# PORTING CLEO SOFTWARE TO LINUX[*]

C. D. Jones, V. Kuznetsov
Cornell University, Ithaca, NY 14853, USA

## *Abstract*

The Linux operating system has become the platform of choice in the HEP community. However, the migration process from another operating system to Linux can be a tremendous effort for developers and system administrators. The ultimate goal of such a transition is to maximize agreement between the final results of identical calculations on the different platforms. Apart from the fine tuning of the existing software, the following issues need to be resolved: choice of Linux distribution, development tools (compiler, debugger, profilers etc.), compatibility with third-party software, and deployment strategy. It would be ideal to develop, run and test software using office desktops, local farm systems, or personal laptops regardless of the Linux distribution choosen. To accomplish this task, you need to have a flexible package management system that is able to install/upgrade/verify/uninstall the necessary software components without particular knowledge of remote system configuration and user privileges. We discuss how Linux became the third official computing platform of the CLEO collaboration, outlining the details of the transition from Tru64 and Solaris operating systems to Linux, and the software model and deployment strategy employed.

## INTRODUCTION

Existing HEP software has passed through many development cycles, following the evolution of computer languages, hardware and operating systems. In the late 90's, the Linux OS was recognized as a valuable platform and HEP experiments started porting their software to the new operating system. But a fear factor exists due to the variety of Linux distributions[1] and lack of support from hardware vendors. Two approaches were taken, developing a local Linux distribution [2, 3], or using a popular distribution, such as RedHat, with recommendations and customized kernels[4]. Regardless of which distribution has been used, developers have faced many problems porting their code to the new operating system. Fortunately, there are a variety of open source development tools to solve these issues. In addition to the well known gcc/gdb GNU compiler and debugger, we found the valgrind toolkit [5] to be very useful for profiling and debugging our code. In this paper we describe the CLEO experience of porting existing software (780+ packages) from Solaris and Tru64 platforms to Linux.

[*] Revised by, September 25, 2004

## *Choice of Linux distribution*

The choice of Linux distribution may be compared to the decision of choosing a car. There are plenty to choose from. But it is clear that all cars will have an engine and accessories that determine how comfortably you will drive. In the case of Linux, it is a combination of a Linux kernel and other components that make it useful. In most cases, different Linux distributions are binary compatible, but there are a few exceptions. For instance, the recent addition of the NTPL[1] threading model to the 2.6 Linux kernel and glibc library causes incompatibilities between distributions utilizing the traditional Linux thread model and ones based on NTPL. It is also possible that some applications compiled on one distribution may expect a particular version of glibc at run time and often will not run with another. But once you stick with one kernel line and compiler/glibc tandem your software may be portable between different distributions.

In CLEO, we organize our software in a distribution-independent way to allow future upgrades and to be compatible with off-site Linux systems. To be able to compile and run CLEO software on Linux, a specific version of the C/C++/Fortran compilers and the associated version of glibc is required. This does not restrict the choice of Linux distribution, since multiple versions of the GNU compiler and glibc can easily co-exist, although their support and mainteinance can be a challenge.

We package all third-party software as a part of a CLEO release. This way we avoid version incompatibilities, simplify future upgrades of CLEO software and eliminate any requirements of having them installed on remote systems. A good example is the Tcl scripting language, which is embeded in our main application. It can be found on almost every Linux distribution, but that does not gurantee that all of them will have the same version. In addition, when it is embeded in a C++ application, the library needs to be compiled with the *-fexception* flag to enable exception handling. As we found, this is not the case of the Tcl library shipped with most Linux distributions.

By choosing this approach, we were able to install our software on many Linux distributions, including RedHat Linux, RedHat Enterprise Edition, Fedora, SUSE, Debian and Fermi Linux LTS. Currently there is no strong preference in the CLEO collaboration for which Linux distribution to use, although RedHat is mostly used at Cornell and one of the MC farms uses Fermi Linux LTS.

[1] Native POSIX Threads for Linux

## *Software fine tuning*

While porting our software from the Tru64 and Solaris platforms to Linux, many problems were found. They can be classified as follows:

- compiler/preprocessor/linker bugs
- STL-related bugs
- OS-specific bugs
- user-related bugs.

The advantage of using different compilers on different platforms is to see their strong/weak features, keeping our code robust against their bugs. In CLEO, a set of C pre-processor bug flags and C pre-processor macros for STL containers and iterators have been used[6]. Table 1 summarizes platform-specific compiler flags used to compile Fortran and C++ code on different platforms.

Our final goal was agreement of physics results on different platforms. We encounted many problems with the track finding part of the reconstruction software. Most of them were found in Fortran code due to round-off differences between Linux/Pentium and Solaris/UltraSparc platforms. Figure 1 shows a comparison between Solaris and

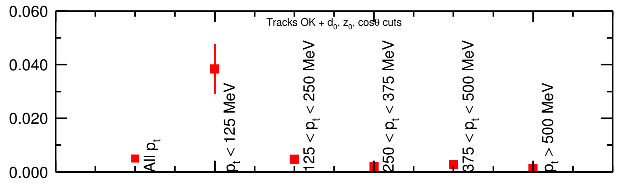

Figure 1: The fraction of unmatched tracks versus $p_T$. The unmatched track found on only one platform (Solaris or Linux) defined as: unmatched track identifier or tracks with the same identifier but no hits in common.

Linux based on one million MC events, the fraction of unmatched tracks as a function of track transverse momentum. We found around 15-20% of low momentum tracks differ in $p_T$ by $\sim 1\%$, but systematic differences between Solaris and Linux were smaller than the intrinstic systematic errors in the MC. As a cross-check we analysed $D^+ \to K^- \pi^+ \pi^-$ events and found good agreement between Solaris and Linux. This level was acceptable for physics needs and Linux was certified as a third official platform.

## *RPM as a deploy tool*

In order to understand how we deploy our software to off-site Linux systems, let us briefly describe the CLEO software model [6], which uses a mixture of Fortran and C++ code. Our main application depends on dynamic loading of software modules. On average, each module can depend on 20-30 libraries. The linking time rarely exceeds a few minutes per module on a system with 256 MB of RAM. However, on Linux it can be significatly improved by using local disk due to poor performance over NFS. To avoid run-time errors and possible mismatches of different versions of shared modules, we embed the absolute path of dependents into shared modules. For instance, module dependencies may look like:

```
ldd shlib/GroupEventStoreModule.so
 /cleo3/lib/libProcessor.so.v02_04_00
 => /cleo3/lib/libProcessor.so.v02_04_00
    (0x00e6c000)
 /cleo3/lib/libDataHandler.so.v01_27_02
 => /cleo3/lib/libDataHandler.so.v01_27_02
    (0x0074c000)
 /cleo3/lib/libToolBox.so.v03_08_00
 => /cleo3/lib/libToolBox.so.v03_08_00
    (0x0022f000)
 libz.so.1
 => /usr/lib/libz.so.1 (0x0011d000)
 ...........
```

As can be seen, *GroupEventStoreModule.so* depends on a specific location of other modules and may not be portable in binary form. Therefore, we deploy our software in a source format.

To deploy our software off-site we decided to use the RedHat Package Management (RPM) system, because:

- it is open source software and can be installed on any Linux, Solaris or Tru64 platform;
- it provides basic package management tasks, such as installing, uninstalling, verification, querying, etc.;
- it eliminates user errors during installation since all compilation rules are included;
- it is very easy to find problems with CLEO software on remote sites using RPM database.

To avoid user intervention in the installation process we provide a shell script which takes care of checking the system configuration, setting up the CLEO RPM database[2] and installing the CLEO RPMs. We use a separate RPM database for two reasons. We keep all CLEO packages apart from the system ones and avoid problems with needing system privileges. Such an approach is not only acceptable for all system administrators, since the system RPM database is not affected, but also convenient for users who can install CLEO software into an arbitrary location on their system, including their home area or special dedicated disks.

The CLEO RPMs have been organized into logical groups as shown in Figure 2. At the ground level there are release infrastructure packages which set up the CLEO release tree, and install Makefiles and setup scripts. They are followed by a set of third-party software RPMs, such as Qt, Mesa, ROOT, etc. Then the main application, suez, is installed. Once this structure is set up, users are able to run

[2]We separate RPM database for CLEO packages from system one.

| Tool/OS | Tru64 | Solaris | Linux |
|---|---|---|---|
| compiler | cxx/f77 | CC/f77 | g++/g77 |
| version | Compaq v6.2/Digital X5.2 | Sun WorkShop 6, v5.3 | v3.2.2 |
| debugger | ladebug | dbx | gdb |
| profiler | pixie | quantiy/purify | valgrind/cachegrind |
| C++ flags | cxx -O -Wall -nopt -nocompress | CC -instances=global -KPIC -mt -O | g++ -I- -O -fPIC |
| Fortran flags | f77 -O -u -static | f77 -O -u -KPIC -mt | f77 -I- -O -fPIC -MM -fno-automatic -fno-second-underscore -finit-local-zero |

Table 1: Platform specific compiler flags and development tools.

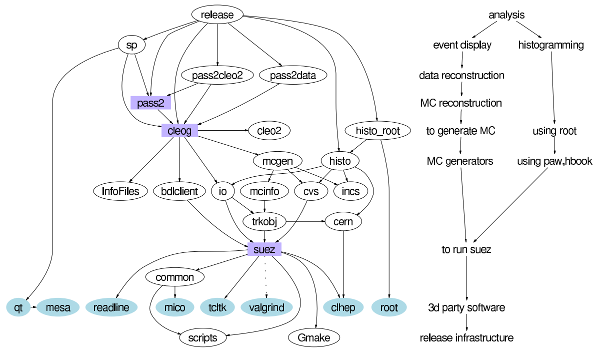

Figure 2: CLEO RPM dependency graph.

particular tasks on their system. A list of tasks includes: 4-vector MC generation, full detector MC simulation, MC reconstruction, data reconstruction, graphics, analysis and histogramming. It is worthwhile to mention that not all of these are required to be installed on every system. For instance, the *cleo3-qt* RPM is only necessary if the graphics package, *cleo3-sp*, is going to be installed. Right now our users decide which RPMs should be installed on their system, but we plan to provide an automatic procedure to customize user systems to perform particular tasks, e.g., to be a MC production node or analysis workstation. Thanks to RPM capabilities, it is possible to install different versions or particular components of the reconstruction software on a single Linux node. Different release versions may share some RPMs, for instance a set of third-party packages.

As previously mentioned, we are forced to deploy source RPMs for CLEO RPMs, although third-party RPMs are offered in binary and source forms. To install CLEO RPMs, users download the necessary pieces from an official location and invoke the installation script. Release name, build and install prefix paths must be specified at this point. A full release can fit into 3GB disk space, but additional temporary disk space is required during the compilation step. Currently, a CLEO release consists of 780 packages and it takes $\sim$ 12 hours on a Linux/Intel/1GHz/256MB laptop with local IDE drive to compile and install it. The existing deployment system can be further improved. It would be desirable to have a single administrative script that can retrieve on demand the necessary RPMs for a forthcoming installation or upgrade of a CLEO release on a user's system.

## *General comments*

One of the common tasks every developer routinely does is debugging and profiling code. A few commercial products served our needs well on Solaris and Tru64 platforms, such as Purify and Quantify from Rational (now part of IBM). Unfortunately, the Linux version of these tools is not usable in our framework. The PurifyPlus toolkit was ported to Linux from Windows. Unlike the Solaris version, where it could be embedded in any application, the Linux version creates its own framework, requiring every package to be part of its IDE. We found this model to be unusable and were forced to look for alternatives. Fortunately, we found excellent open source analog, valgrind [5], a very powerful debugging and profiling toolkit for Linux. It was easily integrated into our legacy application since it does not require any recompilation or re-linking of the program to being measured, and it provides a full set of memory checks, cache-miss profile, data-race detector and heap profiler. We found that in some cases the valgrind toolkit is able to find bugs which Quantify/Purify does not and vice versa.

In general we were very pleased with our transition to the Linux OS. It certainly boosts in many ways our software development and application performance. A good example is the switch of our MC farm from Tru64 to Linux. With modest hardware, it tremendously improved event production yield. Using 28 dual CPU Xeon 2.4GHz nodes, we were able to achive production of 12M/day $D\bar{D}$ events and 14.5M/day continuum events. On the software development front we were able to replace the CLEO III data management system based on the proprietary Objectivity/DB$^{\mathrm{TM}}$ with the new EventStore [7], whose implementation uses many open source software components including very hottechnologies such as web services.

## CONCLUSIONS

Linux has proven to be a powerful platform for the HEP community. Due to open source and rapid development in the Linux community, many new projects have been started. The CLEO experiment is not an exception. We have discussed the CLEO experience of porting our software to the Linux operating system. The physics results produced on Linux have been found to be in good agreement with Solaris. That allowed us to switch the MC production farm from Tru64 to Linux and significantly improved MC event production. In CLEO, we use a flexible RPM package management system to deploy our software off-site. The CLEO software has been successfully installed on many popular Linux distributions, including RedHat Linux, RedHat Enterprise Edition, Fedora Core, SUSE, Debian and Fermi Linux LTS. Linux has become the most successful platform for CLEO, serving almost every aspect of computing, including on-line calibration, MC and reconstruction farms and finally happy users running their favorite OS and CLEO software on their laptops.

## ACKNOWLEDGEMENTS

We would like to thank Dan Riley, Gregory Sharp, Hubert Schwarthoff for fixing various bugs and providing excellent technical advice. We also thank David Kreinick, Jean Duboscq, Werner Sun for pushing the Linux port and making the Solaris vs Linux comparison. We gratefully acknowledge all members of CLEO collaboration for their valuable feedback in software fine tuning during the transition to Linux.

This work was supported by National Science Foundation.

## REFERENCES

[1] http://www.lwn.net/Distributions/

[2] http://www-oss.fnal.gov/projects/fermilinux/

[3] http://linux.web.cern.ch/linux/

[4] http://www-clued0.fnal.gov

[5] http://valgrind.kde.org

[6] M. Lohner, C. D. Jones, “Rapid Software Development for CLEO III”, CHEP 2000, Padova, Italy, February 2000.

[7] C. D. Jones, V. Kuznetsov, D. Riley, G. Sharp “EventStore: Managing Event Versioning and Data Partitioning using Legacy Data Formats”, see proceeding of this conference.