

Filosofia e sviluppo

Dopo aver visto la storia che ha portato all'attuale mondo del software, parliamo delle filosofie che ne regolano la diffusione e lo sviluppo.

Per capire in modo semplice la differenza tra software libero e software proprietario procediamo con un piccolo esempio:



Richard M. Stallman e il MIT:

- ▶ Faceva parte di un gruppo che condivideva liberamente le informazioni
- ▶ Sviluppavano insieme ITS



Lui sostiene che non condividere le proprie scoperte ed i propri risultati significa essere persone cattive

Tutto cambiò quando vennero introdotte nuove macchine che non permettevano a Stallman di fare ciò che aveva sempre fatto.

Decise quindi di programmare un sistema operativo e di metterlo a disposizione di chiunque lo volesse, liberamente.

Il progetto GNU:

- ▶ Significa “GNU is Not Unix”.
- ▶ Punta a ricreare un sistema operativo compatibile con Unix ma completamente libero.
- ▶ Il progetto parte nel 1984.
- ▶ Lo sviluppo viene condotto a partire dagli applicativi più importanti, come l'editor ed il compilatore.
- ▶ Già pochi anni dopo i software GNU venivano abbondantemente usati in ambiente Unix.

Stallman chiamò il suo software Free Software, per indicare che era “Software Libero”.

Free Software è un nome ambiguo, dato che inglese “Free” significa sia “libero” che “gratuito”.

Per spiegare il concetto viene fatto l'esempio:

“Free as free speech, not as free beer”

Per garantire che un software sia libero devono essere assicurati quattro libertà fondamentali.

1. La libertà di eseguire il programma, per qualsiasi scopo.

Imporre restrizioni sull'uso del Software Libero, in termini di tempo ("periodo di prova di 30 giorni", "la licenza scade il 1 Gennaio 2004") o di scopo ("il permesso è accordato per usi di ricerca o non commerciali"), o limitazioni arbitrarie di area geografica ("non può essere usato nel paese X") rende un programma non libero.

2. La libertà di studiare come funziona il programma e adattarlo alle proprie necessità.

Anche imporre restrizioni di fatto o di diritto sulla comprensione o la modifica di un programma, ad esempio richiedendo l'acquisto di licenze speciali o la firma di un "Non-Disclosure-Agreement" (NDA) o, per i linguaggi di programmazione che sono rappresentabili in più forme, vietando l'accesso al mezzo più naturale per comprendere o modificare un programma ("codice sorgente"), lo rende proprietario (non libero). Senza la libertà di modificare un programma, la gente sarebbe alla mercè di un singolo fornitore.

3. La libertà di ridistribuire copie in modo da aiutare il prossimo.

Il software può essere copiato e distribuito praticamente senza costi: se non si ha il permesso di dare un programma a qualcuno che ne ha bisogno (anche dietro pagamento, se lo si vuole), il programma non è libero.

4. La libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio.

Nessuno è un bravo programmatore in tutti i campi, qualcuno non sa programmare del tutto. Questa libertà permette a chi non ha il tempo o le capacità per risolvere un problema di accedere indirettamente alla libertà di modifica. Anche questo può avvenire dietro un compenso.

Per garantire queste 4 libertà è stata creata una licenza apposita:
La GNU GPL (General Public License)

- ▶ Il software definito “libero” viene rilasciato quasi tutto con questa licenza.
- ▶ Si basa sul principio di copyright: il software è di proprietà di chi l'ha creato, e lui permette agli altri di compiere le azioni previste dalla licenza.
- ▶ La licenza è trasferita con il software: il che significa che la modifica di un programma soggetto a GPL dovrà essere ancora GPL.

Il software libero è diverso dal software di pubblico dominio. Quest'ultimo è proprietarizzabile dalle aziende, mentre la GPL garantisce a tutti di fruire delle medesime libertà.

Esistono due modelli di sviluppo del software principali:

- ▶ **Modello a codice aperto (o open source)**

Si basa sul lavoro cooperativo, e consiste nel mettere a disposizione di più persone possibili il codice del prodotto. Questi, usando il programma, riferiranno dei bug, o se ne sono capaci, suggeriranno patch o migliorie.

- ▶ **Modello a codice chiuso (o closed source)**

Si basa sulla pianificazione dei piani aziendali, con team ristretti e gestiti gerarchicamente. Il lavoro viene pianificato a medio/lungo termine, e la ricerca dei bug avviene in modo deterministico.

- ▶ Maggiore sicurezza: il codice sorgente è disponibile a tutti, e quindi molte persone possono analizzarlo per trovarne i difetti in pochissimo tempo. Non esistendo inoltre una logica di rilascio, questi saranno subito disponibili.
- ▶ Sviluppo più rapido: il grande numero di persone che possono contribuire permettono rapide modifiche e miglioramenti.
- ▶ Maggiore scelta: non si è mai dipendenti da un prodotto specifico perchè la grande varietà dello sviluppo permette di avere molti programmi diversi per la stessa funzione, e questo permette di scegliere quello che si preferisce.
- ▶ Ciclo di vita naturale del software: fintanto che un software desta interesse il suo sviluppo sarà rapido, permettendogli di crescere in poco tempo. Se per qualche ragione diventasse obsoleto, verrebbe gradualmente abbandonato, in favore di soluzioni migliori.

- ▶ Dispersività dei progetti: molta gente lavora su software diversi senza un autorità, creando quindi la possibilità che molti progetti si dividano per questioni personali o che se ne creino di paralleli e non realmente necessari, disperdendo così preziose risorse che potrebbero essere usate per raggiungere maggiori risultati.
- ▶ Codice non necessariamente di qualità: molto spesso il codice viene modificato da programmatori un po' troppo artigianali, con il risultato di creare programmi difficilmente sviluppabili e comunque di bassa qualità, come per esempio XMMS.
- ▶ Soluzioni software eterogenee: il rapido e relativamente incontrollato sviluppo può provocare casi di mancata retrocompatibilità o incompatibilità tra software diversi. Tuttavia per ovviare a questo spesso le maggiori distribuzioni mettono a disposizione più release dello stesso software, in modo da limitare simili inconvenienti.

Nell'intervento di Francesco vedremo chiaramente il software di cui si è parlato finora, e di come sono evidenti i vantaggi visti in questa presentazione.

FINE,
GRAZIE