ETNA
End Study Project 2005-2007
RimElse
Production Plan

18th April 2006

Contents

Chapter 1

Introduction

Nowadays, computer's users are more nomad. But, they are also confronted to a new problem : how to have an operating system which agree to their needs without a laptop ? Today, free software's solutions provide mobile solutions like "live cd" or "USBlive" keys systems. But, yet, this solutions still have inconvenient like data access rapidity on the source media (ex: the live CD disk). Moreover there is no persistence of data created when using this solutions and at least the impossibility to personalize to make your own system.

So, for their study's end project, the Else group, ETNA Linux System Engineers composed of six students decided to provide a solution to problems quoted previously. This one : the RimElse distribution is the subject of the present production plan.

This production plan include, first, a description of the distribution's different characteristics, like the Run In Memory and the "evoluability" gestion. In a second time, it describe the different realisations of the ELSE group in order to build the RimElse distribution. And finally, the technical choices made by the development team will be exposed.

Chapter 2

Presentation of the project

2.1 An "Evoluable" Linux Distribution

To give solutions, we have created "RimElse GNU/Linux".

"RimElse GNU/Linux" is a "GNU/Linux Distribution" "Run In Memory".

First, let's see what is a GNU/Linux distribution.

The wikipedia.com[1] definition is very talking :

> A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system, window manager, and a desktop environment. Most of the included software is free software/open-source software which is distributed by its maintainers both as pre-compiled binaries and in source code form, allowing users to modify and compile the original source code if they wish. Other software included with some distributions may be proprietary and may not be available in source code form.
>
> Many provide an installation system akin to that provided with other modern operating systems. Self-hosting distributions like Gentoo Linux and Linux From Scratch provide the source code of all software and include binaries only of a basic kernel, compilation tools, and an installer; the installer compiles all the software for the specific microarchitecture of the user's machine.

[1] http://en.wikipedia.org/wiki/Linux_distributions

> Distributions are normally segmented into packages, each package holding a specific application or service; examples of packages include a library for handling the PNG image format, a collection of fonts, or a web browser. The package is typically providing as compiled code with installation and removal of packages handled better than a simple file archiver. This software is said to be the package management system (PMS) of the distribution. Each package intended for such a PMS contains meta-information like description, version, "dependencies", etc. The package management system can evaluate this meta-information to allow package searches, to perform an automatic upgrade to a newer version, to check that all dependencies of a package are fulfilled and/or to fulfill them automatically. Package management systems include:

The run in memory is based on a simple idea, the complete system is loaded in memory and executed from there.

Many similar products already exist, but we offer to bring a non negligible favour, create an "Evoluable" system, this word is the alliance between Evolution and Malleable.

Evolution, because we offer to user the possibility of change his softwares versions and, add or delete ones like a classical distribution.

Malleable because we want that system be able to answer to each users needs, but also to satisfy the most people. Without omit the fact that the system is "Run In Memory" and so optimized.

"RimElse GNU/Linux" must be your pocket system !

2.2 Employment

It's easy to present concepts without having to determine the "employment".

RimElse will be usable in a lot of contexts, we are about to describe two of them.

The first is an ETNA's student.

2.2.1 *etna_e* is an Etna's student

Be an ETNA's student means "having some constraints on his system":

- Have his Intranet auto login link,
- Have a flash player for his web browser,
- Have a SSH client to connect school computers,
- Have a C compiler and debugger to work on projects,
- etc...

A student working on an another computer don't have all his tools. He wouldn't work in case he lacks a laptop.

The RimElse distribution's interests are:
With an USB KEY (and a CD-ROM if the guest computer's motherboard doesn't support USB boot), the student *etna_e* will have his own work environment, and keep his work and different documents.

Figure 2.1: Student usage schema

2.2.2 *adm_p* is a computer science teacher

adm_p need to make a C or Unix lesson to his students, but the school doesn't own any UNIX computers.

He can adapt a RimElse to his needs with samba client, C compiler, he can load a Unix system on all computers with only one USB key.

Figure 2.2: Teacher usage schema

Chapter 3

The RimElse goals

The ELSE team defined some objectives for the RimElse project. The first is, have a fast and most effective device checking and starting sequence.

Then RimElse will be able to run only in RAM, without access on the media

A system synchronization must be possible when the user wants, in order to save his modifications.

RimElse, like any other distribution have its own configuration tools and binaries packages management. That's why our goal is to improve the dependences management by providing the possibility to do binaries patches. So to add a new support for a software, like LDAP support for Postfix, he just have to apply a patch, and not reinstall the package. That's why we must reimplement a binary package type which be temporarily called .reb (RimElse Binaries). Those binaries must been optimized like for a embedded system, making the distribution lightest as possible.

And finally the last point, we will provide a custom system generation tool. Everyone will be able to generate his own system according to his needs. The *etna_e* doesn't have the same needs as the *adm_p* teacher but both need the RimElse.

3.1 Made RimElse

3.1.1 The realisation

To reach our goals, many steps have been established.

The starting sequence

First of all, it's necessary to optimize the starting sequence, it will result the creation of a devices detection script, the lightest and most efficient as possible.

As RimElse must be performent, "Starting Service Utilities" must be redefined.

The binaries management

RimElse is a Run In memory distribution, that imply a special binaries management. So, it have been decided to define a new packages type and an adapted manager.

The configuration's manager

The configuration manager will allow the user to modify his distribution parameters according to his needs, for example the graphical interface or network settings. As a result it will be linked with the packages manager.

The distribution's generator

To optimize the distribution tunning, it would be possible for the user to generate his own RimElse.

RimElse will provides a lot of profiles like ETNA Students or Java Developer.

According to the different quotes exposed. RimElse will be diffused on many supports, compact flash, CD-Rom, and network share.

Next chapter introduce ELSE will provide with RimElse.

3.2 The technical choices

Prior to the request concerning easy reading, disponnibility, the following choices were found essential.

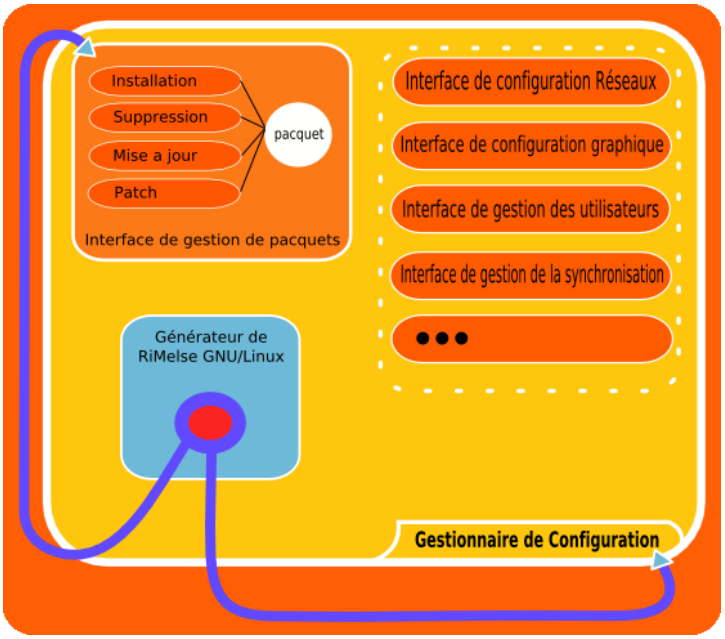

Figure 3.1: Configuration tool implementation

Concerning documentation, Two tools held our attention.

- On the one hand LaTeX, for the project's documentation part.
- On the other hand Doxygen which will generate documentation concerning the source code.

These tools allow to return documents under various formats like standard API's documentation.

In addition, any modification of the source code, or on the documentation must be able to be followed by the whole group. We have decided to use "git", a powerful Source Code Manager (SCM). In particular then used for the management of the source code for the Kernel Linux, it allows to warn the group of any modification.

Beyond the control aspect, this work process allows a total implication of each member, whatever his involvement. Indeed it can be in the project process to retrogress if an error were added in the code.

Regarding to the various developments we meaningly chose two languages :

- Ruby which is a combination of SmallTalk for the utilisation, Python for the simplicity and Perl for the flexibility.
- C, impossible to circumvent because of tis presence in the Linux Kernel.
- We should have to set shell scripts.

Those different choices lead to have a "perene" solution, but also to answer the problems of lisibility and availabilities.

Concerning the Linux kernel, we'll only work on version 2.6 or higher, because the newest are the most provided in hardware supports.

Chapter 4

Conclusion

Simplicity, disponiblity, efficiency plus plurality distribution supports, are RimElse's keywords.

On that view, ELSE will develop appropriate distribution Manager. We highlight some creations, boot scripts, binaries, configuration tools and RimElse Generator.

To conclude, GPLv2 preserves the project freedom.

Bibliography

[1] distribution definition : http://en.wikipedia.org/wiki/Distribution_Linux

[2] LaTeX : http://www.latex-project.org/

[3] Doxygen : http://www.stack.nl/~ dimitri/doxygen/

[4] Git : http://www.kernel.org/git/

[5] Ruby : http://www.ruby-lang.org/en/

List of Figures