# Programming Languages mini-HOWTO

Risto S. Varanka 22 luglio 2000

Un breve confronto tra i principali linguaggi di programmazione per Linux e le principali librerie per creare interfacce grafiche (GUIs) in ambiente Linux Traduzione a cura di Andrea Giancola Revisione a cura di Giulio Daprelà

## Contents



# 1 Introduzione

Linux è un sistema operativo affascinante, perché permette ad ogni utente di prendere parte al suo sviluppo. La varietà di linguaggi disponibili, però, può confondere i programmatori che sono agli inizi. Questo documento elenca le scelte più comuni per gli sviluppi di tutti i giorni e fornisce alcune informazioni chiave su di essi. (Beh, “più comuni ” e “chiave” così come li percepisco io.)

Il mio scopo non è né di recensire i linguaggi nd́i determinare quale sia il migliore. Ogni linguaggio è uno strumento che si adatta bene ad alcune funzioni e ad alcuni gusti. Si possono ottenere ulteriori informazioni (spesso contraddittorie) facilmente, se si chiede in giro o si tengono le orecchie aperte. La sezione Links in questo documento darà alcuni puntatori per fare ricerche per conto proprio.

C’è una pletora di linguaggi e librerie per Linux, perciòquesto documento copre solo i linguaggi e i toolkit per GUI (Graphical User Interface) più comuni in questo momento. Questo documento vuole essere piuttosto neutrale, ma non ho incluso tutti i linguaggi disponibili. Poiché il mio giudizio è indubbiamente per molti versi distorto, consiglio agli sviluppatori seri di controllare i siti che elencano in modo più esaustivo tutti i linguaggi e le librerie disponibili. Da notare anche che sono coperte solo le implementazioni per Linux dei linguaggi e dei toolkit per GUI citati, le loro caratteristiche su altre piattaforme non sono discusse o implicate.

Questo documento è una recente aggiunta al LDP, quindi non ci sono state opportunitè per feedback da parte della comunitè. Comunque viene rilasciato nella speranza che si mostri utile per chi è interessato a programmare sotto Linux, specialmente i principianti. Un punto di domanda nelle tabelle indica mancanza di informazioni. Se si è in grado di riempirlo, per favore si contatti l’autore.

## 1.1 Ultima versione del documento

Si possono trovare gli ultimi aggiornamenti presso http://www.helsinki.fi/~rvaranka/Computer/Linux/HOWTO/

## 1.2 Copyright



## 1.3 License



### 1.3.1 Requirements of Modified Works

## 1.4 Disclaimer



## 1.5 Autore

Siete incoraggiati a mandare osservazioni all'autore presso: risto.varanka@helsinki.fi .

Il sito web dell'autore si può trovare a http://www.helsinki.fi/˜rvaranka/ .

## 1.6 Credits

Sono grato a diverse persone che hanno espresso commenti su problematiche dei linguaggi di programmazione. Queste conversazioni mi hanno aiutato ad ottenere una migliore comprensione dei differenti linguaggi, e spero che future conversazioni consentiranno a questo mini-HOWTO di maturare nel tempo. Vorrei in particolare ringraziare le persone del canale IRCNet #linux: Morphy, Bluesmurf, Vadim, Zonkˆ, Rikkus ed altri i cui nomi ho dimenticato.Ringraziamenti vanno anche a Stig Erik Sandoe per i suoi utili commenti.

### 1.7 Collegamenti

Elenchi esaustivi di librerie di sviluppo e tool per Linux :

- Freshmeat
- Linux Development Tools
- linuxprogramming.com

Le Hacker FAQ di Eric S. Raymond sono un altro testo interessante per sviluppatori Linux alle prime armi. Si concentrano su alcuni aspetti culturali e psicologici dello sviluppo open source.

Altri documenti LDP

che coprono argomenti generali di programmazione comprendono la Reading List HOWTO e la Linux Programmer's Guide - diversi altri sono stati scritti su argomenti specifici.

## 2 Linguaggi di programmazione

C, Lisp e Perl sono i linguaggi tradizionalmente usati per l'hacking nella cultura GNU/Linux; Python, PHP, Java e C++ hanno guadagnato terreno di recente.

### 2.1 Concetti della tabella

**Linguaggio**

Un nome comune del linguaggio.

**Principianti**

Indica quanto un linguaggio è adatto per persone con poca esperienza di programmazione. Un linguaggio marcato con "sì" dovrebbe essere adatto come primo linguaggio di programmazione per un principiante.

**Performance**

Quanto veloci le applicazioni saranno quando verranno messe in un ambiente di produzione. Le prestazioni dipendono più dalle capacità di programmazione algoritmica che dal linguaggio utilizzato. A lume di naso C, C++ e Fortran sono talvolta necessari perché possono offrire migliori prestazioni di altri linguaggi - altre volte possono essere poco maneggevoli per lo scopo prefisso. (Una idea per un "benchmarking" empirico dei linguaggi potrebbe essere implementare un semplice algoritmo di ricerca in tutti quanti e confrontare i tempi di esecuzione. Questo naturalmente non misura le prestazioni del linguaggio in sé - in quanto un tale concetto non ha significato - ma solo dell'implementazione. Naturalmente non é nemmeno un metodo molto affidabile o completo, ma darebbe un'idea di quanto i tempi di esecuzione possono differire tra linguaggi diversi. Nessuno vuol aiutarmi in questa cosa?)

**OOP, Object-Oriented Programming vs. altri paradigmi**

La programmazione orientata gli oggetti è un importante paradigma di programmazione che sta guadagnando popolarità. Nella programmazione orientata agli oggetti le strutture dati e gli algoritmi sono integrati in unità, spesso chiamate classi. L'OOP è spesso contrapposta alla programmazione procedurale (che separa algoritmi e strutture dati). La questione non è strettamente dipendente dal linguaggio usato: si può fare OOP in un linguaggio non elencato sotto tale categoria (per esempio il C), e programmare in stile procedurale in linguaggi elencati come OOP. Ho elencato come OOP i linguaggi che hanno speciali caratteristiche od estensioni per facilitare l'OOP. I linguaggi funzionali (il Lisp per esempio) sono un tipo un po' diverso - tra le altre cose, la programmazione funzionale è un superinsieme dell'OOP. La programmazione logica (Prolog), anche detta programmazione dichiarativa, d'altra parte, non è collegata alle altre categorie di programmazione in un senso simile.

**RAD, Rapid Application Development**

Dipende più dagli strumenti che si stanno utilizzanfo che dal linguaggio in sé. C'è un HOWTO sugli strumenti di sviluppo GUI per Linux, anche se non è aggiornato. Con un buon strumento grafico si può fare RAD. Il RAD può essere potente quando si basa anche sul riuso di codice, perciò il software libero potrebbe fornire un buon punto di partenza.

**Esempi**

Sono gli ambiti della programmazione in cui il linguaggio è più utilizzato. Possono esistere altri buoni (e cattivi) casi di uso, ma sono meno caratteristici.

**Commenti**

Informazioni aggiuntive sul linguaggio, come capacità e dialetti.

## 2.2 Linguaggi principali

```
Perl
Principianti: Sì - OOP: Sì
Esempi: Scripting, amministrazione di sistema, www
Commenti: Potente per gestire testi e stringhe

Python
Principianti: Sì - OOP: Sì
Esempi: Scripting, scripting di applicazioni, www
Commenti:

TCL
Principianti: Sì - OOP: No
Esempi: Scripting, amministrazione di sistema, applicazioni
Commenti:

PHP
```

```
Principianti: Sì - OOP: Sì
Esempi: Www
Commenti: Popolare per database web

Java
Principianti: Sì - OOP: Sì
Esempi: Applicazioni cross-platform, www
Commenti: Si sta diffondendo in nuove aree, ad es. infrastrutture di e-commerce

Lisp
Principianti: Sì - OOP: funzionale
Esempi: modi di Emacs (per Elisp), intelligenza artificiale
Commenti: Varianti Elisp, Clisp e Scheme

Fortran
Principianti: No  - OOP: No
Esempi: applicazioni matematiche (scientifiche)
Commenti: Varianti f77 e f90/95

C
Principianti: No  - OOP: No
Esempi: programmazione di sistema, applicazioni
Commenti:

C++
Principianti: No  - OOP: Sì
Esempi: Applicazioni
Commenti:
```

## 2.3 Programmazione di shell

Anche le shell sono un importante ambiente di programmazione. Non le ho trattate qui perché non conosco ancora l'argomento molto a fondo. La conoscenza delle shell è importante per chiunque lavori in Linux regolarmente, ed ancor più per gli amministratori di sistema. Ci sono analogie tra la programmazione di shell e gli altri tipi di scripting - spesso raggiungono gli stessi obiettivi, e si ha la possibilità di scegliere tra una shell nativa ed un linguaggio di scripting a sé stante. Tra le shell più popolari vi sono bash, tcsh, csh, ksh e zsh. Si possono ottenere informazioni di base sulla propria shell con il comando *man*, ad esempio *man bash*.

## 2.4 Altri linguaggi

Altri linguaggi degni di nota: AWK, SED, Smalltalk, Eiffel, Ada, Prolog, assembler, Objective C, Logo, Pascal (convertitore p2c)

### 2.5 Link

- Un sito di informazioni generali sui linguaggi di programmazione, molte informazioni ed opinioni
- TCL
- Perl
- Python
- PHP
- Java
- clisp

## 3 Toolkit per GUI

Il sottosistema grafico standard per UNIX e Linux, chiamato X, ha le proprie librerie per lo sviluppo di GUI. Forniscono una interfaccia programmativa di basso livello verso X, ma tendono a essere difficili da usare. Vecchie applicazioni verso l'utente finale ed altri toolkit naturalmente ne fanno largo uso. Oggigiorno la scena delle GUI per Linux é dominata da GTK+ e Qt, poichè due popolari e completi ambienti operativi (GNOME e KDE) sono basati su di essi.

### 3.1 Concetti nella tabella

**Libreria**

Nome comune o abbreviazione del toolkit

**Principianti**

Indica se il toolkit è adatto per un programmatore agli inizi.

**Licenza**

Il fatto che toolkit differenti adottino licenze differenti ha una certa importanza pratica. Le licenze GTK+, TK e GNUstep consentono di sviluppare applicazioni sia open-source che closed-source senza pagare per una licenza. La licenza Motif richiede un pagamento, mentre la licenza QT richiede pagamento solo se si scrivono programmi closed-source.

**Linguaggio**

Il linguaggio che viene più spesso usato con il toolkit.

**Bindings**

Altri linguaggi che possono utilizzare il toolkit.

**Esempi**

Applicazioni che fanno uso del toolkit.

| Libreria | Principianti | Licenza | Linguaggio | Bindings | Commenti |
|---|---|---|---|---|---|
| TK | Sì | Free | TCL | Perl, Python, al | |
| GTK+ | No | Free (LGPL) | C | Perl, C++, Pyt | Molto popolare |
| QT | No | Free per sviluppi open source | C++ | Python, Perl, C | Molto popolare |
| Motif | No | A pagamento | C/C++ | Python, altri? | Lesstif è unsostituto free |
| GNUstep | No | Free (LGPL) | Objective C | Guile, Java? | GNUstep è ancora in corso di sviluppo |

**Commenti**

Informazioni addizionali sul toolkit.

## 3.2 Principali toolkit GUI

## 3.3 Link

- TK
- GTK+
- QT
- Motif
- GNUstep