# Appunti
di
# Tecnologie Open Source

Luca De Franceschi

Università degli studi di Padova

# Indice

**1 Introduzione** **2**
1.1 Informazioni tecniche . . . . . 2
1.2 Programma del corso . . . . . 2
1.3 Materiale . . . . . 2
1.4 Introduzione al software libero . . . . . 3
1.5 Libero != Gratuito . . . . . 4
1.6 L'importanza del software libero . . . . . 4
1.7 Introduzione a GPL . . . . . 4

# 1 Introduzione

Questo corso si divide fondamentalmente in due parti:

1. Una prima parte **teorica** in cui si studierà il software libero, le licenze software, il progetto GNU, ...
2. Una seconda parte **pratica** che si baserà su diverse tecnologie e in cui verrà enfatizzato l'aspetto pratico.

## 1.1 Informazioni tecniche

Sito web del corso:

`http://www.math.unipd.it/~tapparo/TOS/`

Indirizzo email del docente:

`tapparo@math.unipd.it` [*attivo solo durante il periodo del corso*]

Le lezioni si terranno in **Aula 1BC50**
Gli orari sono i seguenti:

- **Giovedì**, 9:30 - 12:05
- **Venerdì**, 9:30 - 12:05

Il ricevimento avverrà durante gli intervalli, su appuntamento e dopo lezione.
**48 ore** di lezione, **6 CFU**, tutte lezioni frontali.
La modalità d'esame è solamente **orale**, e avviene tramite iscrizione su *Uniweb*. Verterà in due parti: la prima parte è una discussione sugli argomenti affrontati a lezione, la seconda è una parte pratica sulle tecnologie libere affrontate lungo il corso.

## 1.2 Programma del corso

- Storia del software libero;
- Licenze libere e caratteristiche del software libero;
- Problemi aperti e prospettive del software libero;
- Strumenti liberi di supporto allo sviluppo e alla cooperazione.

## 1.3 Materiale

Appunti delle lezioni. [**PRINCIPALE**]
Materiale nelle **slides**.
Molti libri di riferimento si possono trovare nelle slides. Molti sono reperibili liberamente.
Libri:

- **Open Source: strategie, organizzazione**, è il più accademico e viene toccato marginalmente dal corso. Offre buoni spunti per quanto riguarda la gestione manageriale del software;
- **Il software libero in Italia**, un libro molto interessante composto da diversi interventi. Contiene una buona sezione riguardante le licenze;
- **Hackers: Heroes of the Computers**, libro leggibile come un romanzo, molto consigliato, molto leggero ma va ben oltre gli obiettivi del corso;
- **Software libero, pensiero libero**, per chi ha poca dimestichezza con il progetto GNU. Anche questo libro è composto da una serie di interventi. *Stallman* ha una grande capacità di organizzazione degli argomenti;
- **Free culture**, di *Lawrence Lessig*. Quest'ultimo, oltre a essere l'ex leader di *Creative Common*, ha scritto una serie di libri in cui affronta le tematiche di libertà di accesso ai **contenuti**. È un libro scritto molto bene, affronta il problema della rivisitazione dei modelli di proprietà a fronte di forti cambiamenti (es. l'introduzione di Internet, l'introduzione dei voli aerei...);
- **Privilege and property**, accessibile da Internet. Viene affrontata la nascita del copyright.

## 1.4 Introduzione al software libero

Il software libero non ha nulla a che vedere con il *prezzo*, ma è un software che rispetta **4 concetti fondamentali**, ovvero 4 libertà per l'utente:

- Libertà di **usare** il software, usandolo senza restrizioni. Es. libertà di prendere il prodotto ed utilizzarlo senza limiti di tempo, senza vincoli di paese e per *qualunque scopo* (didattico, lavorativo, privato, ...). Il tipo di utilizzo non è mai precluso. Questa è da un lato la libertà meno importante, ma dal punto di vista dell'impatto sull'utente sviluppatore non è la maggiore;
- Libertà di **studiare** il software. Posso vedere il *codice sorgente* e capire come funziona. Senza questa libertà si ha un blocco della conoscenza ed è una forte limitazione alla crescita del prodotto;
- Libertà di **modifica**, ovvero posso prendere il software e cambiarlo, costruire nuove soluzioni. Il software libero è visto in questo contesto come *piattaforma*. Si tratta di costruire delle proprie versioni a partire da una base. È una libertà molto importante ed è una ricchezza per poi creare altri sviluppi;
- Libertà di **ridistribuzione**, in questo caso le aziende non solo possono creare per se stesse, ma anche mettere la nuova versione a disposizione di altri. Software libero come bene comune (*Routes*, *Beowulf*, *nslu2*). Una volta che posso ridistribuire ad altri allora il mio lavoro diventa realmente usabile.

## 1.5 Libero != Gratuito

È un errore comune confondere questi due concetti, ma essi sono realmente due cose distinte. Esiste software gratuito ma che non è libero ed esiste software libero non gratuito (*openerp*, i programmi della *fsf*, i binari *RedHat Enterprise Linux*). C'è tutta una serie di software **freeware** o **shareware** (es. *winzip*). Un software shareware è collegato ad un acquisto successivo, invita l'utente ad acquistare una versione commerciale.
**Free Software Foundation** distribuisce software libero disponibile per diverse architetture. Ha una serie di programmi non facili da compilare. Si scarica il sorgente e si tenta di compilarlo, oppure si richiede il CD con i file già compilati, e questo CD viene fatto pagare.
**RedHat Enterprise** sistema i bugs previo pagamento e questi vengono risolti molto presto. Loro distribuiscono i sorgenti ma non i binari.
I concetti di **libero** e **gratuito** sono dunque concetti ortogonali.

## 1.6 L'importanza del software libero

L'importanza del software libero è legata in primo luogo alla **riduzione dei costi** (apache, php, ...). Non è importante per il pagamento in sé ma in quanto mobilita le leggi del mercato. È un mercato aperto, in cui è facile entrare e investire.
Un secondo impatto riguarda la **trasparenza**. Se quello che faccio è visibile allora è anche controllabile, quindi gli sviluppatori "*non fanno troppo i furbi*". È difficile fidarsi di un software che non si sa bene cosa faccia. Il software libero è una **garanzia**.
Inoltre con il software libero non abbiamo nessun **lock-in**, il software libero si può adattare facilmente alle versioni precedenti e quindi non vi sono dipendenze (l'esempio perfetto è *Microsoft Word*).
**Sicurezza e affidabilità?** Non ci sono dimostrazioni effettive che questo sia vero. Da un lato il software libero è visibile a tutti, ma dall'altro la manutenzione è costosa ed è facile lasciare bug ovunque. Il software proprietario vive molto spesso di un inflazione di *features*, questo per aumentare le vendite.
Si passa da un modello in cui c'è [**chi fa**] e [**chi usa**], chi ha il potere derivato dalla conoscenza e chi non lo ha e sta alle condizioni. Il software libero è conoscenza libera. Si cambia il modo in cui si sviluppa e con cui si fa impero. È un modello "*social*", il software vale molto di più per il fatto che c'è una **comunità** che gli gira intorno. Il rapporto che si viene a creare con gli utenti è molto importante (*Innovation happens elsewhere*). Intorno al software libero si può creare una comunità in modo che la somma dei costi per fare un prodotto è minore rispetto al costo della comunità stessa.

## 1.7 Introduzione a GPL

Per molto tempo il software libero ha avuto una **posizione di inferiorità**. Le aziende prendevano software libero, sviluppavano una nuova versione e le rilasciavano come software proprietario. Il progetto **GNU** voleva creare una versione completamente libera del software. Ha creato dunque una nuova licenza, chiamata **GPL** (General Public License), in modo che avesse un

*effetto virale*. Una licenza libera ma con una restrizione particolare, ovvero ogni redistribuzione deve essere rilasciata sotto licenza GPL (circo virtuoso). Vedere il software libero come un'enorme libreria di conoscenza sempre disponibile.
Il software libero con questa licenza si arricchisce molto, cresce nel tempo e diventa sempre più interessante. Questa licenza è ancora molto presente (60%, 70% del software libero è sotto GPL).