Programming Languages mini−HOWTO

Table of Contents

Programming Languages mini−HOWTO

Risto S. Varanka

Jul 22nd 2000

A brief comparison of major programming languages for Linux and major libraries for creating graphical user interfaces (GUIs) under Linux

---

1.Introduction

Linux is a fascinating operating system because it lets any user participate in its development. The variety of available languages, however, can be confusing to beginning Linux developers. This document lists the most common options for everyday development and states some key facts about them. (Well, ``most common'' and ``key'' as I perceive them.)

My aim is neither to review the languages nor to determine which one is the best. Each language is a tool that fits some jobs and some tastes. You can get further (often conflicting) information easily, if you ask around or keep your ears open. The Links sections in this document will give you some pointers for your own research.

There is a plethora of languages and libraries for Linux, so this document only covers the most common languages and GUI (Graphical User Interface) toolkits at the moment. This document is intended to be fairly neutral, but I haven't included all languages available. Since my judgment is undoubtedly biased in many ways, I advise serious developers to check out the sites that do a better job in listing all languages and libraries. Also note that only the Linux implementations of the languages and GUI toolkits are covered, their features on other platforms are not discussed or implied.

This document is a recent addition to the LDP, so there has not been opportunity for much community feedback. However, it is released in hopes that it will prove useful for people interested in programming under Linux, especially beginners. A question mark in the tables indicates lack of information. If you can fill it in, please contact the author.

1.1 Latest Version of the Document

You can find the latest modifications at http://www.helsinki.fi/~rvaranka/Computer/Linux/HOWTO/

1.2 Copyright

1.3 License

1.4 Disclaimer

THIS DOCUMENT COVERS A LARGE AND CONSTANTLY CHANGING DOMAIN. THEREFORE, THE INFORMATION CONTAINED IN THIS DOCUMENT MAY BE INCORRECT OR OUTDATED. ALL USE OF THIS DOCUMENT AND ALL INFORMATION CONTAINED IN IT IS AT YOUR OWN RISK. THE AUTHOR DOES NOT GIVE ANY WARRANTY OR GUARANTEE, EITHER EXPLICIT OR IMPLIED.

1.5 Author

You are welcome to send feedback to the author at: risto.varanka@helsinki.fi.

Author's web site can be found at http://www.helsinki.fi/~rvaranka/.

1.6 Credits

I am thankful to several people who commented on language issues. These conversations have given me a better view of the different languages, and I hope future conversations will allow this mini−HOWTO to mature over time. Especially I would like to thank the people at the IRCNet channel #linux: Morphy, Bluesmurf, Vadim, Zonk^, Rikkus and others whose names I have forgotten. Thanks go also to Stig Erik Sandoe for helpful comments.

1.7 Links

Exhaustive lists of Linux development libraries and tools:

- Freshmeat
- Linux Development Tools
- linuxprogramming.com

The Hacker FAQ by Eric S. Raymond is another interesting text for novice Linux developers. It concentrates on some cultural and psychological aspects of open source development.

Other LDP documents covering general programming subjects include the Reading List HOWTO and the Linux Programmer's Guide − several more have been written on specific subjects.

2.Programming Languages

C, Lisp and Perl are traditional hacking languages in the GNU/Linux culture; Python, PHP, Java and C++ have gained new ground recently.

2.1 Concepts in the Table

Language

A common name of the language.

Beginner

Indicates how well suited the language is for people with little programming experience. A language marked with ``yes'' should be viable for a beginner's first programming language.

Performance

How fast your applications are likely to run when you put them into production use. Performance depends more on your algorithmic programming skills than the actual language. As a rule of thumb, C, C++ and Fortran are sometimes necessary because they can offer better performance than other languages − at other times they might be unwieldy for the desired purpose. (One idea for unscientific ``benchmarking'' of the languages would be to implement a simple sorting algorithm in all of them and compare running times. This of course does not measure the performance of the actual language − since that concept does not make sense − but only the implementation. Of course it's also not a very reliable or thorough method, but it would give an example how running times in different languages can differ. Anybody want to help me with this?)

OOP, Object−Oriented Programming vs. other paradigms

Object−oriented programming is an important programming paradigm that is gaining popularity. In object oriented programming, data structures and algorithms are integrated into units, often called classes. OOP is often contrasted with procedural programming (which uses separate algorithms and data structures). It is not strictly dependent on language: you can do OOP in languages not listed as such (C for example), and program in the procedural style in languages that are listed as OOP. I've listed as OOP languages that have special features or add−ons to facilitate OOP. Functional languages (Lisp for example) are a bit different breed − among other things, functional programming is a superset of OOP. Logic programming (Prolog), also called declarative programming, on the other hand, is not related to the other types of programming in a similar sense.

RAD, Rapid Application Development

More dependent on the tools you are using than the actual language. There is a HOWTO on GUI development tools for Linux, although it's out of date. With a good graphical tool you can do RAD. RAD can be powerful when based on code reuse as well, so free software could provide a good starting point.

Examples

Mentions fields of programming the language is most often used in. Other good (and bad) uses exist, but they are less typical.

Comments

Additional information on the language, like capacities and dialects.

2.2 Major Languages

```
Perl
Beginner: Yes - OOP: Yes
Examples: Scripting, sysadmin, www
Comments: Powerful for handling text and strings

Python
Beginner: Yes - OOP: Yes
Examples: Scripting, application scripting, www
Comments:

TCL
Beginner: Yes - OOP: No
Examples: Scripting, sysadmin, applications
Comments:
```

```
PHP
Beginner: Yes − OOP: Yes
Examples: Www
Comments: Popular for web databases

Java
Beginner: Yes − OOP: Yes
Examples: Cross−platform applications, www
Comments: Spreading to new areas, eg. e−commerce infrastructure

Lisp
Beginner: Yes − OOP: Functional
Examples: Emacs modes (for Elisp), AI
Comments: Variants Elisp, Clisp and Scheme

Fortran
Beginner: No  − OOP: No
Examples: Mathematical (scientific) applications
Comments: Variants f77 and f90/95

C
Beginner: No  − OOP: No
Examples: System programming, applications
Comments:

C++
Beginner: No  − OOP: Yes
Examples: Applications
Comments:
```

2.3 Shell Programming

Shells are an important programming environment, too. I haven't covered them because I don't understand the field very thoroughly yet. Knowledge of shells is important for anyone who works on Linux regularly, more so for system administrators. There are similarities between shell programming and other kinds of scripting − often they can achieve the same goals, and you have the option of choosing between native shell and a separate scripting language. Among the most popular shells are bash, tcsh, csh, ksh and zsh. You can get basic information on your shell with the *man* command, *man bash* for example.

2.4 Other Languages

Other languages of note: AWK, SED, Smalltalk, Eiffel, Ada, Prolog, assembler, Objective C, Logo, Pascal (p2c converter)

2.5 Links

- A general info site on programming languages, lots of info and opinions
- TCL
- Perl
- Python
- PHP
- Java

- clisp

3.GUI Toolkits

The standard graphical subsystem for UNIX and Linux, called X, has its own libraries for GUI development. They provide a low−level programming interface to X, but tend to be hard to use. Old end−user applications and other toolkits of course make good use of them. Nowadays the Linux GUI scene is dominated by GTK+ and Qt, since two popular, complete user environments − GNOME and KDE − are based on them.

3.1 Concepts in the Table

Library

Common name or abbreviation of the toolkit.

Beginner

Whether the toolkit is suitable for a newbie programmer.

License

Different licenses for different GUI toolkits have practical significance. GTK+, TK and GNUstep licenses allow you to develop both open source and closed source applications without paying for a license. Motif license requires payment, while the QT license requires payment only if you write closed source programs.

Language

The language that is most often used with the toolkit.

Bindings

Other languages which can use the toolkit.

Examples

Applications that use the toolkit.

Comments

Additional information on the toolkit.

3.2 Major GUI Toolkits

Library	Beginner	License	Language	Bindings	Examples	Comments
TK	Yes	Free	TCL	Perl, Python, others	make xconfig, TKDesk	
GTK+	No	Free (LGPL)	C	Perl, C++, Python, many others	GNOME, Gimp	Very popular
QT	No	Free for open source	C++	Python, Perl, C, others?	KDE	Very popular
Motif	No	Non−free	C/C++	Python, others?	Netscape, Wordperfect	Lesstif isa free replacement
GNUstep	No	Free (LGPL)	Objective C	Guile, Java?	None widely known, but see the application list	GNUstep is still under development

3.3 Links

- TK
- GTK+
- QT
- Motif
- GNUstep