

Fermi National Accelerator Laboratory

FERMILAB-Conf-91/91

**UNIXTM in High Energy Physics:
What We Can Learn from the Initial
Experiences at Fermilab**

J. N. Butler
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

March 1991

* Presented at Computing and High Energy Physics Conference, KEK, Tsukuba, Japan,
March 10-15, 1991.



Operated by Universities Research Association Inc. under contract with the United States Department of Energy

UNIXTM in High Energy Physics: What We Can Learn from the Initial Experiences at Fermilab

Reported by J.N. Butler
Fermi National Accelerator Laboratory

March 21, 1991

Abstract

The reasons why Fermilab decided to support the UNIX operating system are reviewed and placed in the context of an overall model for high energy physics data analysis. The strengths and deficiencies of the UNIX environment for high energy physics are discussed. Fermilab's early experience in dealing with a an 'open' multivendor environment, both for computers and for peripherals, is described. The human resources required to fully exploit the opportunities are clearly growing. The possibility of keeping the development and support efforts within reasonable bounds may depend on our ability to collaborate or at least to share information even more effectively than we have in the past.

1 Introduction

In 1989, the UNIX operating system had virtually no presence at Fermilab. In 1990, after a series of meetings with major collaborations, the general user community, and vendors, Fermilab decided to officially support, and even to encourage, computers running UNIX. In 1991, our work is 'dominated' by activities associated with making UNIX useful for a wide variety of applications in high energy physics. In this paper, we will discuss why this happened so quickly and where we think it might lead.

In section II, we present a model of high energy physics data analysis and indicate where UNIX has made major inroads and where it has not. We discuss whether this is just an 'accident of the moment' or whether it is driven by some fundamental dynamic. We describe the issues related to the areas where it has not caught on and speculate on whether it will eventually emerge victorious in those areas as well. In section III, we discuss the strengths and weaknesses of the 'UNIX environment'. We include discussions of command language interfaces (shells), resource allocation and management, systems administration, application language support, development environment, and network environment. In section IV, we discuss the opportunities and pitfalls of the 'open', multivendor UNIX marketplace. We express some concern about the level of support that is required to fully exploit these 'opportunities'. We touch briefly on similar problems associated with third party peripherals, which, while not strictly a UNIX issue, is definitely part of related drive towards openness and interoperability. In section V, we make some suggestions for increased cooperation and perhaps even for outright collaboration to help deal with the 'new culture' of openness in order to capitalize on the opportunities and avoid the pitfalls described earlier.

2 How Fermilab wound up supporting UNIX

The three main reasons why Fermilab 'decided' to support UNIX systems were:

1. Economics
2. Economics
3. Economics

To understand this, we need to remind ourselves about how we do physics analysis in HEP. Figure 1 presents a model of high energy physics analysis as practiced at Fermilab and, undoubtedly, at the other HEP labs. The raw data sets collected by experiments are in the range of 1 to 10 terabytes. The ultimate goal of these experiments is to produce papers that tell us something new about the fundamental interactions of matter and they will finally do that in a collection of papers that will surely be less than 2.5 megabytes (about 1000 printed pages). The skillful and expeditious reduction of the massive amounts

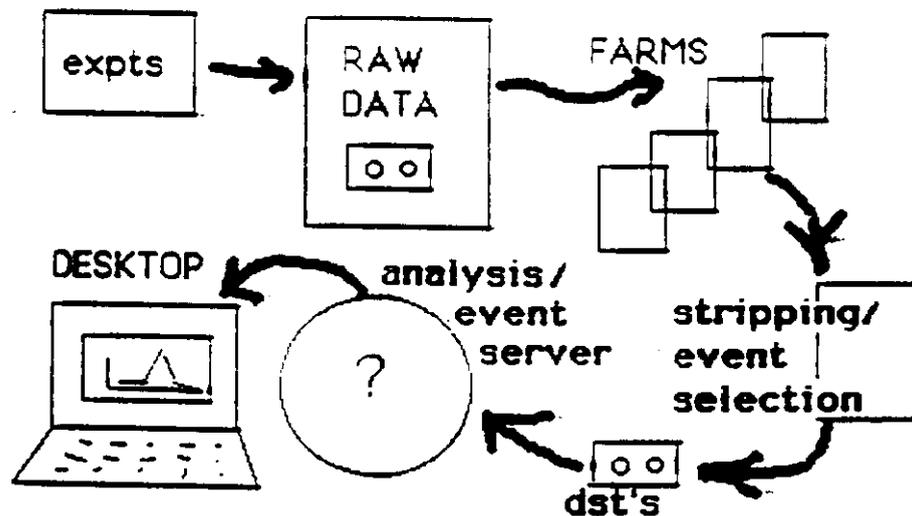


Figure 1: Physics Analysis Model

of raw data to the few (at most) megabytes of printed words and and some plots is what we call 'data analysis'.

The enterprise divides up into several parts:

1. Event Reconstruction.
2. Interesting event or 'candidate' selection. This step may be divided into several parts and may be associated with data compression and reformatting. Major physics judgements have to be made at this point.
3. Event serving- the provision of samples of events, usually sorted by physics topic, to individuals. The individuals are frequently organized in teams which attempt to coordinate data set needs.
4. Physics analysis- the search for signals and the extraction of physics information from them.
5. Report/paper preparation.

Table 1: CPU MIPS/megabyte/s for various analysis activities (estimated)

group	reconstruction (generation)	strip/filter	dst	micro-dst/ analysis
CDF89	2100	22	20	-
D0	250	5	-	-
D0MC	5000			
E708	750	60	-	-
E665	1200	60-500	50	50
E687	2000	500	50	50

Each part is characterized by a somewhat different range of compute cycle and IO requirements. In parallel with all these activities, of course, are the never-ending activities of program development and communication. The generation and analysis of simulated data (Monte Carlo) occurs at every step of the process.

2.1 UNIX for event reconstruction

Table 1 shows the compute/IO ranges of typical Fermilab programs at the various analysis stages, measured in MIPS/megabyte/s of IO. For comparison, a MicroVax III connected to an Ethernet can do about 3 MIPS/megabyte/s so any task requiring a higher number would be compute bound on a MicroVax III. The reconstruction step is typically very compute intensive and the analysis activities at the 'downstream' end tend to be IO limited. It is interesting that some experiments move from CPU limited to IO limited as they move down the analysis chain until they get to the final physics analysis, where there are many highly compressed events that still need significant computing cycles to process. At that point, they may become CPU limited at their workstations. This happens, for example, in fixed target programs with vertex detectors where there usually is a pass over the vertex assignments which involves a large number of fits per event. The issue of how much computing is required for the final physics analysis is a very important one. Table 2 shows estimates of the CPU cycle requirements in VAX-years for several Fermilab experiments. It is clear Fermilab could never support this activity at the cost of \$50,000 per VAX equivalent that we paid for our last mainframe acquisition. Even the present day cost of a Microvax III type machine, approximately \$1500/VAX780-year leaves us with a bill of many millions of dollars. Figure 2 shows the cost of computing on various platforms. It was clearly the low cost of cycles on the RISC processors and the steep derivative which aroused our interest in these processors. Because they all seem to run UNIX, we were forced to get interested in that as well. We will discuss later the nature of the connection between RISC and UNIX.

When we look back over the last year, one of the major turning points in

Table 2: Estimated CPU requirements for first pass event reconstruction in 1990-1992 time frame for selected experiments

group	total VAX-years for reconstruction
CDF91	1200
E665	350
E687	400
E771	2000
D091	250
E791	2000-5000

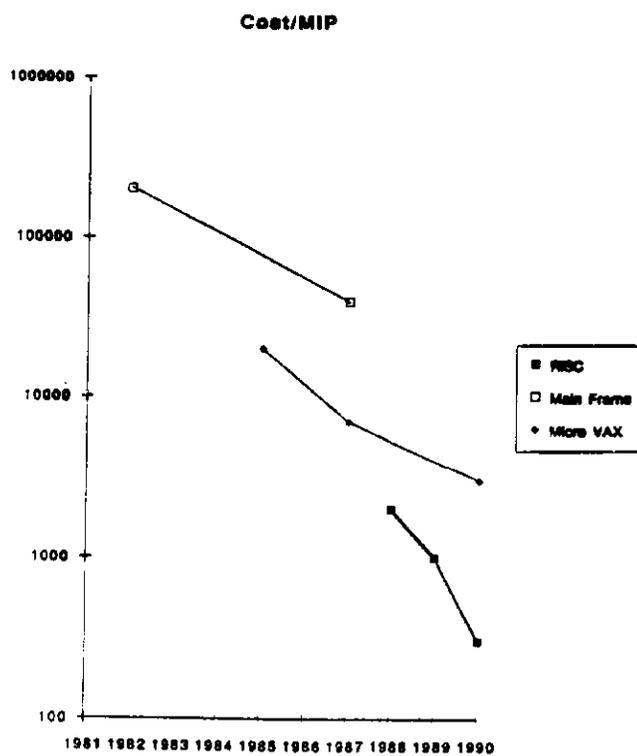


Figure 2: Cost of Computing Cycles

computing at the lab was CDF's decision to use RISC/UNIX machines in their level three trigger. Since the desire was to make selections based on a very sophisticated calculation, this required the collaboration to commit itself to porting a significant fraction of all its code. Early in this process, CDF reached the conclusion that it should bite the bullet and port its entire offline analysis, which is by far the biggest code at Fermilab.

2.2 UNIX for interactive programming and graphics applications

There is a second front for UNIX at Fermilab. This beachhead is not in the HEP physics data analysis area but in the area of technical and scientific computing and drafting. Fermilab has a great number of CAD users, drafters doing drawings and engineers doing solid modelling and finite element analysis. These activities require major CPU cycles and good graphics capabilities. Most of these groups started with VMS and used commercial products such as IDEAS from SDRC. As they begin to suffer from poor response time, they are beginning to move to more powerful UNIX systems which can run the same software. These groups do not do very much applications development. Other groups, such as the accelerator theory group and the astrophysics groups, that develop applications which employ sophisticated graphics and require significant CPU resources also use UNIX platforms. Whether the issue is interactive response or availability of 'production batch' cycles, the choice is driven more by cost issues than a basic love of the UNIX operating system

The point of all this is that the progress of UNIX systems in Fermilab is driven from the two ends of the analysis chain-- the compute intensive reconstruction demand at the top of my picture and the graphics application on the desktop. We do not know whether UNIX will achieve dominance for physics analysis. It probably will for those users who are CPU limited on MicroVAX type processors, but those who are IO limited will probably not feel any pressure to switch. The middle ground in the analysis model involves heavy 'event' and file serving. The application of UNIX engines to this area is very uncertain. VMS or VM may offer a fine way to provide data and file service to both VMS and UNIX workstations and may achieve dominance in that area.

2.3 The relation between RISC and UNIX: accident or inevitability?

All the reasons why UNIX was making progress at Fermilab were tied not to desirable features of the UNIX operating system but to its association with the new RISC technology. In looking towards the future, one is compelled to ask whether this association was an accident or whether it was an inevitable match. One can then ask whether this situation will persist or whether it is a transient phenomenon.

We would argue that the match is inevitable and will endure for the foreseeable future (i.e. the next few years). UNIX achieved its acceptance during a period of rapid development of new CPU's. If one compares the time it takes to develop a new generation of CPU compared to the time it used to take to write a proprietary operating system, one sees that certain characteristics of UNIX make it especially suitable for a period of rapid introduction of new technology. These are:

- Its portability. There are several aspects of this. First, you can get it running on your system in a short amount of time so that you introduce complete systems at the pace determined by the technology. Second, if you introduce a 'better' machine, people can move to it easily. They are not locked in by their massive development in existing, non-portable software. This is, of course, a double-edged sword.
- Its availability. It is easy and inexpensive to get the UNIX source code and the right to use and modify it. You have an almost open environment.
- Its suitability. Its not a bad system and many of its problems can be covered up with 'value-added' user interfaces--just the area where vendors like to differentiate their products anyhow.

These are advantages for the vendor. The user realizes many advantages as well. He can move between systems as tastes and needs change and as new products offer new opportunities. He is not bound to one company which might lose its creative energy or move in a direction which is not interesting him. Competition will certainly result in lower prices and better products. With the potential for a bigger market, more software developers will write code for these systems.

We believe, then, that UNIX possessed features that guaranteed that it would succeed in a era of rapid development of CPU's. That trend will continue so we think UNIX, or some successor which emphasizes portability and availability, will continue to grow in importance.

In this section, we have emphasized the attractive features of this dynamic, competitive environment. It has a downside that will be discussed in a later section.

3 The UNIX environment: The Good, the Bad, and the Ugly

In this section, we discuss the UNIX system from the perspective of an unsophisticated UNIX applications programmer(me) and from the viewpoint of a UNIX system administrator (several of my colleagues at FNAL).

3.1 Good aspects of the UNIX environment

3.1.1 Intrinsically good features about UNIX

UNIX was originally written to support a software development environment. It has many strengths with respect to other operating systems, including VMS. Some of these are:

- The ability to sit at a system from a vendor you're not accustomed to and find that the system works pretty much the same way your own system works. There will certainly be differences and they shouldn't be minimized but usually there is enough commonality in the way the two systems are organized so that by snooping around looking in directories you can get a good idea of what resources are available and how they are organized. The various 'UNIX'es (pronounced UNICIES ??) are like different dialects of the same language rather than different languages.
- An emphasis on extending the base operating system with user defined command procedures. Commands are executed from a search list, called a 'PATH', which can include directories of user supplied command files or applications. The emphasis on 'tool building' is deeply rooted in the UNIX outlook.
- An emphasis on using low level tools to build higher level tools. Constructs like 'pipes' and 'filters' together with very powerful text processing, command interpreting , and lexical analyzing facilities permit one to make very powerful procedures quickly.
- Ease of output and input redirection. It is very easy to send output to a file from a screen-oriented application or to send input from a file to an application that was written for interactive input. This is more natural, and therefore more frequently used, than the approach of redefining logicals to redirect IO.
- The ability to run processes in 'background' while continuing your interactive session. Used in conjunction with IO redirection, this makes it easy to keep several parallel activities going. UNIX, in general, expects a user to have many processes going at once and makes liberal use of subprocesses in its normal operation.
- The availability of a powerful utility for generation of up to date executables with minimal recompilation- the make utility.
- The availability of a 'preprocessor' for source code, the C preprocessor, which can also be used to advantage even in a FORTRAN environment.
- The incorporation of performance profiling tools in a very straight forward way into the compiler and run time environment.

- The ability to 'archive' files in a standard format which allows them to be read by UNIX systems from other vendors.
- Good network access. UNIX networking, based on the TCP/IP protocol, has developed rapidly and is well supported by facilities available to the user. 'Trusted host' operations, such as remote login 'rlogin', remote file transfer 'rcp', and remote procedure execution 'rsh' are widely used. Data may be shared between systems by products such as NFS which are widely available. Support of these protocols is now available on VAXes so that access to data on VAX/VMS systems is available. 'Non-trusted' host tools, such as Telnet and FTP, are also available.
- Task-to-task communication across the network. Socket interfaces are reasonably easy to implement. I have seen more interest amongst users in doing this on UNIX systems for database access than I observed on VMS systems using DECNET task-to-task communication.

Many of these features of UNIX are available in VMS. The features noted above seem to me to be more 'natural' in their UNIX form than in their VMS incarnations. It has been my experience that physicist/programmers have an easier time getting started on a VMS system but rapidly plateau in their capabilities. In UNIX systems, it takes longer to get started but people routinely achieve competence with powerful facilities such as MAKE and the PROFILER. The corresponding tools in VMS, CMS/MMS and PCA, are rarely mastered by the average physicist/programmer.

3.1.2 Features that come from RISC

There are other aspects of programming on UNIX systems that emerge from their close association with RISC systems. Since the connection is not accidental, these should be counted as advantages of the UNIX environment. These include:

- The availability of efficient FORTRAN compilers which support most of the popular VAX extensions.
- The speed with which compilation and linking can be done.
- The speed with which bugs can be 'reached'.
- The speed with which graphics can be generated and screens updated.
- The availability of of standard Graphical User Interfaces and the CPU power to use them.
- The availability of graphical interfaces to the UNIX system itself which use pull down menus and other such devices to eliminate the need to remember the somewhat obscure UNIX commands and options.

- The ability to support several windows efficiently.

These advantages exist mainly because the most powerful CPU's available for interactive use are RISC/UNIX systems.

3.2 Bad stuff about UNIX

The UNIX environment has many frustrating features. Some of these are mere annoyances that can be routinely overcome after a few days of use. Some, however, have lasting impact and can result in lost productivity.

From an application programmer's viewpoint the most annoying features are:

- **Obscure command names.** Command names such as 'grep', 'biff', 'awk', 'df', 'du', 'ls', 'chmod', etc. are not very helpful. Some have interesting stories behind them. I have not found this to be a terrible problem since you can 'alias' sensible names to them and, anyway, you get used to them after a while.
- **Command non-orthogonality.** Many of the commands have overlapping functionality and limited 'scope'. This sometimes gives you several options to choose from which means you also have more ways to make mistakes.
- **Totally obscure and inconsistent options.** Commands may be followed by one or more options. No attempt has been made to make the options consistent across commands. For example, in the command, 'ps', which returns process status (not too bad)

```
ps -u myname
```

the 'u' gives me the status of processes run under 'u'sername 'myname'. That seems to make sense. However, in the remote login command, 'rlogin', the option which allows you to supply a username (so you can login to the remote system under a different user name) is '-l'. Under VMS, when such a parameter is required it appears rather consistently, in many commands, as

```
/username=myname
```

Moreover, options seemed to have been added by the requirements of the moment and frequently you can't do what you want even though you could probably do almost everything else. I view this as a serious problem. You can 'alias' your favorite command strings including options but this gets messy.

- Case sensitivity. This is new to VMS and VM users and results in some problems but you eventually get used to it and may even use it to advantage.
- ‘Launch and Pray’ command line editing. The command line editing is unnecessarily obscure but the main problem is that after you issue what looks like a ‘old fashion’ line editor (TECO-like) command, you can’t see the new command before it executes without doing really arcane things. This problem was fixed by providing a procedure, ‘cedit’, at Fermilab to make ‘VAX-like’ command line editing available under the C-shell. This was not very difficult to achieve but apparently is in violation of the ‘right way to do UNIX’. Fortunately, at Fermilab we haven’t been able to locate the **man** page on ‘the right way to do UNIX’ so we can move ahead in our ignorance. Which brings me to my next point.
- Really crummy documentation. The **man** pages are not all that great. The options, which is what you really usually want to know about, are not alphabetized!! There was a rumor that someone had invented the concept of ‘the example’ in UNIX **man** pages but it turned out to be unfounded. (You can fix this by reading ‘UNIX for VMS Users’ by Philip E. Bourne[1], ‘The UNIX Programming Environment’ by Kernighan and Pike[2], and many other excellent books). In the former, the author attempts to ease the shock of transition for VMS users by actually giving examples of commands and procedures.)
- The limit of one version of each file. This turns out not to be such a terrible drawback and probably saves some disk. When I started using UNIX systems, I thought this would be a very severe problem.
- The failure to support filename extensions. You can legally include ‘.abc’ at the end of the filename but the system utilities, for the most part, do not support any system of defaults. Many groups adopt standard ‘extensions’ for use on UNIX systems so that they can keep track of their files even though they have to keep typing the extra characters all the time. The groups invariably choose extensions which look suspiciously like ‘default’ VAX filename extensions.
- Filename length limitations. UNIX originally limited filenames to 14 characters. Most systems have done away with this but some still have utilities that do curious things if the filename exceeds 14 characters.
- Very serious limitations in the way you can set up your environment. For example, UNIX supports ‘environment variables’ which can act sort of like VAX Logical names, but the space available for them is limited to 5120 characters/process. OK for a single, lonely user. Not good for someone trying to set up the ‘CDF environment’, for example.

- A somewhat strange view of file structure. UNIX files are viewed by the systems as 'byte streams'. This is simple and elegant but totally inconsistent with, for example, FORTRAN's record oriented approach to files. FORTRAN covers the distinction up for you and makes it look like there are records. But there are some curious effects. For example, there are no 'headers' in the file system that tell you about record structure so you had better remember how you wrote those files. You run into situations where you move files from a VAX to a UNIX system (loosing some information about the structure on the way), then move them back and discover that you can no longer read them (at least in the same way).
- A somewhat strange view of devices. Every device is associated with a file and all operations on the device are treated as operations on the file. This produces some really strange behavior. One gets used to this but it can be very confusing to a novice.
- The occasional tendency of the system to move you out of the 'shell' (command interpreter) that you are working in- at Fermilab typically the C-shell into some other shell, usually the Bourne shell. This is akin to falling out of VMS and into VM.
- The occasional tendency of the system to move your job into 'background'. To a novice, this is like falling off the end of the earth.

3.3 The UGLY: Things either completely missing or done so badly that you wish they were

Under the category of things missing completely or done really badly we have:

- Security. This is the stuff out of which legends are made. Also, reputations! You can read about this in 'The Cuckoo's Egg' by Clifford Stoll[3] and in many other popular books. Many obvious holes have been closed. But, be warned that many UNIX systems come 'out of the box' with most of their security features disabled. Specific issues are eight character passwords, readable passwords in several places on the system, and lack of fine-grained granularity in assigning privileges.
- Cluster management: UNIX does not have a true 'cluster' scheme. This places a major burden on support staff. This is somewhat mitigated by the fact that much of the infrastructure for making a kind of cluster environment, such as NFS and Yellow Pages, are in place and solve some of the problems. Much better facilities are incorporated into SVR4 and are planned for future releases. The keepers of UNIX have acknowledged the problem and are moving towards solving it. They are not there yet.

- System management. Utilities are often poor. There are no individual user disk quotas and no ACL's (in some UNIX'es). Many of the utilities have 'bugs' which are not fixed but simply documented and achieve the status of features. Some parameters which should be adjustable are 'hardwired' into the kernel and would require a rebuild (assuming you had a source license).
- There is no real batch system (although third party products are becoming available).
- There is no real resource management or allocation scheme.
- Tape support is very poor. There is no tape drive allocation scheme. There is no support for labelled tapes in the system (so this is done from within the application).
- Backup is cumbersome especially in a large system of workstations.
- There is really no one who takes responsibility for the problems with UNIX commands and features since 'your' vendor didn't write it.
- There are two competing organizations trying to determine the future development of the system.

3.4 How does it all add up?

We have come to believe at Fermilab that UNIX is 'acceptable' and that we can live with it. We are drawn to it because of its association with the only cost-effective solutions to part of our computing problem and NOT BECAUSE OF ITS INHERENT DESIRABILITY AS AN OPERATING SYSTEM. In the grand balance sheet, it not so bad as to erase the economic advantage that it brings.

4 The Joys and Agonies of the OPEN MULTI-VENDOR environment

Once Fermilab decided to support UNIX, it became inevitable that we were going to support more than one vendor and, therefore, several 'UNIX'es. The reasons for this are:

- Law. We are bound by competitive bidding rules so it was impossible to really restrict ourself to a single vendor even if we had wanted to. In fact, Silicon Graphics and SUN each won major bids in the early going.

- Opportunity. Once you build a 'UNIX infrastructure', the cost of adding an additional vendor, while not by any means negligible, is not large compared to your original investment and, at least within limits, you can reap some benefits by promoting competition for your business. Much more will be said about this in the remainder of this section.
- Circumstance.
 1. A national lab like Fermilab is part of an 'extended' enterprise which includes other laboratories around the world and many universities. Choices made by these institutions, especially our university collaborators, must affect what we choose to support because making them efficient is one way to advance the lab's scientific goals.
 2. A national lab like Fermilab is composed of many groups which have their own unique needs and possess the computing skills to reach their own conclusions on what platforms are best for them.

The sum of all these forces proved to be irresistible and Fermilab has already become a multi-vendor lab for UNIX.

Whatever the perceived benefits of supporting a multi-vendor, multi-platform environment, there are many, many problems which should not be ignored or minimized. Obviously, there are major support problems. The basic point is that the answer to every simple question is now a 'vector' and the answer to every complex question is at least a two-dimensional 'matrix' or table. Tom Nash refers to these as 'MATRICES OF INCOMPATIBILITY'. Every entry into one of these matrices represents a significant effort. A few examples will illustrate this point.

4.1 How good is your FORTRAN compiler?

This question has several components. How many bugs are known to exist in the present release? How many will you discover as you port your code? Which VAX extensions are supported and are they REALLY supported? Is the FORTRAN 'environment' adequate to the largest codes you anticipate running and does it have all required facilities? How efficient is the compiler for your application? Each of these questions must be answered for each vendor.

In order to get even an approximate answer to these questions, you must develop a 'benchmark suite' of applications which represents the typical codes you really plan to run and you must develop a checklist of useful features and extensions which you can verify. We did this for a recent acquisition and it took several months and the cooperation of many people. For example, this is the collection of 'benchmark codes' which were contributed:

1. 'QCD calculation', a code that emphasized floating point calculations in a few tight loops.

2. GEANT simulation of the D0 detector-- a very large computationally complex code which reflects the activities of a large collider program.
3. E691 tracking code-- a 'fixed target' event reconstruction code. This read from and wrote to 8mm tape.
4. E400 tracking code-- a 'fixed target' event reconstruction code. This read to and wrote 8mm tape.

There are many other worthy candidates for inclusion into such a suite. We may add to it in the future. We are also likely to get some 'standard' suites-- like the SPECMARK suite-- for comparison. It is important to note that we were doing much more than just testing compiler and machine performance. We were checking that the platform supported the peripherals we were interested in and that it supported very large codes. We were, in particular, very concerned about the ability to handle programs with many thousands of subroutines and to successfully read and write 8mm tapes. We have developed a standard list of mandatory and desirable properties of the so-called 'developer's environment' to make sure that the system is really 'useable' in an HEP environment. This includes, for example, a list of third party products that we expect to be available.

At Fermilab, we have a large number of VMS-based applications programmers. These people use VAX extensions in their code. We can hardly fault them on this since most vendors have acknowledged the value of these extensions and supported some or all of them in their FORTRAN compilers. Moreover, most of these extensions have been included at least functionally in Fortran 90, a tacit acceptance of the reality that VAX FORTRAN has been a de facto semi-standard for the late 1980's. Fermilab attempted to define a set of 'mandatory' extensions for acquisitions and a set of 'semi-mandatory' extensions (you were allowed to miss two) based on a study of various compilers. Below are lists of the mandatories and semi-mandatories and a table showing which VAX extensions, of this subset, are NOT supported on each of several systems. There is much more variability in the VAX extensions that fell outside this list.

A program, called Alice, was written to verify that these extensions really worked as claimed[4].

Of course, even this table doesn't tell the whole story. Something as simple as the 'OPEN' statement has all kinds of 'features' that affect its 'portability'. One amusing one (except to people who discovered it) is that the one compiler uses a 'byte-count' for all logical record lengths whereas a VAX sometimes used a byte-count and sometimes a (long) word count. However, you can make the compiler look 'just like a VAX' in this respect by using a special compiler switch.

Table 3: Fermilab Mandatory Requirements

- 1 Symbolic names
- 2 INCLUDE statement
- 3 Data Types INTEGER*2, REAL*4, REAL*8
- 4 Optional DO-loop Label
- 5 Cross-language Access
- 6 Read/Write 32,760 bytes
- 7 Read/Write Unblocked Tapes

Table 4: Fermilab Semi-Mandatory Requirements

- 1 Comment indicator
- 2 Debugging Statement Indicator
- 3 Tab format lines
- 4 Byte Data Type
- 5 Hex, Octal Data Constants
- 6 Mixed COMMON BLOCK
- 7 IMPLICIT NONE statement
- 8 NAMELIST Statement
- 9 DO WHILE Statement
- 10 Hollerith Constant
- 11 Built-in Functions %VAL() and %REF()
- 12 O Field descriptor
- 13 Z Field Descriptor
- 14 Bit Manipulator Routines
- 15 ANSI Flag
- 16 Profiler Utility

Table 5: Fortran 'semi-mandatory' evaluation for various vendors

system	Failed Feature
SUN	Hexadecimal Constant
DEC Ultrix	Hollerith constant, ANSI flag
IBM AIX	INTEGER*2 or similar function type, TAB formatting
SGI	Hollerith constant, ANSI
Sony NEWS-OS	NAMELIST, Hollerith Constant, ANSI flag

	VMS	Amdahl
CERN	X	X
DI-3000	X	X
TopDrawer	X	X
GKS	X	X
Printing	X	X

Figure 3: 1989 Graphics support matrix

4.2 How do you support graphics applications under UNIX?

Graphics needs to be available on a variety of UNIX workstations. Even those platforms that are being used mainly as reconstruction farms need to have development environments and people immediately request graphics for histogramming and event display. These tools can really facilitate program debugging and validation. Figure 3 shows our 'graphics' support matrix about a year and half ago. Figure 4 shows our present 'graphics' support matrix. Life ain't getting no simpler!!!

4.3 How does 8mm tape support work on UNIX systems?

The use of 8mm tapes for recording data and the output of reconstruction passes is now 'standard' at Fermilab. This medium is also employed at the workstation level because the drives are so inexpensive and they are also used for system backup of both central and local systems. 8mm is now available on every centrally supported Fermilab platform, including the Vax Clusters and the AMDAHL, and on ACP I farms. We are therefore required to understand the behavior of these devices on UNIX platforms. There are many aspects of this problem. There are two modes of recording- fixed and variable block. There are two kinds of file marks- long and short. There are several options for controllers. Firmware is always changing. Figure 5 shows a matrix, one of many that must be considered, of '8mm features seen in different environments', at one particular

	VMS	Amdahl	SGI	SUN	Ultrix	IBM
CERN	X	X	X	X	X	X
DI-3000	X	X	X	X	X	X
TopDrawer	X	X				
DEC-GKS	X				X	
SUN-GKS				X		
GK-3000	X	X	X	X	X	X
PHIGS	X	X		X	X	X
Printing	X	X	X	X	X	X
X Windows	X		X	X	X	X
X terminals	X		X	X	X	X

Figure 4: Present Graphics support matrix

moment in time. Some of these entries are now already out of date. Figure 6 shows another matrix of how various platforms interface with various controllers and firmware versions. The point of this is that much more work has to go into these efforts if we really want to capitalize on the opportunities represented by the open UNIX environment and the open peripherals marketplace.

4.4 The MULTI VENDOR environment revisited

The point of presenting all these tables is that you have to work hard to realize the benefits of a multivendor environment. While this may seem like an obvious point, there are two reasons for harping on it:

- It may not be obvious how difficult it really is. If you underestimate the problem, you will surely get yourself in trouble. If you overestimate, you will scare yourself out of real opportunities.
- It may be very hard to control the number of platforms no matter how hard you try, for the reasons described above.

8mm Features as Seen in Different Environments*

* ABSTRACT
ENOUGH
TO BE
IMPRECISE

	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
READING/WRITING DATA																	
Fixed Blocks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Variable Blocks	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Maximum Block	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Minimum Block	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Know Capacity	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Select Byteswap	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Examine Low-level Errors	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Control Timing	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ERASEABILITY																	
Choose Short or Long File Marks	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
"Clever" Host Adaptor	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GENERICITY																	
Use "Commodity" Drives	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FILE STRUCTURING																	
Native ANSI Label Support	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Add-on ANSI Label Support	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Unstructured Support	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
OPERATING SYSTEM SUPPORT																	
Suitable for Backups	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Restore System Disk	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

5 Major issues for the future

5.1 How extensively will UNIX/RISC systems permeate the analysis model?

At the moment, UNIX systems are dominating the reconstruction and Monte Carlo enterprises at Fermilab. They are making great progress on the physicist's desktop and for physics analysis at Fermilab's collaborating institutions. They are 'naturals' for compute servers for workgroup clusters. We can not tell whether they will succeed at 'event/file serving', especially where VMS file compatibility is required, but UNIX networking seems to be leading the way towards this. I don't think the issue is yet settled but it would not surprise me if UNIX came to dominate all aspects of the analysis model at Fermilab. Even if it doesn't take over the 'event/file server' business, it is clear that whatever system does must connect easily to systems running UNIX.

5.2 How will UNIX develop?

The UNIX community is attempting to address many of the same issues that concern us. The workgroup cluster concept, discussed by Tom Nash, is being developed by several institutions. The recent discussions of 'PLAN 9'[7] center on just the same issues that ought to concern us.

5.3 How will users adjust to UNIX?

There are basically three ways to cope with the deficiencies of the present UNIX environment:

1. Learn to love it as an 'acquired taste'.
2. Fix what we don't like. We can do this because we don't have any loyalty to UNIX. This would include:
 - Hiding the variations, in the manner of the FERMI UNIX ENVIRONMENT[5].
 - Fixing the deficiencies with a layer of procedures, such such as Fermilab's 'cedit' program.
 - Adding third party products such as EDT+ or NQS to improve what is there or provide something which is missing.
 - Developing what is missing, such as labelled tape support.
3. Hide it. This can be done by developing 'application environments' which sit on top of the operating system and shield the user from it. The CERN Physics Analysis Workstation (PAW) package [6] is an example of this approach.

5.4 How can we control and mitigate the escalating support requirements?

5.4.1 Limit the number of platforms

Fermilab has decided to limit the number of 'fully supported' UNIX platforms to three. It may be necessary to add partial support for a fourth. We will review the supported systems periodically to understand whether one ought to be dropped and another added. We will undoubtedly forgo some 'opportunities' but hope that competition in the marketplace will force our selected vendors to stay attractive. The association with three or four companies should assure that at least some will stay dynamic.

5.4.2 How can we lighten the load?

The key ways that we can manage the situation are:

- Devote some resources to constant evaluation of new platforms, products, etc. Keep on top of your preferred vendors and any potentially interesting competitors. You have to know the market, if for no other reason than to defend your choice of selected vendors.
- Communicate information widely, both formally and informally, so studies don't have to be repeated within the organisation.
- Pursue sources of information at other labs, universities, and with industry wherever possible.
- Try to standardize the systems support environment as much as possible. This includes adduser utilities, backup utilities, printing, etc. The FERMI UNIX ENVIRONMENT addresses this issue.
- Try to develop a standard user environment so the different flavors of machine look similar to an applications programmer. The FERMI UNIX ENVIRONMENT addresses this issue, as well. This helps the users and simplifies support and consulting.
- Adopt proprietary tools which can be made available on all supported platforms. Try to chose tools the will be within the budgets of university groups. Always support at least one solution that is non-proprietary.

Each of the items above could be applied across the HEP laboratories as well. If we can work in collaboration, we can achieve two major goals:

- We can distribute the load so that we don't wind up supporting the same functionality several times.

- We can create an environment in which physicists, computer scientists, and engineers can move from lab to lab and work with a familiar environment. This will be especially important in the SSC/LHC era where people will be working on existing programs at Fermilab and LEP while participating in start up efforts at the new machines.

6 Conclusion

The UNIX world is filled with opportunity. The low cost and high power of these systems can open up new possibilities at any stage of the data analysis. To realize these advantages, we must apply more effort to evaluation and support. We must chose carefully so that we can stay within the limits of the available manpower. I would like to summarize the situation by reminding you of the caption from an old POGO cartoon:

"We are confronted with unsurmountable opportunities"

At Fermilab and elsewhere, we are beginning to overcome the obstacles and realize the opportunities. HEP is already benefitting and will continue to do so. It is worth the effort!

Acknowledgements:

Many people at Fermilab have contributed to my knowledge of UNIX and of the issues discussed in this paper. I would like to thank the following groups: ACCESS Liaison Group (leader Judith Nicholls); Online Support Group (leader Ruth Pordes); Central Computing Department (leader Peter Cooper); Distributed Computing Department (leader Al Thomas); Physics Analysis Tools Group (leader Paul Lebrun); and Computer Research and Development(leader Joseph Biel). The fact that this list includes every software group in the Fermilab Computing Division reflects the breadth of our effort on UNIX. I would like to thank the following individuals for specific discussions that were used in preparing this paper: Frank Nagy; Tom Nash; Randy Herber; Matt Wicks; Don Petravick; Dick Adamo, Jack Pfister, Irwin Gaines, and Vicky White.

References

- [1] Phillip E. Bourne, UNIX for VMS Users, Digital Press, 1990
- [2] Brian Kernighan and Rob Pike, The UNIX Programming Environment, Prentice-Hall, New Jersey, 1984
- [3] Clifford Stoll, The Cuckoo's Egg, Doubleday, New York, 1989

- [4] F. Abarghoui, O. Evans, U. Pabrai, H. Shah, FORTRAN Compiler Evaluation of Fermilab Mandatory Requirements, Fermilab, TN0057, 1991
- [5] J. Nicholls, Fermi UNIX Environment, contributed paper to CHEP91, Tsukuba, Japan, 1991
- [6] R. Brun, O. Couet, C. Vandoni, P. Zanarini, PAW, Physics Analysis Workstation, CERN Computer Center Program Library, Long Writeup, Q121
- [7] R. Pike, D. Presotto, K. Thompson, H. Trickey, Plan 9 from Bell Labs, AT&T technical memorandum