Introduzione a Linux e all’Open Source

ricerca di Alessandro De Zorzi

1 dicembre 2001

Indice

1 Premessa

1.1 Fonte e bontà delle informazioni

La principale fonte di questa breve introduzione a Linux e l'Open Source è il libro OPEN SOURCES - Voci della rivoluzione Open Source (Chris DiBona, Sam Ockman e Mark Stone) edito da Apogeo O'Reilly e distribuito sotto il marchio OpenPress (un modello allineato a quello Open Source che prevede l'accesso gratuito alle informazioni i cui contenuti sono liberamente distribuibili).

Il libro OPEN SOURCES, disponibile in formato HTML all'indirizzo http://www.apogeonline.com/openpress/libri/545/index.html, è articolato in 14 capitoli contenenti gli scritti, alcuni già noti e pubblicati, di 14 personaggi del movimento Open Source (fra i quali Linus Torvalds, Richard Stallman, Eric Raymond, Larry Wall).

Altre notizie sono state tratte da articoli disponibili in internet.

1.2 Scopo del documento

Non mi sento certo in grado di esporre la storia del movimento Open Source in modo esaustivo e soddisfacente, spero soltanto di inquadrare le idee fondamentali, in modo da incuriosire e incoraggiare ad un approfondimento. Ho riportato fedelmente parte degli scritti degli autori, integrando con alcune note per legare i contenuti e renderli più comprensibili.

2 Open Source

Linus Torvalds, il padre di Linux, racconta che il suo nome ha origine dall'ammirazione dei suoi genitori per Linus Pauling, scienziato impegnato nella scoperta del DNA che vinse il premio Nobel ben due volte.

Nella ricerca, Pauling e altri scienziati, constatarono come la riservatezza dettata dalla competizione ritarda il progresso scientifico e come le fonti di informazioni debbano essere condivise per riprodurre un risultato. Queste idee sono anche alla base dell'Open Source e dimostrano come la scienza sia in fin dei conti un'impresa Open Source. Sono stati gli ambienti scientifici la culla dell'open source.

3 L'idea di Stallman

Stallman racconta: "Quando cominciai a lavorare nel laboratorio di Intelligenza Artificiale del MIT nel 1971, entrai a far parte di una comunità in cui ci si

scambiavano i programmi, che esisteva già da molti anni. La condivisione del software non si limitava alla nostra comunità è un cosa vecchia quanto i computer, proprio come condividere le ricette è antico come il cucinare". Nei primi anni ottanta la comunità del laboratorio di Intelligenza Artificiale dove lavorava Stallman si sciolse e i nuovi elaboratori che stavano nascevano 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 ilsistema che impone che il software non possa essere condiviso o modificato - sia antisociale, contraria all'etica, semplicemente sbagliata, può apparire sorprendente. Ma che altro possiamo dire di un sistema che si basa sul dividere utenti e lasciarli senza aiuto? I produttori di software proprietario hanno lavorato a lungo e attivamente per diffondere la convinzione che c'è un solo modo di vedere la cosa, ma non è così.

3.1 Il progetto GNU

Nel 1984, Richard Stallman diede inizio al progetto GNU. Uno degli obiettivi del progetto era quello di creare un sistema operativo disponibile gratuitamente che potesse servire da piattaforma per i programmi GNU. Stallman è convinto che il codice sorgente – quindi la conoscenza – che è alla base di un programma eseguibile, dovrebbe essere condivisa e gratuita, e che la distribuzione del codice sorgente è realmente necessaria perchè l'innovazione continui.

Il teorema di Pitagora è noto a tutti, i risultati che Pitagora raggiunse sono a disposizione di chi vuole implementare un sistema che, ad esempio, calcoli l'ipotenusa di un triangolo rettangolo. La furmula è il sorgente. Se oggi usiamo una calcolatrice per fare dei calcoli e otteniamo dei risultati in modo diretto è per comodità, ma sappiamo che i risultati sono riproducibili dalla conoscenza. Se il Teorema di Pitagora fosse chiuso e utilizzabile solo all'interno di una calcolatrice in cui io introduco la lunghezza dei cateti e ottengo l'ipotenusa, sarei legato a chi fornisce la calcolatrice, che a sua discrezione potrebbe smettere di produrla, decidere di farla pagare di più, decidere di farmi pagare le pile per farla funzionare un prezzo alto, accusarmi di reverse engineering se capiso come funziona il Teorema di Pitagora dai risultati che ottengo.

Figura 1: Richard Stallman

3.1.1 GNU is not Unix

Stallman decise che c'era bisogno di nuovo sistema operativo totalmente libero e pensò di rendere il nuovo sistema operativo 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).

3.1.2 Perchè Unix

Il 1969 fu l'anno in cui un hacker dei Laboratori Bell, di nome Ken Thompson, inventò il sistema Unix. Un altro hacker, di nome Dennis Ritchie, inventò un nuovo linguaggio chiamato C, da usare con una versione Unix di Thompson ancora allo stato embrionale. Come Unix, C fu progettato per essere piacevole e facile da usare oltre che flessibile.

Per tradizione, i sistemi operativi erano stati, fino ad allora, scritti in Assembler in modo da ottenere la maggiore efficienza possibile dalle macchine host. Thompson e Ritchie furono tra i primi a capire che la tecnologia dell'hardware e dei compilatori aveva raggiunto un tale livello di maturità da poter scrivere in C un intero sistema operativo: nel 1974 l'intero ambiente operativo era regolarmente installato su numerose macchine di diversa tipologia.

Si tratta di un evento senza precedenti e le implicazioni che ne derivarono furono enormi. Oltre alla portabilità, Unix e C presentavano altri punti di forza. Entrambi si basavano sulla filosofia Keep it simple, stupid! letteralmente Semplifica, stupido!. Un programmatore poteva senza difficoltà tenere a mente l'intera struttura logica di C (a differenza di molti altri linguaggi precedenti, ma anche successivi), e non dover più ricorrere continuamente ai manuali.

Unix era un insieme flessibile di semplici strumenti che si mostravano com-

plementari l'un l'altro. Questa combinazione si rivelò adatta per una vasta gamma di operazioni, incluse alcune completamente nuove, non previste in origine dagli stessi progettisti. La sua diffusione in AT&T fu estremamente rapida, a dispetto della mancanza di programmi di supporto formale. Entro il 1980, il suo uso si era già allargato a un gran numero di università e siti di ricerca informatica, e centinaia di hacker la consideravano come la propria casa.

3.1.3 La General Public License GNU

Stallman pensò che rendendo di pubblico dominio il codice sorgente dei programmi molte aziende sarebbero state tentate di cooptare il codice per loro profitto, questo è uno dei motivi per cui ideò la GPL, una licenza molto particolare.

Secondo la GPL è possibile copiare e distribuire il software coperto da licenza GPL a piacimento a patto che non si faccia pagare il software e non se ne limiti l'uso applicando ulteriori licenze. Prevede inoltre che tutti i prodotti originati da prodotti con licenza GPL devono a loro volta essere distribuiti con licenza GPL.

Per quanto possa sembrare semplice la GPL è una licenza davvero robusta (è stata comunque modificata per adeguarsi a nuove esigenze), che è in grado di dimostrare la sua validità in casi reali in cui si è verificato il "sospetto" che parte di codice GPL fosse stato usato per software commerciale.

3.1.4 Free as in speech not as in beer

Quando Stallman e altri personaggi rappresentativi del mondo open-source usano il termine free fanno riferimento a libertà e non a gratuito, visto il duplice significato del termine nella lingua inglese. Tra l'altro in una intervista rilasciata a un giornale italiano Stallman ha dimostrato di conoscere il significato delle due parole italiane che fanno riferimento a free (libero e gratuito per l'appunto) e le ha usate in modo appropriato per rispondere al giornalista (...probabilmente Stallman conosce la traduzione di free anche in molte altre lingue).

4 Open Source Definition

Il concetto di free software non è nuovo. Quando le università cominciarono ad adottare i computer, essi erano strumenti per la ricerca. Il software veniva scambiato liberamente e i programmatori venivano pagati per l'atto della programmazione, non per i programmi in sè. Solo più tardi, quando il mondo degli affari e del commercio adottò i computer, i programmatori cominciarono a mantenersi limitando i diritti d'uso del loro software e facendosi pagare per ogni copia. Il free

software come idea politica è stato reso popolare da Stallman, la Open Source Definition include molte delle idee di Stallman, e può ben considerarsi un derivato della sua opera.

4.1 Perchè è nata la Open Source Definition

Nella primavera del 1997 un gruppo di leader della comunità del free software si diedero appuntamento in California per trovare un modo di incoraggiare le idee legate alla distribuzione gratuita di software. Secondo il gruppo di lavoro (di cui facevano parte Eric Raymond e Tim O'Reilly) il messaggio anti-business di Stallman allontanava dall'apprezzare il potere del software distribuito gratuitamente.

Figura 2: Eric Raymond

La Open Source Definition offre maggiore libertà rispetto alla GPL. In particolare la Open Source Definition permette una maggiore promisquità fra software proprietario e open source.

La Open Source Definition cominciò la sua vita come un documento di linea di condotta della distribuzione Debian GNU/Linux. Debian, uno dei primi sistemi Linux, è costruito interamente con free software.

È importante capire che la Open Source Definition non è propriamente una licenza software e non è un documento di valore legale bensì una definizione.

5 L'Open Source funziona

Perché esiste e continua ad aumentare in numero e la qualità del software libero e perché i suoi utenti aumentano di più rispetto al trend di crescita attribuibile alla diffusione dei PC tra la gente? E inoltre, perché in molti casi il software open source funziona così bene da essere preferito ai corrispettivi commerciali?

Come si può distribuire un prodotto gratuitamente? I costi della copia di un'informazione come un programma software è infinitesimo. L'elettricità non costa quasi nulla, l'uso dell'attrezzatura poco di più. È come, per fare un paragone,

se si duplicasse una pagnotta usando un solo grammo di farina. L'economia dell'informazione è sostanzialemte diversa da quella degli altri prodotti. Ok, ma il lavoro di sviluppo qualcuno lo ha fatto e ha impiegato del tempo, perché qualcuno dovrebbe decidere di lavorare per sviluppare software per poi distribuirlo gratuitamente? Una ragione potrebbe essere perché gli serve e desidera qualcosa che veramente risponda alle proprie esigenze e in molti campi si riscontra come una cosa sviluppata secondo le proprie esigenze risulti migliore di una fatta perché, ad esempio, qualcuno ha pagato per farla. Succede in tutte le cose, è nella natura umana, quando si fa qualcosa perché nessuno te l'ha chiesto lo fai meglio e più volentieri.

Perché condividere il lavoro? Perché è il primo desiderio di un programmatore trovare altre persone interessate al proprio software. Se si distribuisce oltre all'eseguibile anche il sorgente si otterrà velocemente un miglioramento del software perché verranno proposte migliorie e modifiche in base alle esigenze reali e non in simulazioni di utilizzo di un gruppo ristretto di persone.

Per quanto ben progettato, un software non è mai testato abbastanza, avere a disposizione una comunità che prova e segnala i bug è un gran vantaggio per migliorare il proprio software. Qualcuno dice che dati 1000 occhi tutti i bug si annullano: se è disponibile il codice sorgente questo succede, qualcuno può trovare una miglioria da introdurre che va a beneficio di tutti gli utilizzatori.

Ci sono inoltre casi di progetti abbandonati e che dopo alcuni anni sono stati ripresi da altre persone grazie al fatto che il sorgente era disponibile, è questo per esempio il caso di GIMP il programma GNU per la manipolazione delle immagini molto potente.

Il software distribuito con sorgente o sotto forma di sorgente è in definitiva adattabile alle proprie esigenze ma ha anche un altro grosso vantaggio, quello di essere verificabile e quindi sicuro. È praticamente impossibile che un software open-source contenga parti di codice "cattive" tipo back-doors o disfunzionalità programmate e senza che nessuno se ne accorga.

Un altro aspetto interessante sono i buchi di sicurezza del programma verso attacchi esterni, questo riguarda soprattutto i servizi server. Nei software liberi le migliorie avvengono con una rapidità molto superiore a rispettivi software proprietari perché molte più persone hanno modo di trovare una soluzione sul campo in tempo reale, senza aspettare la patch che il produttore del software metterà a disposizione (se e quando deciderà di farlo).

Il problema virus sotto Linux non è risolto con gli antiviurs (che più che una soluzione sono una toppa) ma rendendo il sistema più sicuro e chiudendo le falle che potrebbero essere utilizzate per scopi non voluti, per questo non esistono antivirus per Linux: il problema è risolto a monte.

5.1 Linux

Eccoci finalmente a parlare di Linux, ma per introdurlo dobbiamo ancora una volta aprire una parentesi per parlare di un altro progetto (Hurd) che avrebbe dovuto essere e rappresentare quello che oggi è Linux ma che ha un'enorme importanza pur di fronte a come si sono sviluppati gli eventi.

Linus Torvalds avrebbe poi nel 1991 dato inizio al progetto Linux che accese l'esplosione del fenomeno Open Source.

Figura 3: Linus Torvalds

5.1.1 GNU-Hurd

Nel 1990 il sistema GNU era quasi completo, l'unica parte significativa ancora mancante era il kernel. Hurd che significa mandria di gnu (herd of gnus) doveva svolgere le funzioni del kernel Unix.

Senza entrare nei dettagli tecnici quello che successe è che lo sviluppo di Hurd ritardò per una serie di ragioni e per rendere Hurd robusto furono così necessari molti anni. Per fortuna era disponibile un altro kernel: nel 1991 Linus Torvalds sviluppò un kernel compatibile con Unix che venne chiamato 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.

In una mail scritta al newsgroup comp.os.minix il 29 gennaio 1992, per rispondere a un lungo dibattito intitolato "Linux è obsoleto", Linus Benedict Torvalds scrive: "Se il kernel GNU fosse stato pronto la scorsa primavera, non me ne sarebbe nemmeno importato di iniziare il mio progetto: il fatto è che non era, e non è, così . . . "

Non vorrei ridurre la nascita di Linux a una questione di tempo, ma questo fattore ha avuto la sua notevole importanza.

Non era scontato che Linux diventasse quello che è. Non era scontato per Linus, ne per Stallman, ne per i fondatori di RedHat, ne tantomeno per IBM che oggi dimostra un forte interesse e che decide di investire su Linux.

5.2 Il vantaggio di Linux

Linux si trova oggi ad avere milioni di utenti, migliaia di sviluppatori e un mercato in espansione. È presente in sistemi integrati, è usato per il controllo di dispositivi robotizzati e ha volato a bordo dello Shuttle. Linux ha avuto successo non tanto grazie al suo intento originario - essere largamente portabile e disponibile - quanto perché basato su solidi principi di progettazione e su un solido modello di sviluppo. La portabilità e la disponibilità risultano da queste robuste fondamenta.

Grossi progetti girano sotto Linux. Ad esempio il motore di ricerca Google: una piattaforma basata interamente su Linux che uniscie le risorse di 9000 server.

All'inizio Linux era indirizzato a un'unica architettura: l'Intel 386. Oggi Linux gira su qualunque cosa, dai PalmPilot alle workstation Alpha, ed è il sistema operativo più soggetto a porting fra quelli disponibili per PC (e non solo, ad esempio si lavora per il porting su PlayStation). Se dunque si scrive un programma che gira su Linux, per un'ampia gamma di macchine, quel programma sarà effettivamente "scritto una volta ed eseguito dappertutto".

6 Guadagnare con software gratuito

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

La vendita di servizi relativi al software libero è ammessa e incoraggiata: come insegnare argomenti quali la programmazione, le personalizzazione dei programmi oppure sviluppare nuovo software.

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.

Per capire come guadagnano le aziende che distribuiscono software gratuito, basti pensare a come viene prodotto un vocabolario. Non si tratta di inventare nuove parole, le parole esistono e tutti le possono usare, ma il lavoro di metterle assieme, ordinarle, dare una definizione che renda la parola fruibile a chi non la conosce, tutto questo è un lavoro che può essere pagato. Nessuno pensa che pagando il vocabolario si paga per usare l'Italiano (o un'altra lingua), è solo uno strumento

per usare meglio quello che è già a disposizione di tutti. Quello che aziende come RedHat, SuSE, Mandrake e molte altre fanno è approssimativamente assomiglia a questo: rendere il software gratuito usabile.

Figura 4: Bob Young (RedHat)

6.1 Le distribuzioni

Esistono veramente molte distribuzioni di Linux, ma quelle più utilizzate sono all'incirca una decina. Le più conosciute sono: Debian, Mandrake, Turbolinux, RedHat, Slackware, SuSE, Caldera. Si differenziano principalmente nella procedura di installazione (più o meno user-frendly, per la scelta del software, per i tool di configurazione, per la scelta dell'ambiente grafico preferito (quasi sempre Gnome o KDE), per il sistema di pacchettazione e installazione del software.