LUDE: A Distributed Software Library

Michael Dagenais - Ecole Polytechnique de Montreal
Stephane Boucher - Bell-Northern Research
Robert Gerin-Lajoie - Universite de Montreal
Pierre Laplante - Centre de Recherche Informatique de Montreal
Pierre Mailhot - Universite de Montreal

LUDE: A Distributed Software Library

Michel Dagenais – Ecole Polytechnique de Montréal
Stéphane Boucher – Bell-Northern Research
Robert Gérin-Lajoie – Universite de Montréal
Pierre Laplante – Centre de Recherche Informatique de Montréal
Pierre Mailhot – Universite de Montréal

ABSTRACT

Numerous software packages are being used and updated regularly on most computer systems. Installing all these software packages is a formidable task because each one has a different procedure for compiling or for placing the files required at run time. The LUDE (Logithèque Universitaire Distribuée et Extensible) software library is an organization for installing software packages, a set of tools to install and uninstall software packages and browse their documentation, and a number of FTP servers offering over 100 pre-installed freely redistributable software packages. It offers functionality and flexibility not available in existing systems.

Introduction

The LUDE software library is a joint project of the Computer Science and Operational Research Department of Université de Montréal (iro.umontreal.ca), of the Electrical and Computer Engineering Department of Ecole Polytechnique de Montréal (vlsi.polymtl.ca, info.polymtl.ca...) and of the Centre de Recherche Informatique de Montréal (crim.ca). The LUDE project was initiated in December 1991 to address the following goals:

- Serve heterogeneous systems.
- Let each disk server decide, on a package per package basis, if it wants a network access or a local copy of the executables and/or source code.
- Provide access to new software packages without editing user configuration files (.login, .cshrc).
- let more than one version of a given package coexist during transitions.
- Keep each software package in a separate subtree to ease the management of disk space and prevent name conflicts.
- Make all the documentation easily accessible through a single user interface.
- Let several independent organizations cooperate by sharing software package installation.

In this paper, an overview of the capabilities offered by Lude is presented. The detailed reference manual is found in the GNU info files that accompany the Lude distribution. The next section discusses where and why this project started. The following section reviews existing systems that address the problem of software distribution. Then, a section presents typical client server organizations. Next, the basic file tree organization is presented and Lude installation tools are described. The paper ends discussing status and availability of Lude as well as possible future developments.

Motivation and Organizational Context

The iro.umontreal.ca domain serves 45 professors and 600 students. It contains about 90 UNIX workstations from Sun, DEC and Silicon Graphics as well as 70 MacIntosh and 15 IBM PC compatible network clients. There are 600 user accounts and 8 full time hardware and software support staff in addition to one local administrator for each of the 8 computer laboratories.

A number of subdomains of polymtl.ca are using Lude and serve approximately 48 professors and 500 students (Electrical and Computer Engineering graduate and undergraduate students). There are approximately 100 UNIX workstations from Sun and HP/Apollo, 6 MacIntosh and 20 IBM PC compatible network clients. There are 700 user accounts, 5 full time technicians that work mainly on UNIX support, two research engineers with significant activities in software support and one departmental network analyst.

The crim.ca domain serves 400 users, researchers, software engineers, support staff and external users. It contains approximately 100 UNIX workstations from Sun, DEC, IBM, HP, NeXT and Silicon Graphics. The CRIM focuses its research activities along the following areas: Knowledge-based systems, Speech understanding and signal interpretation, Software engineering, Parallel architectures, Computerized control of industrial processes and computer vision, Teleinformatics and computer networking, Computer-assisted training environments and user interfaces.

All these sites had in common a large number of workstations used to support scientific and teaching activities. In particular, many software packages had to be compiled and installed independently at each site. Monthly meetings were held at iro.umontreal.ca to discuss software packages, networking and system administration. Representatives from polymtl.ca and crim.ca were invited to these meetings and became aware of the common problems and proposed solutions.

Following informal discussions, Robert Gérin-Lajoie, Pierre Laplante, Stéphane Boucher and Michel Dagenais decided to meet and attack the problem of organizing and sharing /usr/local. When this paper was submitted to LISA, an anonymous reviewer suggested a discussion of the problems involved in joint projects. This was probably left out unconsciously from the first draft, not because it is not an interesting question, but because of the difficulty of jointly writing such a section.

A few points are certainly noteworthy. During the 18 months when Lude was conceived and developed, 2 of the original 4 team members changed jobs. Each site went at least once in a state where because of internal changes (new servers, moving altogether) or personnel leaving they could not put any resource on the project for several months. In such a joint project, there is no hierarchical links between the members and one cannot impact much on the priority level assigned to the project by other team members.

Because of the combined experience of the team members, the final Lude organization is more mature and perhaps simpler than what any single member could have achieved alone. Moreover, the approval of members with different backgrounds brings confidence in the soundness of the proposed organization. On the other hand, in many cases the tools had to support everyone's favorite option (internationalization, minimization of symbolic links...). This made the tools more powerful, perhaps more complex, but certainly more difficult to implement.

During the design and implementation, discussions were held by phone, private e-mail and mailing list and several times minor communication misunderstandings arose. For example, one would understate his reluctance to a feature or his difficulty to meet a deadline with some team members more than others. The result would then be a skewed vision of a problem and of the best remedy, between the different team members.

Several other interesting joint development projects could have been pursued: user accounts management, backups... However, sharing software packages was probably in retrospect the best choice. Indeed, each site becomes immediately aware of the interesting packages used or developed by the other sites, including system administration tools.

Existing Systems

A number of systems were developed for managing the installation of software packages such as Xhier[1], Depot[2, 3], AUTOLOAD[4], and lfu[5] but none was found that met our requirements in terms of flexibility, heterogeneous support and, in particular, documentation indexing and browsing capabilities.

One such system, Xhier[1], was developed at the University of Waterloo. It has been used on a large scale and automates even the editing of system files like /etc/inetd.conf, adding required dummy user accounts. On the other hand, it was still evolving, used server initiated transfers and is relatively complex[1]. Furthermore, it cannot be redistributed because of license restrictions.

Lfu[5] was developed at University of Edinburgh and offers a relatively simple but somewhat inflexible organization. Indeed, the basic organization is a tree of servers with one master server for each architecture. In addition, it cannot accommodate easily more than one version of a software package.

Depot[2, 3] is the only system known to us that has been installed at more than one site. It evolved significantly from the first version and is now relatively close to the Lude organization. Indeed, each software package is in its own directory and symbolic links are created in /usr/local/bin, lib, include... towards the files exported by each package. Depot introduces the notion of package collection, not used in Lude, but does not manage multiple architectures. Furthermore, more than one version of a software package cannot easily coexist on a single computer.

Servers and Clients

The LUDE software library enables a large number of sites to pool the software packages compiled by their system administrators. Each computer can act as a client and/or a server as it desires. A client only needs a network connection to a LUDE server (such as the Internet). A server needs to install software packages as described in the LUDE documentation and to export them through NFS (Network File System) or FTP (File Transfer Protocol). A client may represent a more or less heavy load for a server:

- A client takes a complete copy whenever a new package is available and remains autonomous thereafter. This represents a light load and can be performed between distant sites.
- A client takes a copy of the install and run subtrees and maintains a symbolic link to the source code on the server. If access to the source code is relatively infrequent, this is not a very heavy load either.
- A client only keeps symbolic links to the server for the source code as well as the run

time for most software packages. Thus, each time one such package is accessed, the server is involved. This is only acceptable if the client and server are very close, on the same network and in the same organization.

Typically, a multi-level client server organization will be found:

- Public servers allow clients from around the world to take copies of their packages through FTP or NFS. Some usage restrictions may apply if the load is too high on these servers.
- Departmental servers regularly interact with public servers to keep a large set of up to date packages. Department clients may then use these packages either by taking a copy or even through symbolic links for the infrequently used packages. In some cases, a departmental server can also be a public server.
- A local server takes a copy of frequently used packages from the departmental server and perhaps keeps symbolic links for other packages. The source code for these packages is normally accessed through a symbolic link to the departmental server or to a nearby consenting public server.
- Individual workstations may simply mount /usr/local from the local laboratory server.
- Notebook computers copy packages according to their upcoming needs for standalone, nomadic, operation.

The compiled binaries for a software package will differ according to the target architecture and operating system. The two together form the class of the target system. The following classes have been registered up to now:

dec1.2_alpha	sony4.0_68030
hp8.0_s200	sun3.5_68010
hp8.0_s800	sun3.5_68020
ibm3.1_rs6000	sun4.1_sparc
linux0.99.10_386	ultrix2.1_uvax
pyr5.1_mips	ultrix4.1_mips
sgi4.0_mips	vax4.3_vax
sol2.1_sparc	

It is important that the same names be used throughout the various LUDE software libraries on connected servers; currently, the mailing list lude@iro.umontreal.ca is used for this coordination.

File Tree Organization

In /usr/local, the usual directories are found, in addition to server and soft, and have the following content:

- bin: symbolic links to executable files (adequate operating system and architecture class, default version). For example, emacs points to the file /usr/local/soft/emacs-18.59/run/poly/sun4.1_sparc/bin/emacs.
- lib: symbolic links to files used at execution time by the package. It can be libraries of compiled modules, fonts, macros or even executable files called within a package. As an example, m3 points to /usr/local/soft/modula3-2.11/run/poly/sun4.1_sparc/lib/m3.
- include: symbolic links to include files like declarations of procedures and data structures for library modules. For instance, m3 points to /usr/local/soft/modula3-2.11/run/poly/sun4.1_sparc/include/m3.
- man: symbolic links to man pages. For example, man1/emacs.1 points to /usr/local/soft/emacs-18.59/run/poly/sun4.1_sparc/man/man1/emacs.1
- info: symbolic links to hypertext files as used in the GNU project.
- doc: symbolic links to unstructured documentation and to software description files in Internet Anonymous FTP Archive format (IAFA-PACKAGES). For example, lude-1.6/IAFA-PACKAGES points to /usr/local/soft/lude-1.6/install/IAFA-PACKAGES while lude-1.6/README points to /usr/local/soft/lude-1.6/run/crim/sun4.1_sparc/doc/lude-1.6/README.
- server: symbolic links or mount points to accessible lude servers. For example, poly could be mounted on the directory lude.polymtl.ca:/usr/local/soft.
- soft: one subdirectory for each locally available software package. For example, one finds there emacs-18.59, modula3-2.11...

Thus, each software package is placed in its own subtree in /usr/local/soft. Moreover, every version of a software package is treated as a different package with its own subtree. The unique name of a package is then formed by the concatenation of its name and version number (e.g., emacs-18.59, modula3-2.11). This enables more than one version of the same software to coexist peacefully during transitions and simplifies the management of disk space since each package is kept separate.

However, several modifications (or minor versions) may exist for a package; these usually represent minor modifications to the original source code (for instance in the Makefile). Most often, a single modification is needed and is named after the person or the site performing the compilation.

Inside the subtree, three subdirectories are present src (original source code and modifications), run (everything needed at run time, possibly for several platforms and modifications) and install (description of the package and possibly special actions related to installing this package), as well as a file, history, which traces where this package was copied from. When a software package is installed, symbolic links are created in /usr/local/bin, lib... and point to files or sub-directories in the run subtree of the package.

In more details, a software package subtree contains, for example for a modification named crim and the class sun4.1_sparc, the following subdirectories and files:

- history: actions performed to copy/link locally this software package, for tracing purposes.
- src/orig/*: all the files exactly as they were in the original source code distribution.
- src/crim/*: all the files added to or changed from the original distribution in order to create the crim modification. Often this directory simply contains a modified makefile.
- install/IAFA-PACKAGES: software package description in IAFA format.
- install/crim/LUDE: description specific to the crim modification.
- install/crim/sun4.1_sparc/LUDE: information about who compiled the sun4.1_sparc class, and when, for the crim modification of this software package. Additional files in the same directory may specify files that do not require symbolic links in /usr/local/bin, lib... or actions to perform before and after the local installation.
- run/share: files common to all modifications.
- run/crim/share: files common to all classes within the crim modification.
- run/crim/sun4.1_sparc/bin, lib, include, man, info, doc: all the files required at run time for the sun4.1_sparc class of the crim modification. These files will have symbolic links in /usr/local/bin, lib... pointing towards them. Often the man, info and doc subdirectories in run/crim/sun4.1_sparc will be symbolic links to the corresponding directories in run/crim/share to allow transparent sharing of architecture independent files.

Installation Tools

Lude is both an organization, described in the previous section, and a set of tools. The lude command is a tool that can copy a software package from a server and install symbolic links in /usr/local/bin, lib...; it can also unlink and remove a software package. For example, to install modula3-2.11, modification poly, from the lude-poly server, linking the source code and copying the run time, using the default class for the local machine, the following command is used:

```
% lude -copy run \
    -link -software modula3-2.11 \
    -modif poly -server lude-poly
```

The ludeadm tool is used to create an empty package subtree, separate the local modifications from the original source code and release the compiled package when it is ready for public consumption. The sequence of commands shown in Figure 1 is typically used to compile a new package for Lude.

The ludeindex tool organizes and indexes the documentation. It can generate a keyword database

```
% ludeadm -create -software emacs-19.4 -modification crim
% cd /usr/local/soft/emacs-19.4/src/orig
% ftp prep.ai.mit.edu
ftp> cd pub/gnu
ftp> binary
ftp> get emacs-19.4.tar.Z
ftp> quit
% zcat emacs-19.4.tar.Z | tar xf -
% cd ..
% mv orig/emacs-19.4 .; rmdir orig; mv emacs-19.4 orig
% ludeadm -software emacs-19.4 -modification crim -duplicate
% cd crim
% vi makefile
% make all install
% make clean
% cd ../..
% ludeadm -software emacs-19.4 -modification crim -unduplicate
% vi install/IAFA-PACKAGES
% vi install/crim/LUDE
% vi install/crim/sun4.1_sparc/LUDE.lock
% ludeadm -software emacs-19.4 -modification crim -release
% mail -s emacs-19 lude@iro.umontreal.ca
```

Figure 1: Compiling a new Lude package

for man pages (using the catman command) as well as create a Wide Area Information System (WAIS) database using the synopsis and the first description paragraphs of each man page. It also creates a main menu for GNU info files. Finally, it creates a World Wide Web (WWW) html file for each software package, from the corresponding IAFA-PACKAGES file. The html file also contains hypertext links to the man pages, info and doc files that come with the package. Furthermore, man pages and info files are converted on the fly to html format upon access; the conversion preserves the info hypertext structure and handles adequately the SEE ALSO section of man pages.

Ludeindex can index not only locally installed software packages but also those on remote Lude servers, indicating their availability in the html file. This way, software packages can easily be found either through a main menu or through keyword searches. Moreover, the associated documentation is readily available through hypertext links using the same browsing tool. Even more, chances are that many users are already familiar with WWW browsing tools since they are increasingly used to access public databases such as university course and staff directories.

Status and Availability

The LUDE tools are written in Perl and amount to 5000 lines of commented code. The user manual is a 2000 lines info file. All text messages are kept in a separate file to offer multi-lingual support. At current time, both English and French are fully supported. The LUDE tools and associated documentation are freely redistributable under the terms of the GNU General Public License. They can be obtained through FTP from ftp.crim.ca:lude-crim/lude-1.6.

There are at least three public LUDE servers in operation offering more than 100 different software packages: ftp.crim.ca, ftp.iro.umontreal.ca, ftp.vlsi.polymtl.ca. Three mailing lists are used to coordinate the activities surrounding Lude: lude-request@iro.umontreal.ca to subscribe/unsubscribe, lude@iro.umontreal.ca where discussions and announcements take place and lude-bugs-@iro.umontreal.ca where bug reports should be sent. Nine countries are currently represented on the lude@iro.umontreal.ca mailing list.

Over 1600 software packages (source and/or executables) were downloaded from ftp.iro.umontreal.ca and ftp.crim.ca from the Lude tree. This covers the period from the 23rd of June to the 25th of August 1993.

Slightly over 1000 different usernames (representing perhaps 900 different users) did perform these transfers. All of the following top-level domains were represented: at, au, be, br, ca, ch, cl, com, cs, de, dk, ec, edu, es, fr, gov, gr, hk, hu, ie, il, in, it, jp, kr, mil, mx, net, nl, no, nz, org, pt, se, sk, th, tr, tw, uk, us, ve, za.

Surprisingly, a good proportion of the transfers concerned only source code. It seems to indicate that a major use of Lude is to serve as an extensive source code library, including modifications required to install each package on some architectures. Indeed, even though a site may not want to copy executables, it may copy the local modifications performed to compile the package on the target architecture, verify that these modifications are sensible and perform the compilation; this still represents considerable savings as compared to starting from scratch, while not compromising security.

A publically accessible WWW server demonstrates the documentation indexing and organization achieved through the Ludeindex tool: the ''lude list'' and ''lude index'' menu items in http://froh.vlsi.polymtl.ca:80/usr/local/lib-/WWW/default.html. A publically accessible Gopher server, gopher.crim.ca item RISQ/Lude, provides access to the lude and lude-bugs mailing lists and maintains a list of Lude servers.

Conclusion

The Lude project significantly reduced the time spent compiling software packages. Thus our sites were able to offer a much wider selection of up to date software packages. Managing the local installation of packages through lude is much simpler and the documentation is now well organized. All these immediate benefits demonstrate the success of this project. A secondary benefit is that each site is now more aware of good things happening at the neighboring Lude sites since each newly installed package is advertised to all three sites.

Collaborative development is a difficult task since each site has different priorities at different times. A volunteer must be found each time a new sub task is identified. The amount of work separating a first locally working prototype from a mature, tested and fully documented release is easily underestimated.

Three areas are currently getting our attention regarding future developments. One area is graphical user interfaces. A graphical front end could list the available software packages, modifications and classes on which operations such as copy and link can be applied. A second area is automating the software selection and updating process. A tool could list available software packages sorted by compilation date and/or keywords and even automatically perform the installation based on those criterions. Another possibility is to initially install packages through symbolic links and then based on usage decide which package should be copied locally.

A third area is security. Copying executable files always carries a certain risk. Secure communications through authentication or cryptographic checksums can be used to alleviate the risk.

Author Information

Michel Dagenais received his B. Ing. from Ecole Polytechnique de Montréal in 1983 and his Ph.D. from McGill University in 1987, both in Electrical Engineering. He is a professor in the department of Electrical and Computer Engineering at Ecole Polytechnique de Montréal. His interests include distributed object oriented programming for CAD applications, and system administration. He can be reached at dagenais@vlsi.polymtl.ca.

Stéphane Boucher graduated from Ecole Polytechnique de Montréal with a B. Ing. in computer engineering. He worked as software developer and then as network analyst for the department of Electrical and Computer Engineering of Ecole Polytechnique de Montréal. His work on Lude was completed while he was at Ecole Polytechnique. He is now a software engineer for Bell-Northern Research in Ottawa, Canada. His interests range from operating systems and compilers to Software Engineering. He can be reached at sbo@bnr.ca.

Pierre Laplante is a senior system administrator at the Centre de Recherche Informatique de Montréal. He earned a B. Sc. in Computer Science from Universite de Sherbrooke. His current interests include heterogeneous system administration using UNIX and X programming, in particular developing system administration tools with perl, wafe, c and c++. He can be reached at laplante@crim.ca.

Robert Gérin-Lajoie is Chief Laboratory Manager in the Computer Science and Operational Research Department at Universite de Montréal. He received his M.Sc. in Computer Science from Universite de Montréal in 1981. His interests include large Unix systems administration tools and methodologies, and information and knowledge access tools. He can be reached at rgl@iro.umontreal.ca.

Pierre Mailhot is currently system administrator at the Computer Science and Operational Research Department of Universite de Montréal where he previously obtained B. Sc (1986) and M.Sc (1989) degrees in Computer Science and worked 3 years as programmer-analyst with the department's VLSI laboratory. His interests include system administration, operating systems, security, distributed simulation and CAD tools for VLSI. He can be reached at mailhot@IRO.UMontreal.CA.

Bibliography

[1] J. Sellens, ''Software maintenance in a campus environment: The xhier approach,'' in *Proceedings of the USENIX Fifth Large Installation Systems Administration conference*, (San Diego, California), pp. 21-28, October 1991.

[2] K. Manheimer, B. Warsaw, S. Clark, and W. Rowe, ''The depot: A framework for sharing software installation across organizational and unix platform boundaries,'' in *Proceedings of the USENIX Fourth Large Installation Systems Administration conference*, pp. 37-46, October 1990.

[3] W. Colyer and W. Wong, ''Depot: A tool for managing software environments,'' in *Proceedings of the USENIX Systems Administration LISA-VI conference*, (Long Beach, California), pp. 153-162, October 1992.

[4] D. Pukatzki and J. Schumann, ''Autoload: The network management system,'' in *Proceedings of the USENIX Systems Administration LISA-VI conference*, (Long Beach, California), pp. 97-104, October 1992.

[5] P. Anderson, ''Managing program binaries in a heterogeneous unix network,'' in *Proceedings of the USENIX Fifth Large Installation Systems Administration conference*, (San Diego, California), pp. 1-9, October 1991.

Appendix A

To install LUDE, one needs access to the /usr/local directory, and bin, lib, include, man, info, doc, soft and server sub-directories must exist. A new sub-directory under /usr/local/soft will be created for each software package installed.

To start, the LUDE utilities and the Perl interpreter must be retrieved. The following FTP servers are accessible on the Internet: ftp.crim.ca, ftp.iro.umontreal.ca and ftp.vlsi.polymtl.ca. With FTP, it is often easier to retrieve the complete subtree for a package and then remove the unwanted binaries (for architectures not used at your site).

Here is how to proceed for installing lude and perl:

```
% cd /usr/local/soft
% ftp ftp.crim.ca
Connected to Clouso.CRIM.CA.
220 clouso FTP server (Version 2.0 Mon Apr 12 22:48:26 EDT 1993) ready.
Name (ftp.crim.ca:dagenais): ftp
331 Guest login ok, send e-mail address as password.
Password:
230-
230-This ftp daemon support tar and compress.
230-To get a directory, append ".tar" to the name of the directory.
230-To get a compress version, append ".Z" to the name.
230-
230-
230 Guest login ok, access restrictions apply.
ftp> cd lude-crim
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
X11R5
TeX-3.141
xrolo-v2p6
procmail-2.7
et3.0-alpha.1
lucid-19.3
hyperbole-3.04
wafe-0.92
cvswrapper-0.9
xntp-3.0
bibview-1.4
etgdb
perl-4.035
lude-1.6
226 Transfer complete.
859 bytes received in 0.3 seconds (2.8 Kbytes/s)
ftp> get lude-1.6.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for /bin/tar.
226 Transfer complete.
local: lude-1.6.tar.Z remote: lude-1.6.tar.Z
ftp> get perl-4.035.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for /bin/tar.
226 Transfer complete.
local: perl-4.035.tar.Z remote: perl-4.035.tar.Z
ftp> quit
221 Goodbye.
```

```
% zcat lude-1.6.tar.Z | tar xf -
% rm lude-1.6.tar.Z
% zcat perl-4.035.tar.Z | tar xf -
% rm perl-4.035.tar.Z
% sh
$ PERL=/usr/local/soft/perl-4.035/run/poly/sun4.1_sparc/bin/perl
$ cd /usr/local/soft/lude-1.6/run/poly_eng/sun4.1_sparc/bin
$ $PERL lude -sof perl-4.035 -mod poly -cl sun4.1_sparc -link
$ ./lude -sof lude-1.6 -mod poly -class sun4.1_sparc -link
$ exit
%
```

Then, any other software package is easily installed. Suppose that you also downloaded modula3-2.11.tar.Z in /usr/local/soft using FTP. The following commands are now sufficient to install it.

```
% zcat modula3-2.11.tar.Z | tar xf -
% rm modula3-2.11.tar.Z
% lude -sof modula3-2.11 -mod poly -class sun4.1_sparc -link
```