Elephant, meet Penguin: Bringing up Linux in BaBar

M. Dickopp[4], *S. Gowdy*[1,6], *M. Kelsey*[5], *P. Raines*[3], *A. Romosan*[1], *A. Ryd*[2]

[1] Lawrence Berkeley National Lab, USA
[2] California Institute of Technology, USA
[3] Stanford Linear Accelerator Center, USA
[4] Technical University of Dresden, Germany
[5] Princeton University, USA
[6] Presenting Author

Abstract

During the past year much work has been done on BaBar by various collaborators to allow us to utilise the Linux commodity platform. Some of the decisions made have been forced upon us by outside constraints. Many recurrent problems problems were found and fixed in the more than 2 million lines of code BaBar runs. This talk will summarise the process of and the lessons learned in this venture.

Keywords: chep,linux,babar

1 Introduction

Over the past few years many experiments have been adopting Linux as one (or only) of their platforms. For BaBar this has been complicated with use of C++, for which finding standards compliant compilers is difficult.

BaBar decided to adopt Linux as a supported platform last year. The timescale for support was 1st January 2000. Our Linux support has now been deployed, although there are expected still to be a small number of problems which will not be found before general use.

2 The BaBar Linux Platform

The flavour of Linux finially decided on for BaBar was RedHat 6.0. Initially we developed it on RedHat 5.2 but wished to move to a newer version. We were limited in our freedom due to the use of the Objectivity object database, which is distributed as an archive library. This also determined which C++ compiler we could use.

Objectivity was built on a RedHat 5.2 system using egcs 1.1.1 + patch [1]. Therefore our initial development platform imitated this.

A fraction of our FORTRAN code ($\approx$60 source files) used nonstandard F77 which meant that GNU f77 would not compile them. To work around this we decided to buy a commercial compiler, this would only be used for the nonstandard files. The Portland Group's (`http://www.pgi.com`) compiler met our needs. Almost all of the cases (where there was a problem) was due to DEC STRUCTUREs. Shortly before New Year all of these STRUCTUREs were converted to standard data types and therefore we will no longer require the commercial compiler.

In our initial attempts to move to RedHat 6.0 (and therefore glibc2.1) we found our applications which used Objectivity would not start up. After sometime it was discovered that this was due to the commercial FORTRAN compiler containing a library called libpthread.a. Once this was moved aside we started to use RedHat 6.0. We also changed to use egcs 1.1.2 + patches.

[1] The patch is required due to a bug in egcs with implementing an operator new in a templated class

Function	Fixed in glibc2.1	New header file	Comments
clock_gettime	No	No	
regcmp	No	No	Deprecated
regex	No	No	Deprecated
semctl	Yes	No	Arguments incompatible
isastream	Yes	stropts.h	
statvfs	Yes	sys/statvfs.h	Linux used statfs

Table I: System functions missing in glibc 2.0

To summarise, the BaBar platform consists of;

- RedHat 6.0 Linux
- egcs 1.1.2 + patches (BaBar RPMs have been created)
- Lesstif 0.88.1 (version not thought to be crucial)

There are a couple of local configurations which SLAC will probably install on their machines (which includes the reference machine);

- Increased number of symlinks to follow from 5
- Increased number of processes from 512

These are kernel configuration options and therefore require a kernel rebuild.

Our collaborators are free to use whichever distribution they wish. However, any problems must be reproducible on the reference machine at SLAC to eliminate system configuration issues.

3 System Problems

Previously our computing platforms were Compaq Tru64 Unix and Sun Solaris. Our initial attempts to use Linux were based on the RedHat 5.2 distribution. This utilises glibc 2.0. Our code contained uses of many functions which were not available with this version of glibc (and some are still not available with 2.1).

To overcome this our normal method for working around platform deficiencies was used. This involves a header file (called BaBar.hh) which is included at the beginning of every source file. Within this file declarations are made of the missing functions. These functions are implemented in the same package which contains this header file : BaBar.

However, for some cases we also had to add a header file which isn't available with glibc. These missing header files are distributed as part of our releases. Table I shows the details of these patches.

There are also various preprocessor symbols which are not defined on Linux. Most of these are used for signals in online code which is only actually used on Sun, however this code is also compiled on Linux and OSF. An example of these are SIGEMT and SIGSYS. Definitions of these have been added to BaBar.hh to allow compilation on Linux.

4 C++ Issues

A few recurrent issues needed to be fixed while porting the C++ portion (the majority) of our code. Some of these were initially thought to be bugs in the compiler but most of them turned out not to be.

The egcs compiler has one flaw that requires you to create temporaries in some circumstances. This is usually while creating an instance of an object in a method invocation of another object, for example;

Problem	Workaround
Protected Destructors never accessible	Make Public
pow function with -O2 is incorrect	Use -O
iostreams can't read hex numbers with 0x	Remove 0x from data files

Table II: Problems with egcs 1.1.2

```
   ObjectA a;
   a.( B() );
```

would become;

```
   ObjectA a;
   ObjectB b;
   a.( b );
```

Although, the problem is more complicated that this as the example does not reproduce the error.

With egcs a template's definition must be seen so that it can be instantiated in the source file that uses it. We have a mechanism to accomplish this: as this was required in the past on other platforms. However, many new templated classes hadn't followed the guidelines and had to be retrofited with this mechanism. This is simply to add the following near the end of the header file for the templated class;

```
#ifdef BABAR_COMP_INST
#include "Package/MyTemplClass.cc"
#endif // BABAR_COMP_INST
```

and to make sure that the source file is not compiled into the library. At the moment all our platforms defined `BABAR_COMP_INST`.

Another trouble with templated classes was due to inlined methods invoking an "operator T*" directly without a `this->` (which the compiler requires for context).

The last problem in our code which wasn't due to a compiler problem was side effects during an output operation. For examples;

```
   HepSymMatrix a;
   cout << a << a.invert() << endl;
```

is not guaranteed to do what you expect, and infact egcs does the inversion first.

A few compiler problems that we have worked around in our code are shown in Table II.

These compiler problems are all solved with gcc 2.95.2, however, this introduces other troubles. Declaring a function without a return type is an error by default (`-fpermissive` reduces it to a warning). Various places in our code attempt to pass a pointer to a function without an ampersand before the function name, this is an error with gcc 2.95.2.

However, the real problem for us switching to this newer compiler is that the object format is reported to be different and we do not have a build of Objectivity for this compiler. We are (infrequently) making sure our code will compile with this compiler so that we can switch in the future.

5 FORTRAN Issues

As mentioned in Section 2, a lot of our code used DEC FORTRAN STRUCTUREs. These have now been removed, this was a fairly large effort to accomplish.

Several places in our code assumed that local variables kept their value over invocations of the function. This is not the case by default with g77 (the Sun and DEC compilers SAVE variables by default). SAVE statements have been added to facilitate this. Most of these were located in magnetic field code.

Various symbols also change type in our code, although it has a performance penalty we decided to add the -no-global option to allow this. We may revisit this in the future.

Another trouble which was found a couple of times was concatenating variable length strings. g77 does not allow this and creating a fixed length temporary string is required. Checks were also added to make sure that the variable length string was not longer than the temporary string.

6 Build Issues

Our SoftRelTools package is used to maintain our build system. It was fairly easy to extend to include Linux. The various third party packages we use have their support included in GNUmakefile fragment files (named `arch_spec_<Name>.mk`). Generally, these assume suitable values, however a couple needed extra flags to be specified for Linux. An example for this is that CERNLIB 98 was built using an older version of g77 and we therefore have to fool the linker with `-defsym xargc=f__xargc`.

Last July we started doing production builds on Linux. Our Release Manager did this without trouble. This has been very useful for the port even if the releases were not fully functional.

Before Linux was an officially supported platform it was necessary to test on Linux centrally as our developers were not required to. The reason was to make sure that the code did not drift. Usually something would need fixed in each release.

7 Conclusion

Linux is now an officially supported platform on BaBar. However, there will be a number of problems which need general use to find. With our next production release (the first of 2000) this will begin to happen.

We can look forward to the benefits this will bring. Many of our collaborating institutes now have the majority of their CPU on Intel machines. This port will allow them to contribute to our experiment's Monte Carlo production.

As I see it there are two major problems remaining on the Intel Linux platform. These are the lack of Objectivity distributions for the versions of glibc and compiler that we wish to use and the lack of large file support. It is hoped that the first of these will be solved by collaboration with other experiments wishing to use Linux and Objectivity. The solution to the latter may take a little longer to arrive and the main hope at the moment is the XFS file system from SGI (`http://www.sgi.com/newsroom/press_releases/1999/may/xfs.html`).