

UCOME

Your Content Management ... in Tcl



Mandriva - Ouvert, simple et innovant
Arnaud Lapr votte – directeur des projets de recherche
arnaud.laprevote@mandriva.com

Content

- **History**
- **Structure**
- **How a file is processed**
- **Debugging**
- **Conclusion**

Story

- **Free&ALter Soft : services around free / open source software for industry 12/1996**
- **Bought Linbox in 2001**
 - => Linbox FAS
 - Switch from pure service to software edition / service
 - Linbox Rescue Server : a computer management solution : imaging, file backup, inventory, software deployment, keyboard/display control, integrated in a single web interface => used in Prime Ministry services, Ministère de l'Intérieur, Airbus for A380 flight test, préfecture, industry,
 - Linbox Directory Server : an open source AD killer.
- **Linbox FAS bought by Mandriva in 2007**

Linbox FAS & TCL

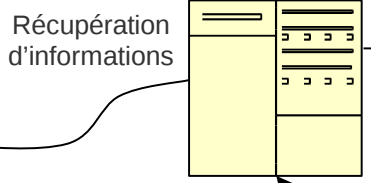
- **1997 : free software distribution for Solaris (then IRIX and HPUX)**
 - Main customers : Renault & MBDA (1998-2007)
 - Installation GUI + generator of environment variables in tcl/tk
- **1998 : tcl cgi program for ticket management in Thomcast**
- **1998-99 : tcl cgi program (fastcgi) to manage the time spent on projects for SNCF**
 - tcl is the main company programming language
 - but it is not an obligation : we are heavy users of scripting language (perl, php, python)
- **1999 : EQUAL project – infomobility**

EQUAL

INTERNET

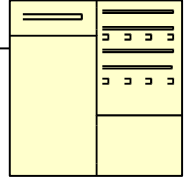
Site Météo	Site Gouv.
Site Info Rég.	Site Info Nat.
Gestion contenu	Site Info Culture
Infos de votre site.
Site web http://www.xyz.fr/rrooll_bus	

Serveur RrOoLI Bus

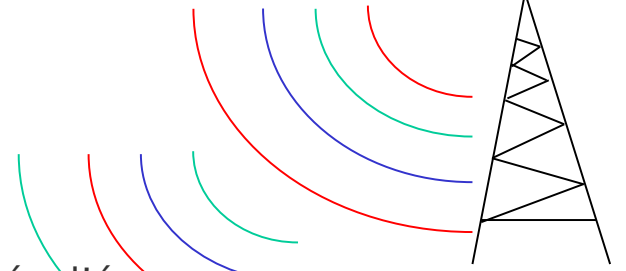


RTC

Modem DARC - (high speed RDS)



Radio Cool (+RrOoLI Bus sur DARC)



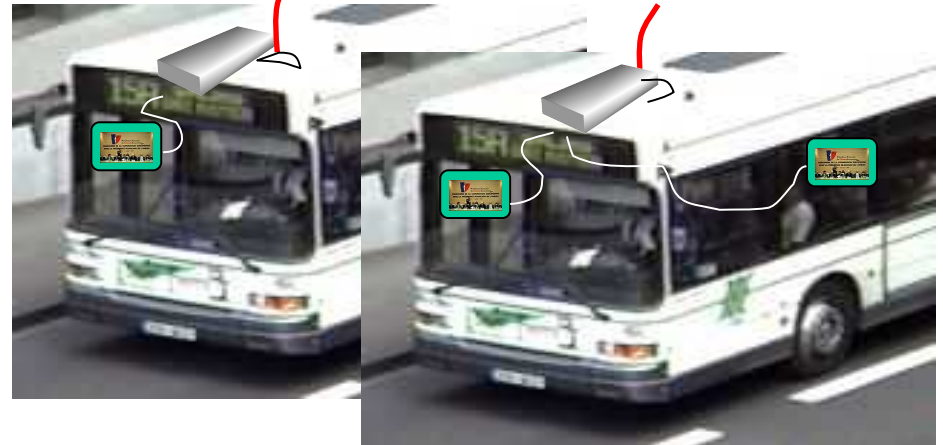
Bus équipés d'écrans LCD affichant RrOoLI Bus

Accès intranet ou internet à RrOoLI Bus



Gérant RrOoLI Bus
Choix / ajout de pages

Consultation des informations vues dans le bus sur le site internet de la société de bus



Architecture RrOoLI Bus

EQUAL

- **Presentation of informations in buses**
- **Pb : content creation cost**
 - take the content where it is : on internet !
 - reshape the content to adapt it to a “television” display
- **System :**
 - content creation
 - static content management (non internet)
 - transmission to display system(s)
 - display

Bravo !!!!

- **But what a mess !!!!!**
 - content aspiration and transformation with NewsClipper in perl
 - static content management with a perl cgi (webfm) + cgis in tcl (choice of pages and sources, set up, ...)
 - control of transmission via a tcl script
 - reception on the terminal and display with ad-hoc scripts : content is totally static
- **A lot of heterogeneous elements => difficult to maintain**

Linbox web site

- **At the same time, I begin to work on Linbox FAS web site content management system :**
 - I start from txt2ml : a script that transform text (à la wiki) in html
 - then I add what is required to manage menus and also have an on-line edition possibility

Enthroner – Ecim

- Follow up of Equal project is the European Enthroner project
- I am said : “just keep the previous code, and do small adaptations”
- Obviously, I take the occasion to create a full content management system in tcl

Constraints & properties

- Fully file based
 - Possibility to easily change and adapt the look
 - Management of rights and “properties”
 - Possibility to create an automatic rolling display
 - Multi-language
-
- Integrated debugging and logging solution
 - independent of server and technology (cgi, tclhttpd, apache rivet, websh, ...)
 - Management of menu by using directory structure and properties
 - Wiki like but with structured presentation of informations (except no history management)
 - Experimental tcl only connected ftpd server for direct file management (taking into account the rights)

File organisation of a site

- `/any` : directory of web site files
- `/any/.mana/.val` : properties of `/any`
- `/any/toto.txt` : a file that will be displayed
- `/any/.mana/toto.txt` : properties of `/any/toto.txt`
- `/cache` : cache of all files created
- `/cache/comp/any/toto.txt&&&` : html file obtained from `/any/toto.txt`. The `&&&` means here that no values were exported with the file, and no cookies are set or used.
- `/session` : session files
- `/template` : directory in which all templates are stored.
- `/comp` : directory in which files storing the actions for each part of a file are stored

How it works

- **Ucome is a transformation engine**
- **You have :**
 - an original file (in its original format)
 - an action
 - a target format (displayable, pdf, original, ...)
- **Transformations are done till getting something**
- **Every operation is cached**



- **directory “ $\${ROOT}$ /any/Programmer”**
 - determination that it is a dir
 - calling `dir::new_type`
 - answer is txt (it looks if there is an index.xxx in the directory, here it finds index.txt,
 - calling `dir::2txt`
 - the file index.txt is loaded and cached
- **text file**
 - calling `txt::new_type`
 - answer is comp (composite file)
 - `txt::2comp`
 - the text file is transformed in html (à la wiki)
- **comp**
 - calling `comp::2newtype`
 - answer is fashtml



Mandriva Example – viewing directory /any/programmer (2)



- **calling comp::2fashtml**
 - I will come back on this function after
- **fashtml**
 - **calling fashtml::new_type**
 - answer is htmf
 - **calling fashtml::2htmf**
 - all url are transformed depending if they are relative, absolute, some keywords (such as fas: are changed into what it should be depending on the url)
- **htmf**
 - **calling htmf::new_type**
 - answer is "" : it is a "final" filetype
 - **calling htmf::display**
 - using the function to send back the content depending on the "engine" used (cgi, tclhttpd, rivet, websh)

What about actions ?

- **file=/any/index.txt action=copy**
- **directory “\${ROOT}/any/index.txt”**
 - determination that it is a txt
- **text file**
 - calling `txt::new_type`
 - answer is ... copy
 - `txt::2copy`
 - function defined for everybody in `tcl/fas_basic_proc.tcl`
 - in variable `::STANDARD_PROCEDURES`
 - return “”
- **copy**
 - `copy::2new_type`
 - return “final” + variable `done=1` (action done)
 - `copy::2final` : the copy is done and a message is created (copy done or not)

Action (take 2)

- once the copy done ... restart a full display with
file=/any message="xxxx" action=view (call
display_file)

comp

- Just using a pipe is not flexible enough
- Each element of a page may be considered as an element coming from a transformation
- For example the menu :
 - file : /any/programmer
 - action: menu
 - target: nocomp

```
if (document.layers) { document.g@De
```

[Screenshots](#)

[User](#)

[Webmaster](#)

[Programmer](#)

[Design](#)

[Template library](#)

[Design of edit form](#)

[Ideas for future](#)

[To do](#)

[Source code](#)

[Buggs](#)

[Debug file](#) [Main log file](#)



The screenshot shows the Linbox website interface. At the top, there is a header with the Linbox logo and the tagline "logiciels libres pour les professionnels". A search bar and language selection (FR, EN) are also present. Below the header is a navigation menu with tabs for Screenshots, User, Webmaster, Programmer, To do, Source code, and Buggs. The "Programmer" tab is currently selected. The main content area displays the following text:

Home> Programmer

Design of UCome - Your Content Management

1. General remark

Tcl is a "glue" language. A language that does not try to do everything, but allows to "glue" different programs, different libraries in a single coherent application, that may have a graphical interface. This web application tries to extend this concept to the management of data in a web site. It will not do everything. No. It will allow to "glue" all intelligent programs that exist to publish existing data on the web as easily as possible. It will allow you to work as little code as possible to jump in the application. Once you jumped in, you will benefit of all the other modules that exist. For example, as soon as you are able to output html, you will benefit of the automatic creation of the menu depending of the place of the file in the directories.

What it means today is that, Free&ALter Soft deals with some filetype and some actions. Now, there is the rest : we do not provide a tex filetype, a gnuplot filetype, an xml filetype, a sgml filetype, an autocad, a dxf, a xfig, a tiff, a ... filetype. Because you use more these applications than we do, then you are much more able to have them fit in this environment. And if there is a real integration problem, it means we missed something and we will try to correct that.

UCome tries to be like Tcl (which is in my mind a very ambitious goal) :

- ♦ small,
- ♦ very quickly efficient,
- ♦ extremely useful.

comp

- You define what to display in a special file :
 - `${ROOT}/comp/${action}.form` or
`${type}.${action}.form` or
`${target}.${type}.${action}.form`
 - ie `${ROOT}/comp/view.form`

global.template *standard.template*

global.title "*Block definitions for the template for looking at a text file*"

global.title.fr "*Définition des blocs pour le canevas permettant de visualiser un fichier texte*"

title.type *file*

title.cgi_uservar.action *title*

menu.type *file*

menu.cgi_uservar.action *menu*

content.type *html*

allow.type *file*

allow.cgi_uservar.action *show_action_list*

allow.cgi_uservar.from *view*

comp

- **When doing `comp::2fashtml` :**
 - template file is downloaded (property `templatedir` and file given in previous `view.form`) => `standard.template`
 - for each section defined in `view.form` which is a file, the action is done with the file name
 - at the end, all html is integrated in the single template file to obtain the final file

Debugging

- **fas_debug.tcl**

```
# Enable (0) or Disable debug (0)
```

```
set DEBUG 1
```

```
# Useful on a big crash to determine where fas_view crashes
```

```
set DEBUG_FILE 1
```

```
# Show debug on html pages
```

```
set DEBUG_SHOW 1
```

```
# Mandatory - global variables in which all messages are accumulated
```

```
set DEBUG_STRING ""
```

```
# At 0 every possible message are enabled for all namespace
```

```
set DEBUG_ALL 0
```

```
# At 0 debug messages of function in no namespace are displayed
```

```
set DEBUG_MAIN 1
```

```
# Enable debug message for a given namespace
```

```
# If a LOCAL_DEBUG_COLOR is defined then messages are in this color
```

```
# for this namespace
```

```
catch {set atemt::LOCAL_DEBUG 0 }
```

```
catch {set fas_depend::LOCAL_DEBUG 0 }
```

```
catch {set fas_depend::LOCAL_DEBUG_COLOR "#00FFFF" }
```

```
catch {set fas_session::LOCAL_DEBUG 1}
```

```
catch {set fas_name_and_dir::LOCAL_DEBUG 0}
```

```
catch {set binary::LOCAL_DEBUG 1}
```

Debugging 2

- **`/tmp/ucome/ucome_1557-4171715781.dbg`**

```
fas_basic_proc::fas_get_value name => file args => -noe -nos -default  
/icons/ok.gif
```

```
fas_view.tcl - file -> /any/bug.txt
```

```
fas_basic_proc::fas_get_value name => action args => -noe -nos -default  
view
```

```
fas_view.tcl - action -> edit
```

```
----- displaying _cgi_uservar -----
```

```
ok.x -> 9
```

```
file -> /any/bug.txt
```

```
ok.y -> 19
```

```
ok -> 1
```

```
action -> edit
```

```
content -> All bugs (there are too many)
```

Logging

- **/tmp/ucome/ucome_1557-4171715781.dbg.log**

```
fas_session::open_session - fas_session::open_session -1-0 -
reading existing session file : /var/lib/ucome/session/1212224229_15502_368481
  cache_and_display - Calling txt::2edit on /any/bug.txt
    cache_and_display - Calling dir::2edit_form on /any
      atemt::read_file_template_or_cache - Using template
/template/linbox5/dir.template
      cache - Caching
/cache/edit_form/any&user=&&{action+edit_form}+{message+{Successful+writing+of+++any+bug.txt}}
      cache_and_display - Calling edit_form::2comp on /any
        cache - Caching /cache/comp/any&user=&&{action+edit_form}+{message-
{Successful+writing+of+++any+bug.txt}}
        cache_and_display - Calling comp::2fashtml on /any
          comp::2fashtml - Using form /comp/edit_form.form
          comp::2fashtml - Processing section path of type file
            comp::file::get_html - Processing file /any
              comp::file::get_html -          - action <= admin_path
              comp::file::get_html -          - new_type_option
```

Conclusion

- Opening
- Community ?
- To many bugs (to many actions, filetypes, cases, backends => websh sessions do not work ...)
- Changing the caching solution !!! everything is written in files, each step => to slow (but very practical for debugging !)
- Creating a good automatic testing solutions (for all actions, filetypes, ...)
- So many other things to do ...

Mandriva Linux

Ouvert, simple et innovant

