioni di pezzi, credo. È composto da un milione di righe di codice. Si tratta di un progetto realizzato essenzialmente senza alcun tipo di sostegno economico, in gran parte scritto da varia gente nel proprio tempo libero.

Il software offre anche un altro grosso risparmio. Dopo aver progettato un prodotto fisico, il passo successivo concerne la costruzione della fabbrica dove produrlo. Operazione che potrà costare milioni o decine di milioni di dollari, laddove per fare delle copie di un programma è sufficiente digitare "copia". Lo stesso comando consente di copiare qualsiasi programma. Volendo copiare su un CD, basta realizzare il master e spedirlo a un produttore di CD. Qui verranno utilizzate le medesime apparecchiature impiegate per copiare qualsiasi contenuto su un comune CD. Non bisogna costruire una fabbrica specializzata capace di produrre ogni articolo specifico. Il tutto comporta una semplificazione enorme e la drastica riduzione dei costi nella fase di progettazione.

Un'azienda automobilistica, che spenderà 50 milioni di dollari nella costruzione della fabbrica in cui verrà prodotto un nuovo modello di autovettura, può assumere degli avvocati per occuparsi delle trattative sulle licenze dei brevetti. Volendo, potranno anche risolvere felicemente eventuali denunce legali. La progettazione di un programma di analoga complessità potrà costare 50.000 o 100.000 dollari. Al confronto, le spese per trattare con il sistema dei brevetti sono schiaccianti – anzi, la progettazione di un programma avente le stesse complessità del progetto meccanico di un'autovettura richiede forse un mese di lavoro. Di quante parti è composta un'automobile... meglio, un'automobile priva di sistemi computerizzati? Ciò non vuol dire che sia facile progettare un buon modello, soltanto che questo non include poi così tante componenti.

(La trasmissione automatica è composta da circa 300-400 pezzi unici, e generalmente questa è la parte più complicata di un autoveicolo. La fase di progettazione della trasmissione può richiedere dai sei mesi a un anno, e a quel punto ci vorrà ancora più tempo per costruirla e renderla operativa. Invece un programma dotato di 500-800 parti funzionanti sarà praticamente composto da 200-300 righe di codice, e probabilmente un buon programmatore impiegherà da un giorno a una settimana per realizzarlo, incluse prove e collaudi).

Ne risulta che il software è veramente diverso da altri settori, perché quando si lavora con elementi matematici la progettazione di qualcosa è infinitamente più semplice. Di conseguenza possiamo realizzare regolarmente sistemi molto, molto più grandi grazie appena a un paio di persone. Il risultato è che invece di essere vicini a “un brevetto, un prodotto”, ci troviamo in un sistema in cui ciascun prodotto ingloba un'enorme quantità di idee che potrebbero essere già state brevettate.

Il modo migliore per illustrare questa situazione è l'analogia con le sinfonie di musica classica. Anche una sinfonia è lunga e comprende parecchie note diverse, e probabilmente usa un gran numero di idee musicali. Proviamo a immaginare cosa accadrebbe se i governi dell'Europa del 1700 avessero deciso di promuovere il progresso della musica sinfonica tramite l'attivazione di un ufficio brevetti per la musica europea, con il compito di assegnare i brevetti a ogni tipo di idea musicale che fosse possibile descrivere a parole.

Immaginiamo di trovarci verso il 1800 e di impersonare Beethoven alle prese con la stesura di una sinfonia. Scoprirete ben presto come metterne insieme una che non infranga nessun brevetto, sia qualcosa di assai più arduo che scrivere una buona sinfonia.

Quando ve ne lamentate, i vari detentori brevetti potrebbero rispondere: "Ah, Beethoven, ti lamenti soltanto perché non hai idee originali. Tutto quello che vuoi fare è rubare le nostre invenzioni". In realtà Beethoven ha un sacco di nuove idee musicali – ma deve anche usarne parecchie tra quelle esistenti per rendere riconoscibile la sua musica, in modo che possa piacere agli ascoltatori, i quali devono identificarla in quanto musica. Nessuno è talmente brillante da poter reinventare della musica completamente differente e realizzare al contempo qualcosa a cui si voglia prestare ascolto. Pierre Boulez disse di volerci provare, ma quanta gente ne ascolta la musica?

Nessuno è così brillante da poter reinventare tutta l'informatica, per rifarla completamente da capo. Se qualcuno potesse riuscirci, il risultato sarebbe talmente strano che gli utenti si rifiuterebbero di utilizzarla. Quando consideriamo un elaboratore testi odierno, vi scopriremo, credo, centinaia di funzioni diverse. Se qualcuno sviluppa un elaboratore testi nuovo e ben fatto, ciò vuol dire che presenta delle idee nuove, ma dovrà comprendere anche centinaia di

idee preesistenti. Nel caso fosse illegale usarle, risulterebbe impossibile realizzare un elaboratore testi innovativo. Poiché il lavoro dello sviluppo del software è così ampio, ne risulta che non abbiamo alcun bisogno di schemi artificiali per incentivare nuove idee. Basta avere qualcuno che voglia scrivere del software e l'ispirazione non mancherà di arrivare. Se volete scrivere un programma di buon livello, vi verranno sicuramente delle idee e troverete il modo di applicarne alcune.

Visto che opero nel campo del software fin da prima dell'arrivo dei relativi brevetti, di solito succedeva che la maggior parte degli sviluppatori pubblicava qualsiasi nuova idea ritenuta valida, per le quali ritenevano di poter meritare qualche lode o riconoscimento. Le idee troppo ridotte o non sufficientemente valide non venivano pubblicate perché sarebbe stato sciocco farlo. Ora, si suppone che il sistema dei brevetti debba incoraggiare la manifestazione delle idee. In realtà in passato nessuno le custodiva gelosamente. È vero che tenevano segreto il codice. In fondo scrivere codice rappresentava il grosso dell'attività. Non rivelavano il codice e ne pubblicavano le idee, in modo che gli sviluppatori potessero ottenerne qualche riconoscimento e sentirsi apprezzati.

Dopo l'introduzione del sistema dei brevetti, continuarono a tenere segreto il codice brevettando al contempo le idee, e di fatto non viene fatto assolutamente nulla per incoraggiare la diffusione delle idee. Quello che veniva tenuto segreto allora rimane tale ora, ma le idee che solitamente venivano pubblicate in modo che altri potessero usarle oggi è probabile vengano brevettate e tenute fuori portata per 20 anni.

Cosa può fare un paese per cambiare questa situazione? In che modo dovremmo riformare l'attuale politica onde risolvere il problema? Due sono i fronti che è possibile attaccare. Uno è il luogo fisico

addetto al rilascio dei brevetti, l'omonimo ufficio. L'altro è laddove tali brevetti trovano applicazione. Questa faccenda riguarda quel che copre effettivamente un brevetto.

Un modo è quello di stabilire un buon criterio per l'assegnazione dei brevetti. Ciò può funzionare in un paese che finora non ha ancora autorizzato il ricorso ai brevetti sul software, come accade ad esempio nella maggior parte dei paesi europei. Una buona soluzione per l'Europa sarebbe semplicemente quella di rafforzare con chiarezza le norme dell'ufficio brevetti europeo, che stabiliscono la non brevettabilità del software. Attualmente l'Europa sta vagliando una direttiva per i brevetti sul software. (Credo che tale direttiva sia di portata più ampia, ma una delle implicazioni più importanti riguarda i brevetti sul software). Sarebbe sufficiente modificarla ribadendo che le idee sul software non possono essere coperte da brevetti, così da tenere gran parte dei problemi fuori dall'Europa, fatta eccezione per alcuni paesi che potrebbero trovarsi davanti a problemi interni, e uno di questi purtroppo è la Gran Bretagna (purtroppo per voi).

Un approccio simile non funzionerebbe negli Stati Uniti. Il motivo è che qui esiste già un'ampia quantità di brevetti sul software, e qualsiasi mutamento nel criterio per l'assegnazione non potrà liberarsi di quelli precedenti.

(Quando parlo di "brevetti sul software" cosa intendo dire in realtà? L'ufficio brevetti statunitense non divide ufficialmente i brevetti sul software dagli altri. Così qualsiasi brevetto che è possibile applicare a qualche tipo di software viene considerato la base presumibilmente valida per poter denunciare chiunque scriva dei programmi. I brevetti sul software sono brevetti che si possono potenzialmente applicare al software, brevetti che potenzialmente possono motivare la denuncia contro chi sviluppa del software).

Così per gli Stati Uniti la soluzione dovrebbe materializzarsi trami-

te il cambiamento dell'applicabilità, dello scopo dei brevetti: affermando cioè che la pura implementazione del software, operante su un hardware generico che in sé non infrange il brevetto, non è coperta da alcun brevetto e non si può subire alcuna denuncia unicamente su tali basi. Questo è l'altro tipo di soluzione possibile, mentre la prima, quella relativa ai tipi di brevetti che possono risultare validi, è una buona soluzione da applicare in Europa.

Quando negli Stati Uniti venne introdotto il sistema dei brevetti non ci fu alcun dibattito politico. Anzi, non se ne accorse nessuno. Per la maggior parte, neppure quanti operavano nel campo del software ne presero nota. Nel 1981 una decisione della Corte Suprema prese in esame il brevetto su un procedimento per la lavorazione della gomma. Secondo la sentenza, il fatto che l'apparecchiatura in questione fosse dotata di computer e di programma come parte del processo per la lavorazione della gomma, non ne impediva la brevettabilità. L'anno successivo, la corte d'appello che si occupa di tutti i casi relativi ai brevetti chiarì meglio il concetto: l'esistenza di un computer e di un programma rende il prodotto brevettabile. Il fatto che all'interno di un oggetto qualsiasi ci sia un computer e un programma, consente la brevettabilità di tale oggetto. Ecco perché negli Stati Uniti piovvero le richieste di brevetti sulle procedure commerciali: queste venivano eseguite tramite un computer e ciò le rendeva brevettabili.

Così venne emanata quella sentenza, e subito dopo credo che il brevetto per il ricalcolo secondo l'ordine naturale fosse uno dei primi a essere assegnato, se non addirittura il primo.

Per tutti gli anni '80 non ne sapemmo nulla. Fu intorno al 1990 che i programmatori statunitensi iniziarono a rendersi conto dei pericoli cui andavano incontro con il sistema dei brevetti. Ho visto come operava il settore prima di quel periodo e come lo fece dopo.

Dopo il 1990 non notai alcuna particolare accelerazione del progresso operativo.

Negli Stati Uniti non si ebbe alcun dibattito politico, ma in Europa se ne è avuto uno di ampie proporzioni. Parecchi anni fa vennero segnalate forti pressioni per apportare degli emendamenti al trattato di Monaco che implementava l'ufficio europeo dei brevetti. Una clausola del documento stabilisce la non brevettabilità del software. Le pressioni miravano a modificare tale clausola in modo da iniziare a consentire i brevetti sul software. Ma la comunità si accorse di questa manovra. Furono anzi gli sviluppatori e gli utenti di software libero a guidare le proteste. Non siamo solo noi a subire i pericoli del sistema dei brevetti. Ogni sviluppatore ne è minacciato, e lo stesso vale anche per gli utenti.

Ad esempio, Paul Heckel – dopo che la Apple non venne intimidita dalle sue minacce – avvertì che avrebbe preso a denunciarne gli utenti. L'eventualità preoccupò non poco la Apple, la quale comprese che non poteva permettersi di lasciar denunciare i propri clienti a quel modo, anche se in ultima analisi avrebbero vinto la causa. Ma il punto è che anche gli utenti possono subire una denuncia, sia come modo per attaccare gli sviluppatori sia soltanto per spremere loro dei soldi o provocare gravi danni. Tutti gli sviluppatori e gli utenti sono vulnerabili.

Ma in Europa è stata la comunità del software libero a organizzare l'opposizione. Fu così che per due volte i paesi responsabili dell'ufficio europeo dei brevetti votarono no all'emendamento del trattato. Allora intervenne l'Unione Europea e le varie commissioni si mostrarono divise sulla questione. Quella il cui compito riguarda la promozione del software è contro i brevetti, almeno così pare, ma non aveva potere decisionale su questo tema. Ne è responsabile la commissione sul libero mercato, e chi la presiede sembra favo-

revoles ai brevetti sul software. In pratica tale commissione non ha tenuto alcun conto delle posizioni espresse dal pubblico, proponendo una direttiva che consente i brevetti sul software.

Il governo francese ha già dichiarato la propria opposizione. Molta gente sta facendo pressione sui vari governi nazionali affinché si oppongano ai brevetti sul software, ed è vitale iniziare a muoversi anche qui in Gran Bretagna. Secondo Hartmut Pilch, uno dei leader europei nella battaglia contro i brevetti sul software, l'impeto maggiore arriva dall'ufficio brevetti britannico, il quale è aprioristicamente a favore dei brevetti sul software. L'ufficio britannico ha condotto una serie di consultazioni pubbliche, rivelatesi in maggioranza di segno contrario. Poi ha diffuso un documento in cui si sostiene che la gente sembra apprezzare quei brevetti, ignorando completamente le risposte ricevute dal pubblico. In ogni caso, la comunità del software libero aveva avvisato gli utenti: "Per favore inviate le risposte sia a loro che a noi". Così hanno pubblicato tali risposte, che in genere esprimevano opposizione. Sarebbe stato impossibile desumere tutto ciò dal rapporto pubblicato dall'ufficio brevetti britannico.

Questo ricorre spesso a un termine chiamato "effetto tecnico". È una definizione che può essere ampliata in maniera tremenda. Dovremmo credere che questa stia a indicare che l'idea di un programma possa essere brevettata soltanto nel caso in cui si riferisca a specifiche azioni fisiche. Se questa è l'interpretazione corretta, in gran parte risolverebbe ogni problema. Se fosse davvero possibile brevettare soltanto le idee di un programma effettivamente correlate allo specifico risultato tecnico, fisico brevettabile in assenza di tale programma, ciò andrebbe bene. Il problema sta nel fatto che quel termine può subire delle estensioni. Quel che si ottiene facendo girare un certo programma può essere descritto come un effet-

to fisico. In che modo quest'ultimo si differenzia da qualsiasi altro risultato? Bé, lo è in quanto deriva da quel calcolo specifico. Di conseguenza l'ufficio brevetti britannico sta proponendo qualcosa che sembra condurre per lo più alla soluzione del problema, ma che in realtà offre carta bianca per poter brevettare quasi ogni cosa.

I responsabili dello stesso dipartimento sono coinvolti anche sulle tematiche del copyright, che in realtà non c'entra nulla con i brevetti sul software, eccetto per il fatto che in questo caso se ne occupano le stesse persone. (Forse sono stati indotti dal termine "proprietà intellettuale" a mettere insieme le due questioni). Si tratta di interpretare la recente direttiva dell'Unione Europea in tema di copyright, normativa orribile tanto quanto il Digital Millennium Copyright Act statunitense, pur se i singoli paesi hanno qualche spazio di manovra sulla sua implementazione. La Gran Bretagna vorrebbe massimizzare l'effetto tirannico della direttiva. Sembra che sia un certo gruppo – forse il Ministero del Commercio e dell'Industria? – a meritare la nostra attenzione. È necessario monitorarne le attività, in modo da bloccare la creazione di nuove forme di potere.

I brevetti sul software possono incastrare ogni sviluppatore e ogni utente informatico in una nuova forma di burocrazia. Se gli imprenditori che utilizzano i computer riuscissero a comprendere il gran numero di problemi che ciò finirà per provocare loro, sarebbero pronti a dar battaglia, e sono sicuro che riuscirebbero a fermare queste iniziative. L'imprenditoria non ha alcuna voglia di farsi legare le mani dalla burocrazia. Naturalmente, talvolta questa è utile al raggiungimento di obiettivi importanti. Esistono alcuni settori in cui vorremmo che il governo britannico si fosse dimostrato più rigoroso nell'imporre maggiore burocrazia a certe aziende, come per lo spostamento e il commercio di animali (onde rendere difficile la

diffusione della variante del morbo di Creutzfeldt-Jacob, meglio noto come “mucca pazza”). Ma nei casi in cui ciò non persegue altro scopo se non la creazione di monopoli artificiali in modo che qualcuno possa interferire con lo sviluppo dei programmi – spremendo denaro dagli sviluppatori e dagli utenti – allora dovremmo opporre un rifiuto. Dobbiamo informare i dirigenti imprenditoriali sulle conseguenze dei brevetti sul software nei loro confronti, così da ottenerne il sostegno nella lotta contro i brevetti sul software in Europa.

La battaglia non è ancora finita. Possiamo ancora vincerla.

Trascrizione dell'intervento tenuto all'Università di Cambridge, Londra, il 25 marzo 2002. Questa versione fa parte del libro *Free Software, Free Society: The Selected Essays of Richard M. Stallman*, GNU Press, 2002.

La copia letterale e la distribuzione di questo testo nella sua integrità sono permesse con qualsiasi mezzo, a condizione che sia mantenuta questa nota.

# Appendice



# Risorse utili

Per una buona infarinatura generale, è bene iniziare da questi libri italiani:

AA.VV., *Open Sources: Voci dalla rivoluzione open source*, Apogeo, 1999, euro 14,46 (disponibile anche online: <http://www.apogeonline.com/libri/00545/scheda>)

Mariella Berra e Angelo Raffaele Meo, *Informatica Solidale: Storia e prospettive del software libero*, Bollati Boringhieri, 2001, euro 14,46

Sam Williams, *Codice Libero (Free as in Freedom): Richard Stallman e la crociata per il software libero*, Apogeo, 2003, euro 14 (disponibile anche online: <http://www.apogeonline.com/libri/02108/scheda>)

Per approfondimenti e aggiornamenti vari, meglio sbizzarrirsi sul web. Questi sono alcuni dei maggiori siti da seguire, a partire ovviamente da quelli in inglese:

Free Software Foundation e GNU Project: <http://www.fsf.org>

Sito personale di Richard Stallman: <http://www.stallman.org>

Free Software Foundation Europe: <http://www.fsfurope.org/>

Eurolinux alliance: <http://www.eurolinux.org>

Associazione Software Libero: <http://www.softwarelibero.it>

Software libero nella didattica: <http://scuola.softwarelibero.org>

PLUTO Free Software Users Group: <http://pluto.linux.it>

## **Associazione Software Libero: fondazione e storia**

L'Associazione Software Libero (Assoli) è un'entità legale senza scopo di lucro che nasce nel novembre 2000 e che annovera, tra i suoi obiettivi, la diffusione del software libero in Italia e una corretta infor-

mazione sull'argomento. Nel maggio 2002, diventa l'affiliata italiana della Free Software Foundation Europe e, attraverso le sue liste (in particolare [discussioni@softwarelibero.it](mailto:discussioni@softwarelibero.it) e [diritto@softwarelibero.it](mailto:diritto@softwarelibero.it)), riesce a radunare circa duecento persone interessate a dibattere di licenze, questioni legali, avvicinamento alla pubblica amministrazione e attività pubbliche a sostegno del software libero.

La decisione di creare Assoli nasce da una semplice constatazione: se il software libero, dal punto di vista tecnico, ha iniziato ad attecchire ormai da qualche anno, non è accaduto altrettanto per la comprensione dei diversi tipi di licenza e per le conseguenze giuridiche della loro adozione. Lo prova la confusione - ancora attuale anche negli ambienti degli "addetti ai lavori" - verso termini come freeware, shareware, open source e software libero, utilizzati come sinonimi quando invece le differenze che queste diverse forme di software hanno in termini di uso privato e aziendale, creazione di un mercato e incentivo allo sviluppo tecnologico sono notevoli, specialmente in un'ottica di lungo periodo.

Altra prova della diffusa mancanza di conoscenza sull'argomento è stata la legge italiana sul diritto d'autore (n. 248/2000), legge dove tutto il software è stato equiparato a quello proprietario, determinando così una effettiva difficoltà per la diffusione di software distribuito con licenze differenti. Gli ostacoli insorti al momento dell'entrata in vigore della legge sono stati in parte corretti dal regolamento attuativo che segue di oltre un anno la nuova normativa e le recenti modifiche a esso apportate, ma permangono ancora oggi problemi alla diffusione del software libero, collegati a legislazioni potenzialmente restrittive, come la direttiva europea 2001/29/CE (European Union Copyright Directive, Eucd).

### *I progetti*

Eucd: le conseguenze e i pericoli (<http://www.softwarelibero.it/progetti/eucd/index.shtml>).

La campagna nasce per diffondere una maggiore conoscenza della

direttiva europea 2001/29/CE, più nota come European Union Copyright Directive (Eucd). Il provvedimento introduce una serie di novità legali nel campo del “diritto d’autore”, puntando unicamente alla salvaguardia degli interessi economici dei grossi editori e dei produttori di software proprietario. I diritti degli utenti (e non solo) sono messi completamente in secondo piano, se non addirittura calpestati. Le norme della direttiva mettono in grave pericolo il diritto alla copia privata, la possibilità di usufruire delle opere in formato digitale (come e-book, dvd, cd musicali) secondo condizioni ragionevoli, la futura garanzia di poter accedere senza censure a documenti di rilevanza storica, la possibilità di cedere o rivendere materiale digitale regolarmente acquisito, la possibilità di produrre software libero interoperante, la libertà di ricerca e di espressione su Internet. L’applicazione dell’Eucd in Italia appare molto vicina, dato che è già pronto uno schema di decreto legislativo per il recepimento della direttiva. Esiste già una legge in vigore che applica le stesse norme previste dall’Eucd: si tratta dello statunitense Digital Millennium Copyright Act (DMCA). <http://www.anti-dmca.org/>).

Bollino Howto (<http://www.softwarelibero.it/bollino/html/Bollino-HOWTO.html>).

Dal momento dell’approvazione della legge 248/2000, nota come “legge del bollino”, e del successivo regolamento attuativo, molti gruppi e osservatori hanno mosso varie critiche e osservazioni a questi testi, osservazioni incentrate su aspetti legati alla possibilità di una reale applicazione della legge e ai diritti dei consumatori. Inoltre si sono andate delineando difficoltà a cui sarebbero andate incontro piccole realtà produttive esistenti in Italia. Per questo, l’Associazione Software Libero e il Lug Roma (<http://www.lugroma.org/>) hanno cercato di capire come far “convivere” il software libero con questa legge: sono avviati dei contatti con la sede centrale della Siae, con gli organi preposti per discutere delle problematiche che l’interpretazione della legge pone in relazione alle opere libere e delle possibili solu-

zioni per rendere manifesta l'esclusione di questo genere di opere dall'ambito di applicazione della normativa. In seguito al primo incontro e alla luce dei chiarimenti avuti, l'Associazione Software Libero e il progetto GNUtemberg! (<http://www.gnutemberg.org/>) hanno presentato, in luoghi e circostanze diversi, e ottenuto una richiesta di esenzione. Sono seguiti i cd-rom e il materiale creato da numerosi Lug (Linux User Group), distribuiti sul territorio nazionale. Il documento è stato scritto, revisionato e pubblicato con la volontà di favorire la conoscenza della legge e degli strumenti che questa mette a disposizione per evitare l'applicazione del contrassegno.

### *Formati*

Il progetto ha come obiettivo la definizione di formati di dati e di formati di dati liberi individuando quali siano le migliori applicazioni nell'ambito pubblico e privato. Ogni volta che si usa un applicativo software, vengono prodotti dati, generalmente memorizzati su hard disk, floppy o altri supporti oppure inviati a un altro elaboratore via rete. Poiché i dati sono di proprietà dell'utente che li ha creati, è fondamentale che egli possa disporre di essi, indipendentemente dal formato di memorizzazione o di trasmissione utilizzato. In altre parole, l'utente deve essere nella condizione di accedere ai propri dati conservando la libertà di scelta del software da utilizzare. Affinché i dati siano utili, è necessario che utenti differenti possano dividerli e utilizzarli, senza alcun vincolo di dipendenza da un unico produttore. Dunque, un requisito nella creazione e nell'accesso ai dati è costituito da formati chiari, facilmente accessibili e riutilizzabili all'interno di prodotti differenti. Benché il concetto di libertà dei formati di dati sia strettamente correlato al concetto di libertà del software, una definizione di formati di dati liberi prescinde dalla natura del software essendo valida sia per il software libero che per quello proprietario.

Dizionario libero (<http://www.softwarelibero.it/progetti/dizionario/>). Il Progetto Dizionario Libero ha come obiettivo la realizzazione di un

dizionario italiano e di ulteriori strumenti linguistici disponibili al pubblico sotto licenza libera. Uno degli aspetti in cui il software libero in italiano si è dimostrato più carente è quello della mancanza di un vocabolario di qualità sufficientemente elevata. Oltre a ciò, manca in generale quello che invece è disponibile per molte altre lingue, come un dizionario e una raccolta di sinonimi e contrari. Questo progetto mira a costruire le basi necessarie per colmare queste lacune e i risultati saranno rilasciati con licenza libera (GPL, LGPL o FDL, a seconda dei casi). Il primo passo è stato quello di creare una mailing list di coordinamento ([dizionario@softwarelibero.it](mailto:dizionario@softwarelibero.it)) e qui si stabiliscono le modalità di evoluzione del progetto, obiettivi ulteriori e modalità di realizzazione. Il secondo passo è la creazione di deposito centralizzato per una lista di parole semplici, a cui diventi possibile fare riferimento come punto di raccolta per la stesura del vocabolario. A questo si aggiungerebbe una procedura automatica per la raccolta di nuove parole, e procedure più o meno automatiche per la relativa integrazione.

Storia del software libero in Italia (<http://www.softwarelibero.it/progetti/storia/>).

Scopo del progetto è l'individuazione e la descrizione dei personaggi e dei momenti che hanno contribuito alla diffusione del software libero e del movimento di pensiero a esso collegato. Il lavoro, attualmente in via di sviluppo, è aperto a interventi, suggerimenti, contributi, che possono essere sottoposti all'indirizzo della mailing list di coordinamento, [storia@softwarelibero.it](mailto:storia@softwarelibero.it)

### *Le attività*

Una parte importante del lavoro di Assoli è la partecipazione a conferenze ed eventi pubblici presentando quelli che sono i concetti filosofici e legali correlati al software libero. Inoltre, sono state portate avanti iniziative in collaborazione con associazioni che si occupano di argomenti simili. Tra queste, insieme alla Italian Linux Society

(ILS, <http://www.linux.it/>), ha sollecitato una raccolta di firme per sostenere la discussione in parlamento del disegno di legge sul software libero: “Norme in materia di pluralismo informatico sulla adozione e la diffusione del software libero e sulla portabilità dei documenti informatici nella Pubblica Amministrazione” (XIV Legislatura Atto Senato n. 1188, [www.parlamento.it/leg/14/Bgt/Schede/Ddliter/16976.htm](http://www.parlamento.it/leg/14/Bgt/Schede/Ddliter/16976.htm)), attualmente all’esame della Commissione Affari Costituzionali del Senato). In proposito, ha lavorato anche Promezio.net (<http://openmind.promezio.net/>), Opensource.it (<http://www.opensource.it/disegnolegge.php>) e ezboard (<http://pub47.ezboard.com/fsicurezzaforum2.showMessage?topicID=68.topic>).

Con Agnug (Associazione Gnug, <http://www.gnug.it/>) e alla sezione italiana delle Free Software Foundation Europe (<http://www.fsfeurope.org/>), sostiene la campagna “Libera il tuo software!” (<http://www.liberailsoftware.org/>) per la creazione di una rete di economia solidale a sostegno dello sviluppo del software libero. Il primo obiettivo è favorire la realizzazione in tempi brevi della nuova versione di Samba, in grado di consentire una migrazione indolore dei server Windows Nt a Gnu/Linux piuttosto che a Windows 2000.

Assoli ha infine contribuito alla realizzazione delle mozioni comunali per l’introduzione del software libero e dei formati liberi con una serie di gruppi consiliari italiani, tra cui Firenze, Torino e Bologna. Per maggiori informazioni e contatti: <http://www.softwarelibero.it/>

# Indice

Introduzione .....	3
<b>PARTE PRIMA:</b>	
<b>Il progetto GNU e il software libero .....</b>	<b>9</b>
Il progetto GNU .....	11
Il manifesto GNU .....	42
La definizione di software libero .....	59
Perché il software non dovrebbe avere padroni. ....	64
Cosa c'è in un nome? .....	72
Perché “software libero” è da preferire a “open source” .....	78
Rilasciare software libero se lavorate all'università .....	88
Vendere software libero .....	92
Il software libero ha bisogno di documentazione libera .....	97
La canzone del software libero .....	101
<b>PARTE SECONDA:</b>	
<b>Copyright, copyleft e brevetti. ....</b>	<b>103</b>
Il diritto di leggere. ....	105
L'interpretazione sbagliata del copyright – una serie di errori. ....	114
La scienza deve mettere da parte il copyright .....	134
Cos'è il copyleft? .....	137
Copyleft: idealismo pragmatico .....	141
Il pericolo dei brevetti sul software .....	146
<b>APPENDICE .....</b>	<b>181</b>



Contro il comune senso del pudore, contro la morale codificata, controcorrente. Questa collana vuole abbattere i muri editoriali che ancora separano e nascondono coloro che non hanno voce. Siano i muri di un carcere o quelli, ancora più inviolabili e resistenti, della vergogna e del conformismo.

Saggi scelti di Richard Stallman  
**Software libero**  
**Pensiero libero**

A cura di Bernardo Parrella e Associazione Software Libero

Traduzioni di Bernardo Parrella e Gruppo traduttori italiani del progetto GNU

Titolo originale: Free Software, Free Society: The Selected Essays of Richard M. Stallman

Copyright © 2002 Free Software Foundation, Inc.

Free Software Foundation  
59 Temple Place, Suite 330,  
Boston, MA 02111-1307, USA  
Email: [gnu@gnu.org](mailto:gnu@gnu.org) Web: <http://www.gnu.org>

Si consente la copia letterale e la distribuzione di uno o di tutti gli articoli di questo libro, nella loro integrità, a condizione che su ogni copia sia mantenuta la citazione del copyright e questa nota.

progetto grafico **Anyone!**

impaginazione **Littlered**

© 2003 **Nuovi Equilibri**

su concessione della Free Software Foundation

Casella postale 97 - 01100 Viterbo fax 0761.352751

**Attenzione!** I manoscritti inviati all'editore non si restituiscono.  
Non vengono forniti pareri e schede di lettura.  
Non si considerano testi inviati per e-mail.

finito di stampare nel mese di aprile 2003

presso la tipografia **Graffiti**  
via Catania 8 - 00040 Pavona (Roma)



