LinuxFocus article number 350
http://linuxfocus.org

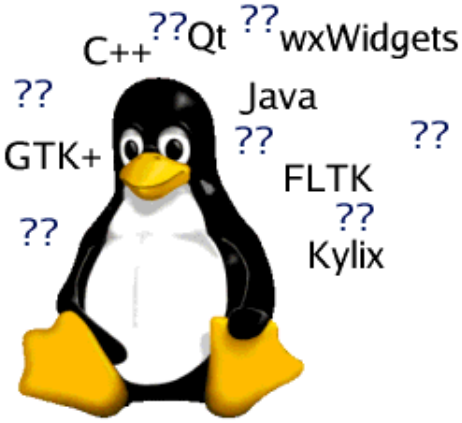
Platform Independent Software Develoment

by Michael Tschater
<tschater/at/web.de>

About the author:

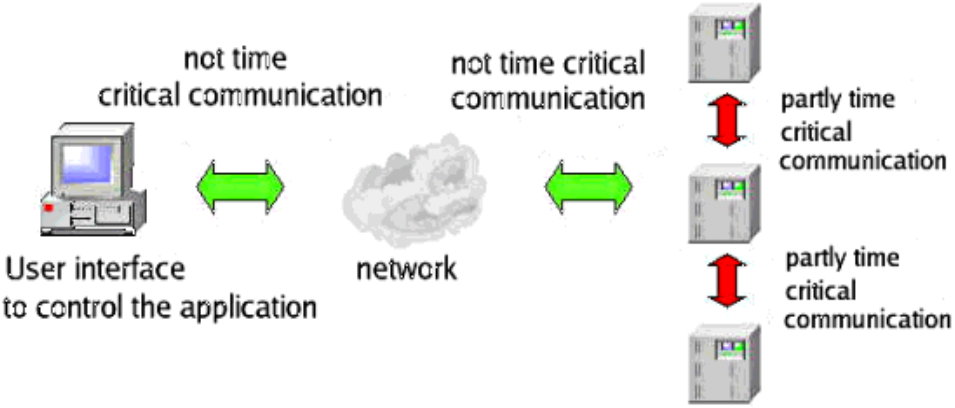
Michael is primarily busy with hardware related software development (firmware). For his current project an additional decision on the strategy of the development environment - to be utilized for the programming of the front-end of his firmware - has to be made.

Translated to English by:
Jürgen Pohl
<sept.sapins/at/verizon.net>



Abstract:

Almost all equipment used in industry may be controlled over a network. Th user interface runs on off-the-shelf hardware and works as a simple client, receiving and sending data, not time critcal (e.g. initializing parameters and measurement results). In the following diagram this is shown in green:



Software projects often require an answer to the question which operating systems shall be supported. While the readers of this magazine tend to lean toward Linux, also other operating systems (mostly Windows) are requested principle, the operating system to be used does not present a dominating issu the application; the user has to be able arrive at the results intuitively.
The following article shall demonstrate that a decision on a specific platform not required since it is feasable to write software which can be compiled for various operating systems. This article shall be limited to PCs with Linux an Windows. It should be possible to use the applications also on Mac and Mac but this could not be demostrated due to missing hardware.

With platform-independent libraries we differentiate between two approaches to produce controls in dialogs:

1. Native libraries: For the display of elements the corresponding routines of the operating system are utilized. This assures that all controls appear like the standard application of the operating system. A native library presents controls differently under Linux than under Windows 2000 or XP.
2. The second possibility is to program an appropriate look & feel, meaning all controls are to be displayed by the library and appear identical in all operating systems.

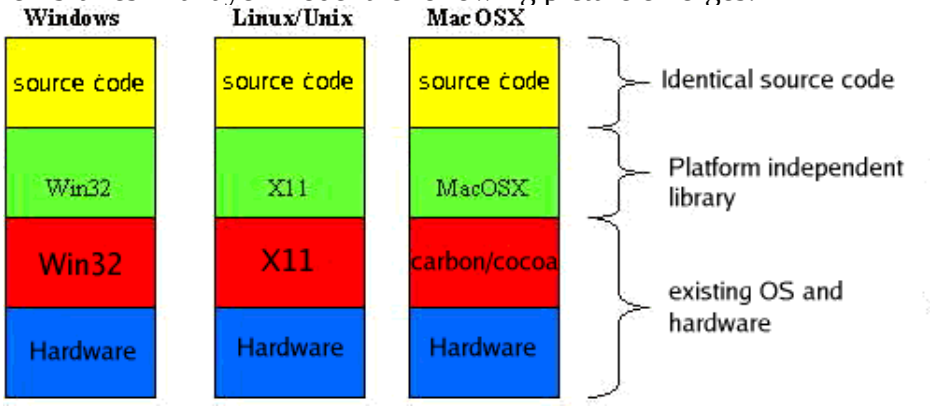
Besides the technical quaracteristics of the libraries additional operational factors play a role which shall be compared:

- Development environment: an integrated development environment (e.g. GUI builder, makefile generator) simplifies the software development.
- Documentation and support: immediate help in case of occurring problems.
- Costs: While most libraries are freely available for private use, for commercial applications sometimes costs inccur. For fundamental decisions on software projects such costs have to be justified to the decision makers.
- Actual costs for the porting between systems

In an actual case another issue has to be taken in consideration; this does, however, not apply to all projects:

- The software produced shall utilize native controls to integrate seemlessly into the existing system architecture. The user should not be able to identify differences between new and the existing software on the system.

By displaying the libraries in a layer-model the following picture emerges:



Programming Languages

The first criteron to be selected is the programming language. There are several choices which will be

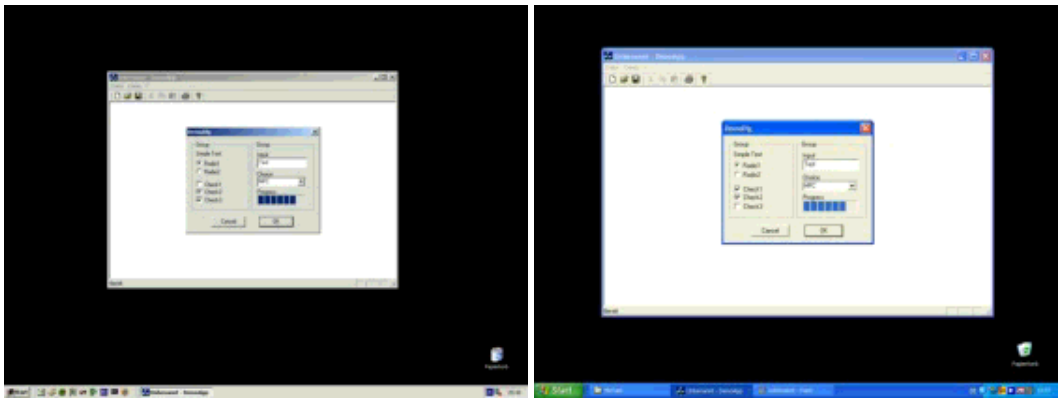
discussed below:

1. C/C++ libraries
2. Java
3. Kylix
4. Smalltalk
5. Mozilla

The alternatives to C and C++ will be explained more in detail since they are less established with software developers.

A Sample Application

To be able to compare the various software packages a sample application, using all libraries, shall be generated. The implementation of the application does not possess any functionality but it shows the most important controls. For the windows page pure Windows software will be created (Visual C++ 6.0, MFC Class-Library), the other packages will be compared to it in regard to look & feel. As linux distribution I will use RedHat Fedora Core 2 and Debian 3.0.



Windows 2000 and Windows XP Screenshot (source code for Visual C++ here (win32_src.zip)).

C/C++ Libraries

Trolltech Qt

Qt is a class library of the Norwegian company Trolltech for platform-independant programming with C++. The Linux window manager KDE is based on the Qt package. Originally, Qt was under a license which was not acceptable for many Linux users. For this reason the GTK+-library was developed, which is the basis for the Gnome window manager. In the meantime, the Linux-version, as well as the MacOS-version, is available under the GPL, including all source code. Qt for Windows is still available commercially. A time-limited test version for evaluations may be downloaded from the webpage - it will

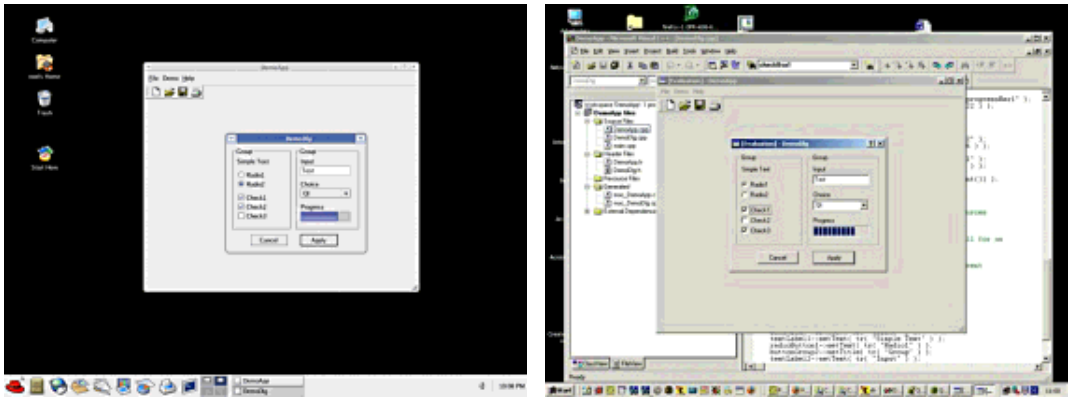
be differentiated between commercial application or academic use. In the following the commercial evaluation version will be explained. This version requires registration.

Besides the versions for Windows, Linux(Unix) and Mac an embedded version is available, it runs on embedded Linux variants and offers a leaner window administration.

The installation under Linux completes as expected without any problem. Included is the GUI generator Qt Designer. The detailed documentation example projects, quick start guide and class overview. Qt Designer generates as output an XML description of the GUI. Using the Qt-Tool qmake you can generate a valid Makefile from the XML description. This Makefile generates then from the GUI description C++ source code (Qt-Tool: uic) and calls the Meta Object Compiler (Qt-Tool: moc). The later translates Qt specific language extentions into C++ source code. After that a standard make procedure to compile the executable can be used.

The following sequence is needed to generate the source files manually (Input file is MyDialog.ui):

- uic MyDialog.ui > MyDialog.h
- uic -impl MyDialog.h MyDialog.ui > MyDialog.cpp
- moc -o moc_MyDialog.cpp MyDialog.h



Linux and Windows 2000 Screenshot (source code for QtDesigner here (qt_src.tar.gz)

Qt Overview

Name:	Trolltech Qt
Version:	3.3.2
Operating Systems:	Linux, Win32, MacOS, Solaris, IRIX, AIX, HP-UX
Programing language:	C++
License:	GPL or proprietary License (commercial)
Advantages:	• base library for KDE Windows Manager in Linux • installation packages in all standard distributions (installation very simple) • generic controls under Windows • mighty development environment(s) • proven • migrations support for Win32 MFC applications allows incremental conversion of MFC source code.
Disadvantages:	• possible license costs (expensive) • Evaluation software produces errors during the installation under windows
Development environment:	e.g. QtDesigner, KDevelop
WWW:	http://www.trolltech.com
Documentation:	manuals, tutorials, mailing lists e.g. http://doc.trolltech.com/3.3/index.html
Reference projects:	• KDE Desktop (Default e.g. with SuSE) • Opera Browser • Photoshop Album
Distribution:	very wide spread

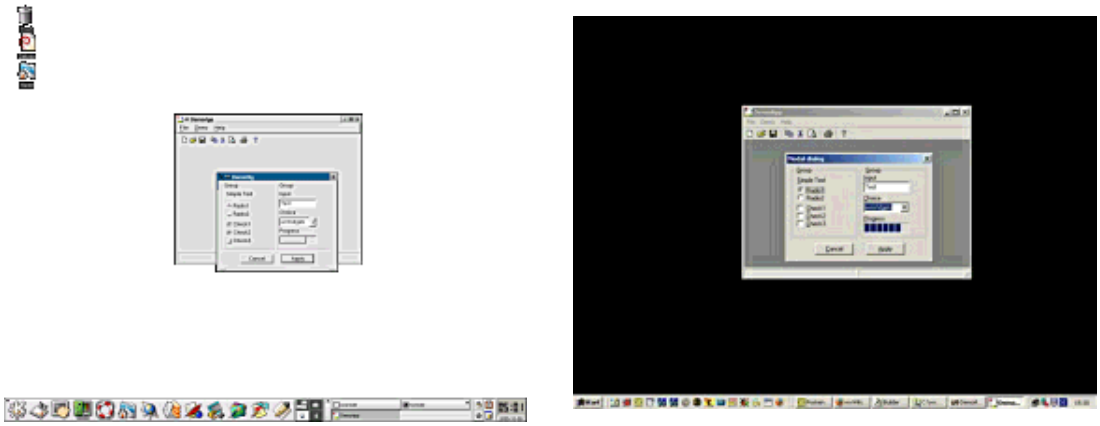
wxWidgets

The wxWidgets toolkit is available since some 12 years, but only a few month ago the package got its today's name. The name wxWindows, used until that time, was abandoned after "talks" with Microsoft. wxWidgets includes a huge collection of classes for all application areas. The list of reference applications demonstrates the maturity of the software package.

Programming is being done in C++, it is similar to Visual C++ under Windows.

A disadvantage is that you get with wxWindows2.4.2 under RedHat Fedora Core 2 errors when compiling the example programs. The cause are GTK+ calls which are declared private in the from RedHat patched GTK+ version. Calling those functions is therefore not allowed. However those are minor problems. Everything will run without problems when the standard GTK+ library is used. Under Debian everything worked right away.

The installation under Windows worked without problems.



Linux and Windows 2000 Screenshot (source code for here (wx_src.zip)).

Overview of wxWidgets

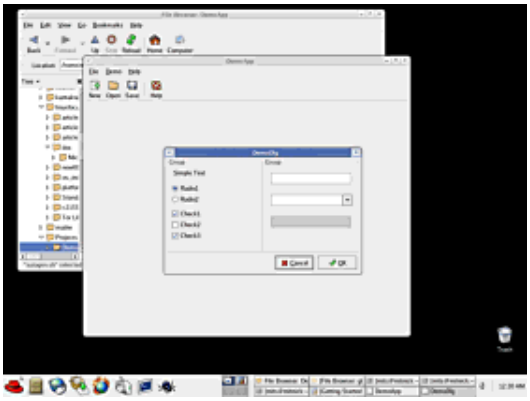
Name:	wxWidgets
Version:	2.4.2
Operating systems:	Linux, Win32, embedded devices
Programming Language:	C++
License:	LGPL
Advantages:	• simple handling (many examples). • very good documentation.
Disadvantages:	• Problems with the combination: Fedora Core 2 - wxWindows2.4.2
Development environment:	
WWW:	http://www.wxwidgets.org
Documentation:	manuals, tutorials, mailing lists, wiki e.g. http://wiki.wxwidgets.org
Reference projects:	AOL Communicator
Distribution:	not wide spread

GTK+ (with gtkmm)

The acronym stands for "The GIMP Toolkit". The two well-known projects are the Gnome Windows Manger - part of any Linux standard distribution - and the graphic application GIMP. Gnome is the second major desktop environment - besides KDE (see Qt) - under Linux. It is the default environment of many distributions . With the introduction of GTK+ version 2 the look & feel has been substantially improved.

One particularity of GTK+ is its complete implementation in C. Consequently, the GUI builder *glade2* produces C-code. By using *gtkmm* (formerly GTK--) programming can also be done in C++.

Contrary to the professional appearance of GTK+ for Linux, 'GTK+ for Win32' is not impressive. Clicking on the link on the GTK+-mainpage immediately results in the warning "**The program(s) might crash unexpectedly or behave otherwise strangely ".** (But of course, so do many commercial programs on Windows.) The stability seems to depend a lot on the machine, display drivers, other software installed or not present (status 2004-09-06). The courageous software developer clicks on the download page anyway and faces a long list of individual software components for downloading. For a comprehensive package, one searches in vain. Instead of that an instruction on how to install a number of software components can be read and to return to the download page if some specific components are missing. This matches the statement of the 'GTK+ for Windows' webpage: "You are expected to be quite experienced to be able to use GTK+ in your own programs. This isn't Visual Basic." After installing the initial components and an unsuccessful attempt to start one of the sample applications, most of the developers may have lost their desire to get deeper into it. The very unprofessional presentation of the 'GTK+ for Win32' components disqualify the software 'package' for any professional application.



GTK+ Screenshot for Linux (source code for glade2 here (gtk_src.tar.gz))

GTK+ Overview

Name:	GTK+ - The GIMP Toolkit
Operating systems:	Linux, Win32
Programming languages:	C (C++ mit gtkmm)
License	LGPL
Advantages:	• base library for Gnome Windows Manager under Linux • installation package included in all standard distributions (installation very simple) • generic controls under Windows • well-proven (under Linux)
Disadvantages:	• Win32 implementation is unwieldy, does not run stable (status 09-2004)
Development environment:	e.g. *2glade* (GUI Builder), Anjuta
WWW:	http://www.gtk.org
Documentation:	manuals, tutorials, mailing lists e.g. http://developer.gnome.org/doc/API/2.0/gtk/index.html
Reference projects:	• Gnome Desktop • GIMP • Gnumeric
Distribution:	Linux: very wide spread, Windows: marginal distribution

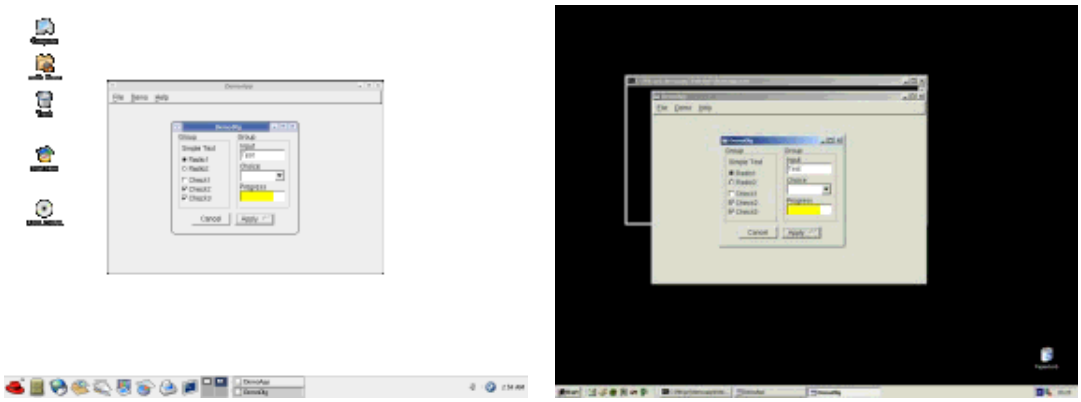
FLTK

FLTK Toolkit ((Fast, Light Tool Kit) is a largely unknown package, it was implemented as successor of XForms. The complete sources are being offered for downloading from the program's website. The size of 2.3MB (Linux) or 3MB (Windows) proves its name. Installation under Linux without a hitch: unpacking and run '*make*', finished. Subsequently the user has libraries, sample applications, the GUI builder "*fluid*" and a programming handbook at his (her) disposal. Obviously the number of classes at disposal is smaller than those of the heavy-weights of Qt and wxWindows. The classes included cover the GUI domain, meaning: windows, menus, controls, OpenGL and display of pictures. Classes for network communication and such are not included.

The installation under Windows was more complicated. When using the Visual C++ Development Environment only the main project needs to be translated. This causes however problems with the graphic libraries. A simple solution is to uncomment them in the config.h configuration file. A second Windows speific feature is that the DEBUG version of the FLTK library opens always an additional DOS window. This ensures that programs which are started from the command line will be able to write to stderr and stdout.

All together the FLTK Toolkit leaves the impression of being well thought-out. The documentation

emphasizes the small size of the executables (80kb for a "hello world") and lean fast 2D and 3D graphics (OpenGL). Furthermore, the good portability shall be mentioned.



Linux and Windows 2000 screenshot (source code here (fltk_src.tar.gz))

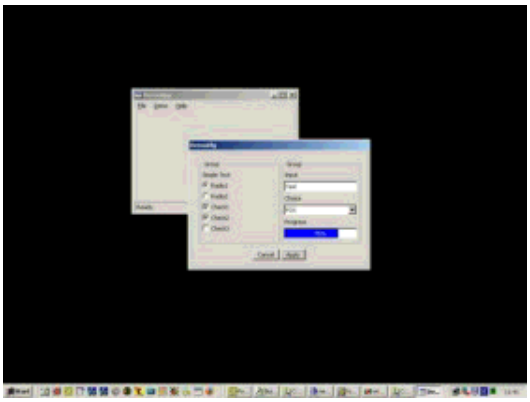
FLTK Overview

Name:	Fast Light Tool Kit
Version:	1.1.5rc2
Operating systems:	Linux, Win32, MacOS
Programming language:	C++
License:	LGPL
Advantages:	• a very lean library • Source code including documentation and development environment "fluid". • good OpenGL support (was not tested) • generic controls under Windows
Disadvantages:	• Installation under Win32 (Visual C++) not without problems • The fluid development environment does not run stable under Windows.
Development environment:	e.g. fluid (GUI Builder)
WWW:	http://www.fltk.org, Download: http://freshmeat.net/projects/fltk/
Documentation:	Manuals, Tutorials, Mailing Lists e.g. http://
Reference projects:	• http://vtkfltk.sourceforge.net/
Distribution:	low distribution, mostly unknown even amongst software developers.

FOX Toolkit

The Fos Toolkit claims to be the fastest available toolkit. It offers a large number of GUI elements and an OpenGL interface.

The installations completed under Windows and Linux without any problem. Detailed documentation and example projects are available. A class overview is not included in the version presented here but is available online.



Windows 2000 Screenshot (Source code hier (fox_src.zip))

FOX overview

Name:	FOX Toolkit
Version:	1.2.9
Operating systems:	Linux, Win32
Programming languages:	C++
License:	LGPL
Advantages:	• Good documentation
Disadvantages:	
Development environment:	
WWW:	http://www.fox-toolkit.org
Documentation:	Manuals, Tutorials, Mailinglist
Reference project:	• X File Explorer (Xfe)
Distribution:	low distribution

Other possibilities

In addition to the above mentioned libraries I would also like to mention the following projects which I will however not discuss further:

- GNUstep [http://www.gnustep.org/]: Limited usability under windows
- Visual Component Framework [http://vcf.sourceforge.net/]: No complete Linux version available

JAVA

In 1995 the company Sun introduced a new programming language. Besides the customary desktop-PC Java was planned for industrial products (coffee machines, toasters, etc.). The main breakthrough came initially through internet applications (applets) in connection with webbrowsers. In the meantime, Java is being used for standalone applications, for which it is well suited for its variety of features.

Below we will itemize and explainin short the most important features of Java.

Platform-independent

Java is platform-independent. Java applications consist of byte-code which may be interpreted by a virtual engine. Thus, the applications are able to run on any hardware for which a virtual engine exists. Interpretation by a virtual engine means a lower processing speed, compared to compiled software. To counter this disadvantage, improvements have been developed, like just-in-time compilation (JIT), which translates program instructions of the virtual engine into instructions for the physical machine. The result in this case is an aligned program in memory, which can be executed rapidly without interpretation. Aditional analysis of the runtime behavior with Hotspot-technology results in additional improvements.

Object Orientation

Java is object oriented. The developer of the object orientation language were inspired by Smalltalk. Presumably for performance reasons there are still primitive types of data which are not administered as objects.

Language Syntax

The language syntax is similar to the one of C and C++, however, bug inducing inconsistencies were not adopted. One principle for the development of the language was to combine the best concepts of the existing programming languages.

Some examples:

- no pre-processor. Pre-processor and header files are no longer necessary since all information is being read directly from the class files. .
- pointer: Java does not use pointers, references are used instead. A reference represents an object.
- garbage-collector: to prevent problems with creating and deleting objects, the object administration is being handled by the Java runtime-environment. By leaving the active array, the objects are automatically deleted. Objects or memory arrays, which are not enabled, as well as false destructors are being prevented by this technique.
- exeptions: contrary to the treatment of exceptions in C++ Java exceptions are used more intensely, they are often mandatory.

Class Library

Java includes an extensive class library: JFC (Java Foundation Class) for the generation of surfaces. (The code name *Swing* has caught on.)

Security

Java code is initially being checked by a verifier for structural correctness and security of types. A security-manager watches the accesses to the periphery. Any security problems are reportet as exceptions of the runtime.

Suitability for Projects

The advantages mentioned have side effects which render Java not wise for all projects. These properties are no mistakes or weaknesses but they were consciously not implemented, they belong to the philosophy of the language.
Amongst them are e.g.:

- platform-specific periphery accesses
- direct hardware accesses
- intervention in the operating system

Java Development Kit (JDK)

The Java Development Kit can be downloaded from Sun’s internet site. It includes a basic scope of applications, java classes and online documentation. The applications are a compiler, a debugger, an applet-viewer, as well as a variety of auxilliary programs, necessary to generate and test Java applications and Java applets. This kit offers only the most essential, the compiler needs to be run by command line. In addition, the package contains the Java Runtime Environment (JRE, includes the virtual engine), which is required to excute the byte code. The documentation finally describes the whole API.

JHelloWorld

With the help of standard JDK the mandatory "hello world"-application shall be implemented.
Step 1: Generation of the source code.

```
sh>vi Helloworld.java


public class HelloWorld {

  public static void main (String[] args) {


    System.out.println("Hello World!");

  }

}
```

File name and class name must match.
Step 2: Translation

```
sh>javac Helloworld.java
```

Step 3: Start application with the use of the virtual engine.

```
sh>java Helloworld
```

JavaScript and Java

JavaScript and Java are often assumed to have similiarities. That is basically wrong. JavaScript was originally developed by Netscape as a script language to be embedded in HTML. It is not a self-contained programming language, it depends on the browser application. The name JavaScript is more to be seen as a marketing gag .

Attempts to Standardization

Up to now all attempts to standardize Java have failed. Reason for this may be Sun’s reluctance to relinqish exclusive control over further development of the Java standards.

Dekompilierung

A problem might be that applications can be de-compiled. Despite all security it is at the moment possible to convert the Bytecode back into source code. This is possible because the Bytecode is written

for a virtual processor and contains in contrast to traditional assembler important additional information. The additional information makes it much easier to de-compile the code. You can therefore not hide a propriatary API or special knowledge in the code.

Miracle Language Or Short-lived Hype

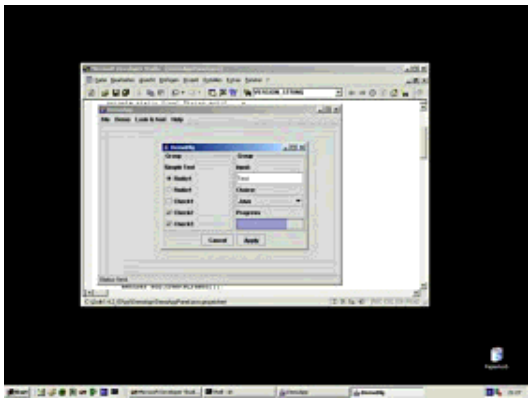
The Java concept was seen at the beginning as the ultimate answer to all platform independent development. However the original hype has disappeared. There are version conflicts between the different Java machnines and execution speed is an issue. Many companies went after first trials back to stnadard C++ Programming. The increased numbers of downloads seen by the wxWidgets developers is one proof for this.

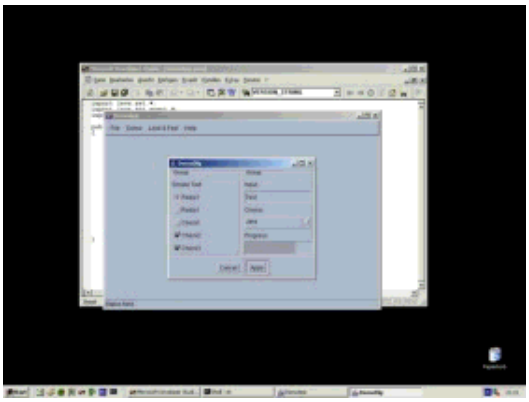
An intersseting Website in this context is:
http://www.internalmemos.com/memos/memodetails.php?memo_id=1321 where employs from Sun provide arguments against Java.

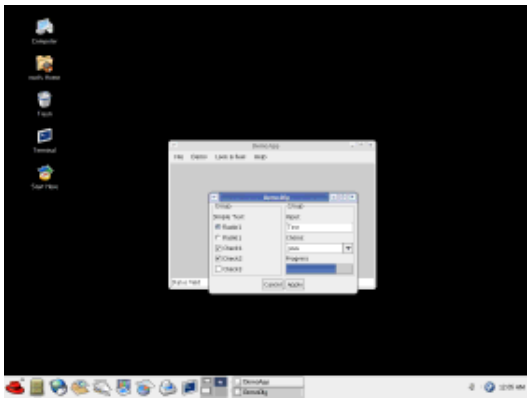
GUIs with Java

Java offers by default 2 possibilities to program graphical interfaces:

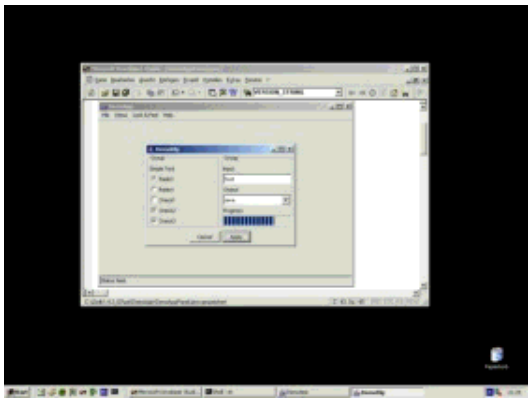
1. Jave comes with a rich class library (JFC, Swing). No oberating system functions are used here. All Widgets are draw with Java instructions. This makes it possible to change the look and feel at runtime. This can be seen in the screenshots below.
2. The basic AWT functions. AWT does not have complex elements like e.g trees it is therefore not suitable for most applications.

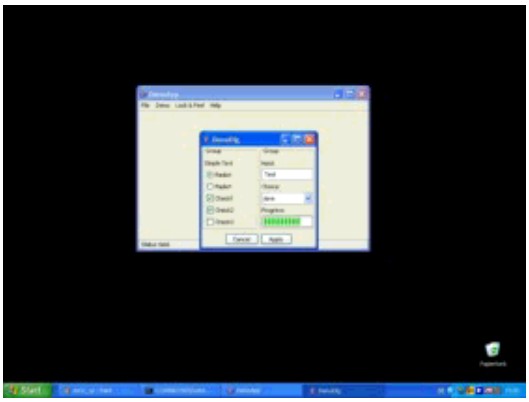






Java Screenshots im Metal-, Motif- and GTK+ Look & Feel (Quellcode hier (java_src.zip))

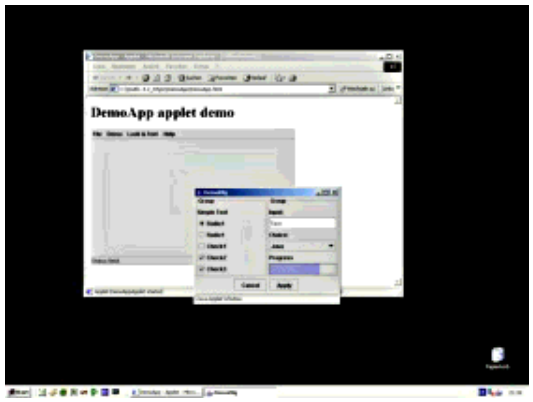




Java Screenshot with Windows Look & Feel under Windows 2000 and Windows XP (idential source code)

As all common used browsers support Java. Applications can therefore also be written such that they run as so called applets inside a Webbrowser. This technology can e.g be used for embedded technologies where the Java-Bytecode is downloaded from a webserver which is integrated in the application.

The following screenshot shows the identical application as a Java-Applet integratet into a webpage.



Java screen shot with the example application as an Applet (code here (java_applet.zip))

SWT and Eclipse

Eventhough Java offers similar GUI elements as other toolkit developers where complaining about them. The biggest problems where insufficient execution speed and lack of fuctionallity. IBM developed as an alternative the Standard Widget Toolkit (SWT) which allows the use of native GUI elements under Java. A refernce project is the also form IBM developed IDE Eclipse which offers platfrom independent development tools. The toolkit and the development environment are both free software.

Abbreviations used in context with JAVA

JDK (Java Development Kit)	The complete Java package to generate Java applications consists of application, Java classes and documentation.
JRE (Java Runtime Environment)	comprises the virtual engine, it is mandatory for the use of Java applications.
J2ME (Java 2 Micro Edition)	Java for devices with limited resources.
J2SE (Java 2 Standard Edition)	Java for the desktop (Linux, Windows, ...)
J2EE (Java 2 Enterprise Edition)	Java for the generation of multi-layer client/server-applications as well as Java-servlets and Java server-pages.
JFC (Java Foundation Class)	Classes to develop GUIs (->Swing)

Java Overview

Name:	JAVA 2 PLATFORM STANDARD EDITION DEVELOPMENT KIT 5.0
Version:	5.0
Operating systems:	• Linux, Windows, Solaris (SUN) • Linux, Windows, AIX, Solaris (possibly MacOS, OS/2, FreeBSD, Amiga, BeOS) (Jikes -> IBM)
Programming language:	JAVA
License:	proprietary license (SUN)
Advantages:	• robust language (many sources for errors are eliminated by the concept of the language). •
Disadvantages:	• proprietary language, controlled exclusively by Sun • virtual engine, must match the target platform • slow speed of execution • SWT programming is more complex than Swing
Development environment:	e.g.. Eclipse
WWW:	http://java.sun.com
Documentation:	manuals, tutorials general: http://java.sun.com/j2se/1.5.0/docs/, http://www-e.uni-magdeburg.de/mayer/java.html SWT: http://eclipse-wiki.info/SWT, http://www.java-tutor.com/java/swtlinks.html
Reference projects:	
Distribution:	very wide distribution

Kylix

Kxlix is a cross-platform development environment for Linux and Windows. With the help of Borland's CLX library (Component Library for Cross-platform) applications may be developed under Delphi and C++, which are able to run under both platforms. According to a report of the wikipedia homepage (Link de.wikipedia.org/wiki/Kylix) this library is only a wrapper for the previously described Qt library. In addition, the Kylix IDE is obviously a *wine*-based non-native Linux-application (Link de.wikipedia.org/wiki/WINE_Is_Not_an_Emulator) whose executables have to linked to *libwine*. Considering all this, Kylix may not make much sense for C++ developers since the use of Qt with a free IDE is more straightforward.

Kylix Overview

Name:	Kylix
Version:	3
Operating systems:	Windows, Linux
Programming language:	Delphi, C++
License:	Proprietary software
Advantages:	• develoment under Delphi and C++
Disadvantages:	• license costs
Development environment:	Kylix
WWW:	http://www.borland.de/kylix
Documentation:	
Reference projects:	
Distribution:	not wide spread

Smalltalk

Smalltalk is a classic amongst the programming languages. It was developed in 1969/70 by Xerox and is until today a good example for an object oriented language. Everything is an object in smalltalk. There are no simple datatypes. Smalltalk works like Java and .Net (see below) in a virtual machine. The syntax tries to be close to spoken English but is totally different from any other programming language. Smalltalk used to be programmend already from the beginning in a graphical environment. Smalltak was about 10-15 years ahead of its time. Smalltalk was quite successful until Java came.

Here the ’Hello world !’ programm under smalltalk:

```
Transcript show: ’Hello world !’; cr.
```

Smalltalk is still used today. The most widely available variant is Smalltalk-80 (standardized in 1980). A powerful development environment is e.g Squeak.

Overview of Smalltalk

Name:	Smalltalk (e.g. Squeak)
Version:	3.6
Operating systems:	Windows, Linux, Solaris, MacOSX, Darwin
Programming language:	Smalltalk
License:	Open Source
Advantagese:	Totally object oriented
Disadvantages:	Smalltalk is pushed asside by Java and has a significantly smaller user base.
Development environment:	e.g. Squeak
WWW:	http://www.smalltalk.org
Documentation:	
Reference projects:	
Distribution:	not wide spread

Mozilla

Mozilla? A web browser? How can you program with a web browser? Mozilla is not only a web browsers but also a platform independent Framework that includes different standards such as the XUL (XML based interface language). XUL is used to define the structure and content of an application. All files are used in clear text. Mozilla does not distinguish between programms and webpages.

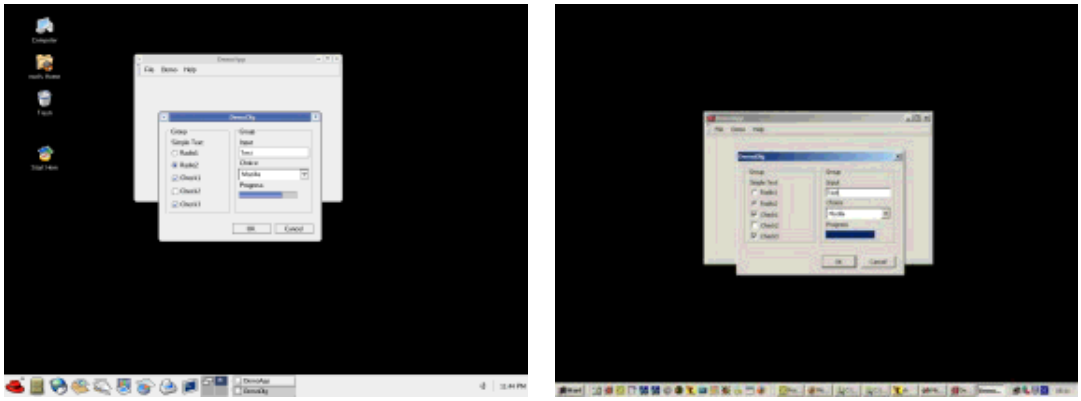
If you enter the following string into the URL field of mozilla then the browser itself will be shown:

```
chrome://navigator/content
```

The following code displays a button in the Mozilla browser which will open a window with the text "Hello World" when you click on it:

```
<?xml version="1.0"?>
<!-- Beispiel XUL Datei -->
<window
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<box align="center">
  <button label="Push" onclick="alert('Hello World');" />
</box>
</window>
```

Software development with mozilla is very differnt from classical software development. Mozilla has many innovations such as the seperation of the application and its presentation. This makes it possible to change the look of the application ("Themes"). Successful projects such as the firefox web browser show that it is a robust framework.



Linux and Windows 2000 screenshot (source code here (moz_src.tar.gz)).

Overview Of Mozilla

Name:	Mozilla
Version:	1.6
Operating systems:	Windows, Linux,
Programming language:	XUL
License:	Mozilla Public License, Netscape Public License
Advantages:	• innovative concepts • support for many web standards (JavaScript, Stylesheets,...) • applications run in the browser or standalone
Disadvantages:	
Development environment:	
WWW:	http://www.mozilla.org
Documentation:	Manuals, tutorials, mailing lists. E.g www.xulplanet.com
Reference projects:	Mozilla firefox
Distrubution:	widely distributed, but rarely used for software projects.

Microsoft’s Answer

In the meantime, Microsoft has of course recognized the signs of time and introduced it’s own approach. Under the name of .NET a platform was developed, which, last not least, shall reduce the migration of software developers to the competing Java platform. A closer look reveals indeed many parallels of the competitors, even though they are concealed by differnt names. The equivalent to Java’s ’Bytecode’ is

named C# ’Intermediate Language’ MSIL).

What is .NET ?

.NET is a proprietary Microsoft technology which shall be the base for all future Microsoft products. Support for the until now favored MFC-library for Visual C++ was abandoned with the introduction of .NET. .NET shall simplify the the development of network- and internet-applications; many ideas of Java were adopted. It supports object-oriented programming and is provided with a single class library which may be utilized by several programming languages (C#, VB.NET). That means, the ’Intermediate Language’ - which accesses the target hardware - is being generated from the program code (compare Java Sourcecode -> Java Bytecode -> virtual engine -> physical hardware)..

Future versions of Windows are to be supplied with the .NET framework.

What Is Visual Studio .NET ?

Visual Studio .NET is a programming environment to simplify the development of .NET-.software, but it is not mandatory.

Differences Between Visual Basic (VB) And VB .NET

Even though VB.NET - for reasons of compitability - supports many original VB-functions and the syntax of the language was maintained, it is a full-blown new programming language.

Which Programming Language Is The Best Suitable?

Since the VB.NET-source code and the C#-source code are translated into the MSIL, the programming language does not make a difference. There are, for example, no differences in speed between C#-code and VB.NET-code. The C# compiler should be the more suitable tool since it was developed specifically for the .NET framework.

.NET And Linux

Despite the platform-independant approach, Microsoft will most likely not develop a Linux .NET-variant, which is the reason why a developer team - close to Miguel de Icaza (Ximian: Evolution) - is engaged with this task. The open source package *Mono*, version 1.0, is in the meantime available.

Overview Of .NET

Name:	Microsoft .NET-Framework
Version:	
Operating systems:	Windows, Linux
Programming language:	C#, Windows: VB.NET
License:	proprietary license
Advantages:	• part of future Windows
Disadvantages:	• proprietary software • no Linux .NET Version available • completely new API
development environment:	Visual Studio .NET
WWW:	
Documentation:	
Reference projects:	
Distribution:	Low distribution at present

Summary

Prior to the final evaluation, the task to be accomplished shall again be referred to: the goal is the development of a front-end, which shall communicate by network with the connected hardware. For this the source code shall able to be translated in the Linux- and the Win32-platform. The application shall not be distinguishable from existing software on the system. With this task the view of the packages tested will appear skewed and cannot be assumed to be a valid general judgement.

Best example for this is the FLTK toolkit. With it we receive a very capable system in a very small package. The strengths are small source code, good graphic interface and good portability. These properties render the toolkit suitable for projects of embedded and graphic applications. For frontend development the number of available classes, the handling and the appearance of the application generated are more of an issue. Therefore FLTK is less suitable for this job.

A harsh disappointment for software developer may be the GTK+-project under Windows. The Linux community could demonstrate quite a bit more engagement. Warnings placed on the website are not really confidence-building. This is even more regrettable since the GTK+-package as such looks quite accomplished. The potential is quite large; implementation to the Windows platform is also wanting.

To utilize the outsiders Smalltalk and Mozilla remains personal preference. A company, which earns its income with in-house developed hardware, may have little understanding for philosophical attempts. Even though Smalltalk is the better object-oriented programming language and Mozilla's XUL-programming gives the included browser even more of a meaning, these packages are not mainstream products for software development.

In this review Kylix, as well as GTK+ for Win32, are leaving a more negativ impression. Very little remains from the glory of the original product Turbo Pascal. In the 80's Borland provided a powerful IDE with this product, which ran on home computers as well as on very early PC's. It was known for its reasonable price and fast code. In the meantime much has changed. Borland became Inprise and went back to Borland. Turbo Pascal changed to Object Pascal, then Delphi and finally Kylix (of course with expansions and changes). The use of it does not make sense at present - at least for new projects.

In this environment Microsoft demonstrates that it has recognized the demands of our time. Initially, the company tried to push the Java standard with Visual++. Besides the standard commands of Java, Win32-API access and even access to the Windows registry were permitted (which is quite contrary to the language philosophy). In addition, Win32 executables were generated automatically. After some legal wrangling with Sun, a warning had to be shown to indicate that the newly created application may not run on other operating systems. The end of the story was Microsoft stopped its engagement with Java. A completely new strategy was developed. With .NET and C++ an entirely new standard was generated. The combination of Windows, .NET and C# are certainly a good matching package, but that was also the case with the now retired combination of Windows with Visual++ and the MFC class library. Disadvantage is that one is unconditionally at the mercy of the provider who wants to force "his" standard (Windows). Microsoft is most likely not planning any implementation of .NET to other operating systems in the forseeable future. The free conversion Mono has to prove its real-life capability first. Despite initial achievements, at present no conclusion may be drawn.

Without limitation recommendable are the packages Qt, wxWindows and Java. The final choice is difficult since all three products are capable to generate complex front-end software. Different oppinions may evolve here, depending on the weighting of support, costs, readiness, programming philosophy, etc. The distinctions may be found in the details; the Java philosophy, in fact, does not permit direct hardware access, but it may have advantages in other aspects. From the technical point of view the three competitors can handle the requested task without problems.

Remains one subjective conclusion by the author: the Open Source-fan may lean to wxWindows for the task to be accomplished. Besides an agreeable concept and good tool support sufficient documentation is available.

Webpages maintained by the LinuxFocus Editor team © Michael Tschater "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Translation information: de --> -- : Michael Tschater <tschater/at/web.de> de --> en: Jürgen Pohl <sept.sapins/at/verizon.net>

2005-01-14, generated by lfparser_pdf version 2.51