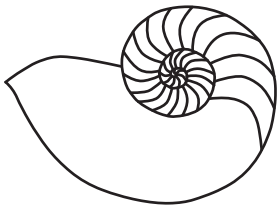
Volume 6, Number 3 January 1994 $2.50



MUUG Lines

Manitoba UNIX® User Group

Newsletter of the Manitoba UNIX® User Group

A Very Brief Look at Unix History

By Pierre (P.) Lewis

Submitted by Andrew Trauzzi

Unix history goes back to 1969 and the famous "little-used PDP-7 in a corner" on which Ken Thompson, Dennis Ritchie (the R in K&R) and others started work on what was to become Unix. The name "Unix" was intended as a pun on Multics (and was written "Unics" at first — UNiplexed Information and Computing System).

For the first 10 years, Unix development was essentially confined to Bell Labs. These initial versions were labeled "Version n" or "Nth Edition" (of the manuals), and were for DEC's PDP-11 (16 bits) and later VAXen (32 bits). Some significant versions include:

V1 (1971): 1st Unix version, in assembler on a PDP-11/20. Included file system, fork(), roff, ed. Was used as a text processing tool for preparation of patents. Pipe() appeared first in V2!

V4 (1973): Rewritten in C, which is probably the most significant event in this OS's history: it means Unix can be ported to a new hardware in months, and changes are easy. The C language was originally designed for the Unix operating system, and hence there is a strong synergy between C and Unix.

V6 (1975): First version of Unix widely available outside Bell Labs (esp. in universities). This was also the start of Unix diversity and popularity. 1.xBSD (PDP-11) was derived from this version. J. Lions published "A commentary on the Unix Operating System" based on V6.

V7 (1979): For many, this is the "last true Unix", an "improvement over all preceding and following Unices" [Bourne]. It included full K&R C, uucp, Bourne shell. V7 was ported to the VAX as 32V. The V7 kernel was a mere 40 Kbytes!

These Vn versions were developed by the Computer Research Group (CRG) of Bell Labs. Another group, the Unix System Group (USG), was responsible for support. A third group at Bell Labs was also involved in Unix development, the Programmer's WorkBench (PWB), to which we owe, for example, sccs, named pipes and other important ideas. Both groups were merged into Unix System Development Lab in 1983.

Work on Unix continued at Bell Labs in the 1980s. The V series was further developed by the CRG (Stroustrup mentions V10 in the 2nd edition of his book on C++), but we don't seem to hear much about this otherwise. The company now responsible for Unix (System V) is called Unix System Laboratories (USL) and is majority-owned by AT&T. Novell bought USL in early 1993.

But much happened to Unix outside AT&T, especially at Berkeley (where the other major flavor comes from). Vendors (esp. of workstations) also contributed much (e.g. Sun's NFS).

The book "Life with Unix" by Don Libes and Sandy Ressler is fascinating reading for anyone interested in Unix, and covers a lot of the history, interactions, etc. Much in this article is summarized from this book.

This Month's Meeting

Meeting Location:
Our next meeting is scheduled for Tuesday, January 11, at 7:30 PM. Once again, the meeting will be held in the auditorium of the St-Boniface Hospital Research Centre, just south of the hospital itself, at 351 Taché. You don't have to sign in at the security desk — just say you're attending the meeting of the Manitoba UNIX User Group. The auditorium is on the main floor, and is easily found from the entrance.

Meeting Agenda: See inside for details.

Inside This Issue

Newsletter Editor's Ramblings 2
President's Corner 3
QNX: A Realtime UNIX 4
C++ Q & A 6
UNIX Q & A 8
OS/2 2.1: UNIX Utilities Invade 9
SIG Sideline 10
Jan 11th Meeting Agenda 10

UNIX is a registered trademark of UNIX System Laboratories, Inc.

New Year, New Attitude?

By Andrew Trauzzi

First of all, happy new year everyone! I'm sure many of you have made new year's resolutions — some you will keep, and some you won't. My new year's hope is that I will become less pessimistic about the computer industry and the direction it takes. I am pleased to see results from the alliances formed between companies (e.g. PowerPC, Taligent, etc.), and I hope that 1994 will be a banner year for them.

This month's speaker will be discussing one of these alliances — COSE. I would like to quickly bring you up to date on the COSE story (aside from all the bad puns).

COSE

COSE (The Common Operating Software Environment) is an effort by the major UNIX vendors to create a standard desktop environment that provides users with a common look and feel. COSE's members include: HP, IBM, USL, SCO, Univel, DEC, and SunSoft — all coordinated by the X/Open consortium (the people whom Novell trust to further develop UNIX). The final product will include components from HP's visual environment, IBM's Common User Access mode and Workplace Shell, Open Software Foundation's Motif and Window Manager, SunSoft's OpenLook and Deskset productivity tools, and UNIX SVR4.2 Desktop Manager from Unix Systems Laboratories.

Last June, COSE released a 100 page specification document outlining the incorporation of the products listed above. The spec also outlined some of the tools that developers required in order to make UNIX more popular such as: basic administration, data editing and display, desktop integration, object/folder management, and window management. The main complaint with the spec is that it had nothing concrete to say. As one analyst put it "This is one of the biggest non-announcements I've seen in my time!" Furthermore, the tools outlined in the spec are already kind of "standard", such as X-Windows and Motif.

Spam?

Many people are asking themselves: "Why would all the UNIX vendors want to get together?" Most people agree that COSE was formed to unite UNIX vendors against Microsoft's muscle and all the alpha-beta-beta-spam-beta versions of NT and/or Cairo. I think that the biggest obstacles to COSE are the vendors themselves. Can you imagine so many vendors with such varied interests and priorities agreeing on standards? The technical task of standardizing UNIX seems far easier than attempting to resolve the political issues at large.

Look for COSE-approved technologies in the second half of 1994. If you want a copy of the spec, send a request to:

<XoCdeSpecs@xopen.co.uk>.

Did I sound a little pessimistic? Oh well, old habits die hard! ;)

The 1993-1994 Executive

President:	Bary Finch	(W) 934-2723
Vice-President:	Ramon Ayre	(W) 947-2669
Treasurer:	Rick Horocholyn	(W) 474-4533
Secretary:	Brad West	(W) 983-0336
Membership Sec.:	Greg Moeller	(H) 786-6132
Mailing List:	Roland Schneider	1-482-5173
Meeting Coordinator:	Roland Schneider	1-482-5173
Newsletter editor:	Andrew Trauzzi	(W) 986-6009
Publicity Director	Rory McLeod	488-5168
Past President	Susan Zuk	(W) 631-2530
Information:	Bary Finch	(W) 934-2723
		(FAX) 934-2620
(or)	Andrew Trauzzi	(W) 986-6009
		(FAX) 986-5966

Copyright Policy and Disclaimer

This newsletter is ©opyrighted by the Manitoba UNIX User Group. Articles may be reprinted without permission, for non-profit use, as long as the article is reprinted in its entirety and both the original author and the Manitoba UNIX User Group are given credit.

The Manitoba UNIX User Group, the editor, and contributors of this newsletter do not assume any liability for any damages that may occur as a result of information published in this newsletter.

Advertising Rates

Quarter page	$50
Half page	$75
Full page	$100
Insert (1-4 pages)	$100

Above prices are per issue. The first ad is charged at the full price; each successive month is 1/2 price.

Ad copy must be submitted by the final copy deadline for an issue (usually 3 weeks prior to the monthly meeting) in a format acceptable to the editor. (Please make arrangements with editor beforehand.)

Internet E-mail: editor@muug.mb.ca

Group Information

The Manitoba UNIX User Group meets at 7:30 PM the second Tuesday of every month, except July and August. Meeting locations vary. The newsletter is mailed to all paid-up members one week prior to the meeting. Membership dues are $25 annually and are due as indicated by the renewal date on your newsletter's mailing label. Membership dues are accepted at any meeting, or by mail.

Manitoba UNIX User Group
P.O. Box 130, Saint-Boniface
Winnipeg, Manitoba R2H 3B4

Internet E-mail: membership@muug.mb.ca

PRESIDENT'S CORNER

The New Year

By Bary Finch

Well now I have to get used to writing 1994. This is always an interesting challenge which may take a few weeks and a few spoiled cheques to perfect. But it is a new year and that always carries a sense of optimism with it. We have a lot to look forward to in this new year, but first I'd like to wrap up last year.

We held our annual Wine and Cheese as our December meeting and had a good turnout. There was a great selection of wines, cheeses, and other delectable munchies, and we have Roland Schneider to thank for that. He coordinated the liquor and liquor license as he would in his new role of Meeting Coordinator. However, he went all out and prepared all the cheeses and other food as well. Thanks again for all the help!

I mentioned Roland taking over as Meeting Coordinator as Paul Hope has had the luck/misfortune of too much happening at work. This is actually a promising sign these days, to know that you are a needed part of the organization. Paul will continue to arrange for our meetings to be held at the St. Boniface Research Center. Thanks for all your contributions as Meeting Coordinator!

Display!

Another big thank you goes out to Andrew Sametz of The CD Factory. He provided an excellent display of CD-ROM cutting technology (no pun intended). He also had the unexpected delight of being the sole display present. We had several other displays have to cancel at the last minute due to technical difficulties (really!!).

And that wraps up 1993 with a successful celebration. Now we continue into 1994 with a great series of upcoming meetings to keep you informed on the current and emerging technologies.

Our January meeting topic is being provided by Hewlett-Packard. Their speaker will be talking on COSE, the Common Open Software Environment. This is one of the evolving efforts in the UNIX industry to standardize on a common front end to all UNIX operating systems, or as many as possible. This presentation should be of great interest to anyone who works with UNIX.

Our February presenter will be Smoot Carl-Mitchell of Texas Internet Consulting. This will be part 2 of our presentations on the Internet, where part 1 was presented by Dr. Roger Taylor. Smoot Carl-Mitchell is a consultant on network management, as well as a regular contributor for magazines. This should be an outstanding presentation, so book February 8 on your calendars now!

Future Topics

To continue our discussions of emerging technologies, we will have a presentation on the new PowerPC chip for our March meeting. This will be provided by IBM, whose PowerPC product is the RISC System/6000 Model 250. Okay, okay, it's no coincidence that I know the model of RS/6000 that the PowerPC is in, seeing as how it's the product line that I support as an IBM Systems Engineer!

With this review of the upcoming presentations I hope it gets you as excited about our 1994 meetings as it does me. I think we are getting the "hot" topics presented to you. If you have any opinions on what you think about past topics, or any you want to see in the future, please feel free to contact myself or anyone else on the board. We're always glad to hear from any members.

Looking forward to seeing you at our January meeting!

CORPORATE SPONSORS

The Manitoba UNIX User Group

gratefully acknowledges the generous support of the following

Corporate Sponsors

IBM

digital™

HEWLETT PACKARD

UM

XEROX *The Document Company*

Great-West Life Assurance Company

QNX: A Realtime Unix

By Rennie Allen

What is a real-time Unix? To answer this, it's best to explain the term real-time as it applies to computers. Real-time, simply stated, is the ability for a computer system to interact with an external system without compromising the external system's inherent time constraints. This implies that a computer system can be considered real-time (for the system in which it is intended to operate), if it exhibits a deterministic response to events within the system which are within the bounds of allowable worst-case response times for that system.

Determinism?

If you're new to real-time, you may not be familiar with the term "determinism" (or deterministic), as it is rarely used outside of the real-time industry. A deterministic system is one in which the worst-case time for a particular action can be determined. It is generally considered highly desirable, (although not technically necessary to be considered real-time), that a real-time system exhibit minimal variation between best-case and worst-case times for a particular action, and that it exhibit high capacity for I/O throughput (i.e. fast).

Back to our question; what is a real-time Unix? Well, a real-time Unix is a Unix-compatible operating system (we'll get to the specifics of what "compatible" means later), which exhibits the required deterministic qualities for a relatively large domain of applications. Why do I place the caveat "relatively large domain of applications" on my definition of real-time Unix? Well, my definition of real-time states that the "external system" provides the constraints that determine whether a system can be considered real-time; therefore, you can't assert that any computer system is real-time without stating what the application domain is! While this statement is technically correct, I am sure you know that there are computer systems out there which claim to be real-time without any mention of a particular application domain. What is generally meant when people refer to a real-time system, is that the system is designed to be fast, deterministic, and reliable (it is up to the software engineer designing the real-time system to determine whether it is fast enough, and reliable enough :-).

"A real-time operating system is generally designed to be fast, deterministic, and reliable."

While a complete discussion of real-time involves the external system, the computer hardware, the operating system, and the software applications I am going to discuss only the operating system, since it is typically the most important link in designing a real-time system. Because of my familiarity with QNX, I will discuss real-time operating systems using QNX as an example.

QNX History

The originators of QNX (Dan Dodge and Gord Bell), did not specifically set out to develop a real-time operating system. Back in 1980, they were hobbyists playing around with a little OS designed around a relatively new kind of architecture called a micro kernel. It just so happens that micro kernels (being small), lend themselves very nicely to real-time (small = less code = fast). When Dan and Gord decided to commercialize QNX, they chose to pursue the area of Intel based (at that time the IBM PC) real-time, since it was a totally untapped area. Both Dan and Gord had a great deal of exposure to Unix at the University of Waterloo, and consequently when it came time to create a user interface for their RTOS (real-time operating system), they decided to make the utilities and shell "Unix like" (there were no standards to follow in 1982), hence the birth of Qunix (before AT&T's lawyers suggested that they change the name ;-).

QNX developed a large following as a RTOS in the following 7 years, when, in 1989 — with standards becoming available for Unix, QSSL (QNX Software Systems Limited), embarked on a project to make QNX a real Unix operating system, rather than just "Unix-like."

Easy Porting

To this end they provided Posix 1 (1003.1) and Posix 2 (1003.2) compliance, as well as selected SYS V (System V) and BSD (Berkeley Software Distribution) libraries. This level of "Unixness" is sufficient to allow successful compilation, linking, and execution, of any Unix source which is ANSI conforming, and that obeys Posix 1, (but may utilize SYS V and BSD calls where Posix 1 has no say), typically with little alteration. This level of compatibility is high enough to allow simple porting of most freeware on the Internet. Internet code which exhibits a high degree of portability, and is "modern" (i.e. was developed on an ANSI standard compiler), usually ports with no alteration. What does this level of compatibility really mean? Well, I ported gopher2.0+ in 3 hours.

Gopher client is a non-trivial application utilizing curses and sockets. 90% of the time was spent configuring the headers and makefile correctly. I have also ported ncftp (15 minutes), elm (2 hours), gzip (0 min), and gfax (5 hours). Other QNX users have ported ghostscript, dbm, perl, groff, xv, tvm, xmpeg, and several other packages. Of course, the Posix 2 compliance makes operating at the shell identical to any other Posix 2 system, and very nearly the same as most other non-Posix Unixes (there isn't a great deal of difference between Posix 2 and traditional Unix). QNX is currently preparing a new version in order to fully support the realtime Posix standards (the 4's) which should be completed next year. Based on QSSL's commitment to standards, I would expect that QSSL will take QNX to spec1170 compliance as well, in order to earn the legal right to call QNX a realtime Unix, and to further enhance the portability of Unix code to and from QNX.

With both the "what is real-time" and "what is Unix" bases (hopefully :-) covered, we can proceed with a discussion of some of the unique features that make QNX an OS well suited to the development of real-time and/or embedded applications.

What makes QNX different?

QNX has taken a different approach than many other RTOS vendors. Most other RTOS's host their development environment on another platform (typically Unix or DOS), and provide a small OS that is designed to run comfortably in an embedded computer (a very common configuration for a real-time system). QNX, being a micro kernel based operating system (the micro kernel is about 9000 bytes), is fully modular, and therefore allows you to "plug-in" modules in order to bring the OS up to a level suitable for use as a development platform. For instance, QNX has two "process managers" or Proc's (pronounced P'rock) in QNX lingo — one a 16 bit (which runs only 16 bit applications), and the other a 32 bit (which runs 16 bit or 32 bit applications). Many real-time/embedded applications do not require a 32 bit operating system (with the additional overhead), and as such would be configured ☛

with the 16 bit Proc, while a development system (running X Windows, and utilizing gnu utilities etc.) must have 32 bit (try porting X without it!), and would load a 32 bit Proc. Since the 32 bit Proc will also run 16 bit applications, the developer can test the applications he is developing on his/her own workstation.

Networking

A very notable area where QNX provides functionality outside of the realm of Unix or other RTOS's is in its networking. While QNX has a full BSD Net-2 distribution of TCP/IP with NFS (both client and server), QNX also has its own proprietary network. QNX's network protocol is a "lightweight" protocol designed to facilitate the merging of multiple micro kernels (running on different processors across the network), into a single logical network-wide micro kernel (macro kernel?). This makes QNX a fully network distributed OS, which adds another layer of scalability and reliability (QNX supports multiple network connections with load balancing, and automatic fault recovery). When combined with TCP/IP, an entire network of QNX machines will consume only one IP address (since the QNX "network" is logically a single computer). Being a distributed OS affords developers another method to test his/her applications. Since the QNX network is one logical computer, the developer can (in a suitably configured network), command the OS to run his test application on an actual embedded processor (configured identically to the actual target processor), without any "downloading", or other productivity-robbing processes (the embedded processor is simply another "cpu space" in which to execute the application).

Gobs of GUI

When it comes to GUI support, QNX is unrivaled. QNX supports (in no particular order): QNX Windows (an object oriented GUI similar to NeXTstep), Photon (an embedded windowing environment — 256 KB in size), X-Windows (we all know what this is :-), and MS-Windows (we all know what this is too ;-). The plethora of windowing options, while somewhat confusing, provides a great deal of flexibility. QNX's application builder (a GUI code generator), can generate applications for QNX Windows, X Windows, or Photon. Development of MS Windows apps is supported through the standard SDK's, or MS-Windows hosted GUI builders. MS-Windows apps can communicate with QNX apps through file-descriptor based I/O (pipes). Photon (the embedded windowing environment) is a fully network distributed windowing system. By fully network distributed I am not talking client-server (like X-Windows or QNX-Windows), but absolutely network transparent. This can get very weird, (if you let it) since Photon is inherently capable of dragging a window off a display attached to one computer across the network to another computer. This would allow (for instance), one person to "give" an application to a user on another machine (while it's running) simply by dragging the window into the virtual space managed by that users' computer. While this behaviour is simply a "side-effect" of Photons' design (and the underlying QNX architecture), and it may not be terribly practical, it does serve to illustrate the power of the underlying distributed operating system (remember Photon is only 256KB!).

Development tools

Anyone developing real-time applications will be very concerned about the quality of development tools available. QNX provides both the Watcom C and C++ compilers. The Watcom 32 bit C/C++ compiler is a highly optimizing (Pentium optimizations included), compiler, which utilizes register-based argument passing (rather than stack-based). The Watcom compiler is highly rated in the DOS-Windows-OS/2 world for the speed of the executables, and QNX has the distinction of being the only Unix OS to currently host it. The Watcom compiler has passed the Plum Hall validation suite (for ANSI compatibility) with 100% compliance, and the C++ compiler supports true exception handling, and templates. In addition I have heard recently that a port of gcc to QNX has been completed, so if you're game for writing your own libraries, you can develop commercial software with a free compiler, or alternatively, if your projects are for internal use you can "simply" write the QNX specific portions of the libraries (this is a non-trivial but doable project) and use the glib, for the non QNX-specific standard library functions.

Debugging? Ha!

As for debugging tools... who needs 'em?... only people who have bugs in their code... right ? Well for those of you like me, it is fortunate that the Watcom compiler comes with a very good, full screen, source-level debugger, which fully supports C++ name demangling, and definitely requires a mouse to be usable (you should hear the whining from people who try to use it without a mouse ;-). Finally, QNX ships RCS with the operating system, as well as all the usual tools you expect in a Unix environment (and must have to be Posix 2 compliant). In addition to the compilers, debugger, and general utilities for development, QNX, as mentioned earlier, has a product called the application builder to aid in producing GUI applications. There is also a third party macro assembler available (called MAX), for those masochistic (or purist — depending on your point of view), souls, who like to code in assembler (sick, sick, puppies). I should point out that a human cannot develop better assembler code than the compiler for either the 80486 or Pentium parts, since these parts require instructions to be out-of-sequence in order to be fully optimal (I suppose it is possible for a human to reorder instructions and be as efficient as the compiler, but doing this would be the act of a desperate and depraved individual :-).

" There is no need for 'tool starvation' in a QNX-based environment. "

I think it should be fairly obvious that there is no need for "tool starvation" in a QNX based environment, as the tools are very complete and powerful.

Potential applications

Well, you say, all of this is well and good but why do I need a real-time Unix as opposed to a non-real-time Unix? The answer might be that you don't, it all depends on your application domain. One of the currently hot application areas that most definitely requires real-time capability is multimedia. Picture if you will, an MPEG movie. In order to obtain full-motion video you need 30 frames/second, not an average of 30 frames/second, but 30 frames/second — no exceptions. If you don't get the required frame every 1/30th of a second there will be a noticeable "jitter" in the sequence. In this case the application domain requires that X bytes of picture data be delivered and displayed in 1/30th of a second. It must be "determinable" (of the OS) that the worst-case time for the load and display of the frame image is less than 1/30th of a second (excess time after the load and display are done, can be used by waiting on a timer event). In this case it is necessary that the system provide deterministic response for both the I/O necessary to load the picture, the computation necessary to prepare the picture, and the I/O necessary to move the picture to the display device, and have it displayed. To obtain high-quality multi-media (you've probably witnessed jittery non-realtime MPEG movies) a real-time OS is required. In a non-deterministic OS it might be possible to have 30 frames/sec displayed, as long as the loading on the system is ☛

low; if suddenly 3 or 4 apps all hit the disk at the same time, the frame for "this" 1/30th of a second interval could be compromised, and jitter introduced.

In a real-time OS, the MPEG player would be assigned a higher priority than the non-real-time apps accessing the disk, and when the MPEG player wanted to run (i.e. 1/30th of a second has elapsed since the last frame was displayed), the MPEG player would pre-empt, all the other processes in the system, thereby gaining complete access to the computational and I/O capacity of the system. It can therefore be calculated that by supplying X computational and I/O bandwidth, we can guarantee that a frame will be loaded every 1/30th of a second.

The classical application domain for real-time systems is control systems (and robotics). This is the area most people think of when they think real-time systems. It is true that missing a deadline in a control system can have far greater impact on people's lives, than if an MPEG movie player misses its frame display deadline, however, this does not mean that the MPEG player is any less of a real-time system, simply that it's not a mission critical, real-time system.

The other big application area of real-time systems is in transaction processing. Here, the requirement for deterministic response is driven mainly by customer satisfaction requirements. This is known as soft real-time. Soft real-time (previous discussions centred around hard-real-time), is characterized by the absence of a fixed-deadline. For instance, in transaction processing (let's say a credit-card validation system), Visa (for example) might say to the system developer "95% of the time we want a transaction to take less than 10 seconds". The remaining 5% of the time Visa is willing to let its customers wait; they don't want to spend the money to buy the computational horsepower to deliver 100% deterministic response, but they do want it to be "mostly deterministic". In this case it is still necessary to use a real-time OS, since you still must be able to determine that you can reach the 95% level of 10 second (or less) response.

Finding out more

Perhaps you've been intrigued by this modest article, and wish to find out more about real-time Unixes? Well have I got a deal for you! As you may recall from spring of this year, I had made some noises about starting a real-time SIG (an RSIG... I love acronyms :-), well, I am finally getting around to doing something about it. At the January meeting, I will give a short presentation, on the proposed RSIG, and leave a sign-up sheet on the desk (the one by the doors to the auditorium). I would ask that those of you who are interested, to put your name down on the sheet, and I'll announce at the following meeting, whether there is sufficient interest (we need around 15 interested parties). Remember this is a RSIG (not a QSIG :-), so whatever RTOS you use (or are considering using), whether it's LynxOS/HP-RT, AMX, RMX, Venix, homegrown, or ? come on down to the January meeting and sign-up. QSSL has agreed to sponsor the creation of the RSIG by donating software so that we can set up a real-time demo/test platform, and we will be seeking sponsorship from other RTOS vendors as well (depending on interest levels of RSIG members).

C++ Q&A

By Marshall P. Cline

The C++ Q&A column was started last month, and this month continues answering questions.

Question 3: Who uses C++?

Lots and lots of companies and government sites. Lots. Statistically, 20 to 30 people will consider themselves to be new C++ programmers before you finish reading the responses to these FAQs.

Question 4: Are there any C++ standardization efforts underway?

Yes; ANSI (American) and ISO (International) groups are working closely with each other. 'X3J16' is the name of the ANSI-C++ committee. 'WG21' is the name of ISO's C++ standards group.

The committees are using the 'ARM' as a base document: 'Annotated C++ Reference Manual', Ellis and Stroustrup, Addison/Wesley. ISBN 0-201-51459-1

The major players in the ANSI/ISO C++ standards process includes just about everyone:

AT&T, IBM, DEC, HP, Sun, MS, Borland, Zortech, Apple, OSF, <add your favorite here>, ... and a lot of users and smaller companies. About 70 people attend each ANSI C++ meeting. People come from USA, UK, Japan, Germany, Sweden, Denmark, France, ... (all have 'local' committees sending official representatives and conducting 'local' meetings).

Optimistically the standard might be finished by 1995-6 time frame (this is fast for a proper standards process).

Question 5: Where can I ftp a copy of the latest ANSI-C++ draft standard?

You can get a paper copy by sending a request to:

Standards Secretariat
CBEMA/X3
1250 I Street NW Suite 200
Washington, DC
20005

Ask for the latest version of 'Working Paper for Draft Proposed American National Standard for Information Systems — Programming Language C++'. The last known phone number: 202-626-5738. The last known price is $25.

Question 6: Is C++ backward compatible with ANSI-C?

Almost. C++ is as close as possible to compatible with ANSI-C but no closer. In practice, the major difference is that C++ requires prototypes, and that 'f()' declares a function that takes no parameters, while ANSI-C rules state that 'f()' declares a function that takes any number of parameters of any type. There are some very subtle differences as well, like the sizeof a char literal being equal to the sizeof a char (in ANSI-C, sizeof('x') is the sizeof an int). Structure 'tags' are in the same namespace as other names in C++, but C++ has some warts to take care of backward compatibility here.

Question 7: How long does it take to learn C++?

I and others teach standard industry 'short courses' (for those not familiar with these, you pack a university semester

course into one 40hr work-week), and have found them successful. However mastery takes experience, and there's no substitute for time. Laboratory time is essential for any OOP course, since it allows concepts to 'gel'.

Generally people start out wondering why the company has devoted a full 5 days to something as trivial as another programming language. Then about half way through, they realize they're not being taught just a new syntax, but an entirely different way of thinking and programming and designing and . . . . Then they begin to feel dumb, since they can't quite grasp what is being said. Then they get mad and wonder why the course isn't taught in two or three weeks instead. Finally about Wednesday afternoon the lights go 'clink', and their faces brighten, and they 'get it'. By Friday, they've had numerous laboratory 'experiments' and they've seen both sides of reusable components (both how to code *from* reuse, and how to code *for* reuse). It's different in every time I teach, but the 'reuse' aspect is rewarding, since it has a large potential to improve software production's overall economics.

It takes 9 months to 'master' C++/OOP. Less if there is already a body of experts and code that programmers have regular access to, more if there isn't a 'good' general purpose C++ class library available.

Dr. Marshall P. Cline is the founder and President of Paradigm Shift, Inc., a firm that specializes in on-site training for C++, OOD, OOA, consulting, and reusable/ extensible C++ class libraries. For more information, send e-mail to "info@parashift.com".

THE FORTUNE FILE

Original Author Unknown

Submitted by Andrew Trauzzi

Dictionary of Evaluation Comments

Some of you might like to know what the supervisor is really saying in all those glowing employee work performance evaluations s/he keeps cranking out.

AVERAGE:
Not too bright.

EXCEPTIONALLY WELL QUALIFIED:
Has committed no major blunders to date.

ACTIVE SOCIALLY:
Drinks heavily.

ZEALOUS ATTITUDE:
Opinionated.

CHARACTER ABOVE REPROACH:
Still one step ahead of the law.

UNLIMITED POTENTIAL:
Will stick with us until retirement.

QUICK THINKING:
Offers plausible excuses for errors.

TAKES PRIDE IN WORK:
Conceited.

TAKES ADVANTAGE OF EVERY OPPORTUNITY TO PROGRESS:
Buys drinks for superiors.

INDIFFERENT TO INSTRUCTION:
Knows more than superiors.

STERN DISCIPLINARIAN:
A real jerk.

TACTFUL IN DEALING WITH SUPERIORS:
Knows when to keep mouth shut.

APPROACHES DIFFICULT PROBLEMS WITH LOGIC:
Finds someone else to do the job.

A KEEN ANALYST:
Thoroughly confused.

NOT A DESK PERSON:
Did not go to college.

EXPRESSES SELF WELL:
Can string two sentences together.

SPENDS EXTRA HOURS ON THE JOB:
Miserable home life.

CONSCIENTIOUS AND CAREFUL:
Scared.

METICULOUS IN ATTENTION TO DETAIL:
A nitpicker.

DEMONSTRATES QUALITIES OF LEADERSHIP:
Has a loud voice.

JUDGEMENT IS USUALLY SOUND:
Lucky.

MAINTAINS PROFESSIONAL ATTITUDE:
A snob.

KEEN SENSE OF HUMOR:
Knows lots of dirty jokes.

STRONG ADHERENCE TO PRINCIPLES:
Stubborn.

SLIGHTLY BELOW AVERAGE:
Stupid.

OF GREAT VALUE TO THE ORGANIZATION:
Turns in work on time.

IS UNUSUALLY LOYAL:
Wanted by no-one else.

ALERT TO COMPANY DEVELOPMENTS:
An office gossip.

HARD WORKER:
Usually does it the hard way.

ENJOYS JOB:
Needs more to do.

HAPPY:
Paid too much.

WELL ORGANIZED:
Does too much busywork.

COMPETENT:
Is still able to get work done if supervisor helps.

CONSULTS WITH SUPERVISOR OFTEN:
Pain in the ass.

WILL GO FAR:
Relative of management.

SHOULD GO FAR:
Please.

USES TIME EFFECTIVELY:
Clock watcher.

VERY CREATIVE:
Finds 22 reasons to do anything except original work.

USES RESOURCES WELL:
Delegates everything.

DESERVES PROMOTION:
Create new title to make h/h feel appreciated.

UNIX Q&A

Originally Compiled by Ted Timar

Submitted by Andrew Trauzzi

UNIX Q&A is a monthly column that will hopefully answer some commonly asked UNIX questions. If you have any specific UNIX questions, please submit them to Monsieur Ex via e-mail <m-ex@muug.mb.ca>.

Q1: How do I remove a file whose name begins with a "-"?

Figure out some way to name the file so that it doesn't begin with a dash. The simplest answer is to use

```
rm ./-filename
```

(assuming "-filename" is in the current directory, of course.) This method of avoiding the interpretation of the "-" works with other commands too.

Many commands, particularly those that have been written to use the "`getopt(3)`" argument parsing routine, accept a "`--`" argument which means "this is the last option, anything after this is not an option", so your version of rm might handle "`rm -- -filename`". Some versions of rm that don't use getopt() treat a single "-" in the same way, so you can also try "`rm - -filename`".

Q2: How do I remove a file with funny characters in the filename?

If the 'funny character' is a '/', skip to the last part of this answer. If the funny character is something else, such as a ' ' (space) or control character or character with the 8th bit set, keep reading.

The classic answers are

```
rm -i some*pattern*that*matches*only*the*file*you*want
```

which asks you whether you want to remove each file matching the indicated pattern; depending on your shell, this may not work if the filename has a character with the 8th bit set (the shell may strip that off); and

```
rm -ri .
```

which asks you whether to remove each file in the directory. Answer "y" to the problem file and "n" to everything else. Unfortunately this doesn't work with many versions of rm. Also unfortunately, this will walk through every subdirectory of ".", so you might want to "`chmod a-x`" those directories temporarily to make them unsearchable.

Always take a deep breath and think about what you're doing and double check what you typed when you use rm's "-r" flag or a wildcard on the command line; and

```
find . -type f ... -ok rm '{}' \;
```

where "..." is a group of predicates that uniquely identify the file. One possibility is to figure out the inode number of the problem file (use "`ls -i .`") and then use

```
find . -inum 12345 -ok rm '{}' \;
```

or

```
find . -inum 12345 -ok mv '{}' new-file-name \;
```

"-ok" is a safety check — it will prompt you for confirmation of the command it's about to execute. You can use "-exec" instead to avoid the prompting, if you want to live dangerously, or if you suspect that the filename may contain a funny character sequence that will mess up your screen when printed.

What if the filename has a '/' in it?

These files really are special cases, and can only be created by buggy kernel code (typically by implementations of NFS that don't filter out illegal characters in file names from remote machines.) The first thing to do is to try to understand exactly why this problem is so strange.

Recall that Unix directories are simply pairs of filenames and inode numbers. A directory essentially contains information like this:

```
filename        inode
file1           12345
file2.c         12349
file3           12347
```

Theoretically, '/' and '\0' are the only two characters that cannot appear in a filename - '/' because it's used to separate directories and files, and '\0' because it terminates a filename.

Unfortunately some implementations of NFS will blithely create filenames with embedded slashes in response to requests from remote machines. For instance, this could happen when someone on a Mac or other non-Unix machine decides to create a remote NFS file on your Unix machine with the date in the filename. Your Unix directory then has this in it:

```
filename        inode
91/02/07        12357
```

No amount of messing around with '`find`' or '`rm`' as described above will delete this file, since those utilities and all other Unix programs, are forced to interpret the '/' in the normal way.

Any ordinary program will eventually try to do unlink ("`91/02/07`"), which as far as the kernel is concerned means "unlink the file 07 in the subdirectory 02 of directory 91", but that's not what we have - we have a *FILE* named "`91/02/07`" in the current directory. This is a subtle but crucial distinction.

What can you do in this case? The first thing to try is to return to the Mac that created his crummy entry, and see if you can convince it and your local NFS daemon to rename the file to something without slashes.

If that doesn't work or isn't possible, you'll need help from your system manager, who will have to try the one of the following. Use "`ls -i`" to find the inode number of this bogus file, then unmount the file system and use "`clri`" to clear the inode, and "`fsck`" the file system with your fingers crossed. This destroys the information in the file. If you want to keep it, you can try:

- create a new directory in the same parent directory as the one containing the bad file name;
- move everything you can (i.e. everything but the file with the bad name) from the old directory to the new one;
- do "`ls -id`" on the directory containing the file with the bad name to get its inumber;
- unmount the file system;
- "`clri`" the directory containing the file with the bad name;
- "`fsck`" the file system.

Then, to find the file,

- remount the file system;
- rename the directory you created to have the name of the old directory (since the old directory should have been blown away by "`fsck`")
- move the file out of "`lost+found`" into the directory with a better name.

Alternatively, you can patch the directory the hard way by crawling around in the raw file system. Use "`fsdb`", if you have it.

Next month we'll look at some prompt hacks, as well as some shell script advice. If you have any comments, or other methods of solving the problems presented here, mail <editor@muug.mb.ca>.

OS/2 2.1 - UNIX Utilities Invade!

By Gord Tulloch

IBM's OS/2 2.0 and lately 2.1 have been very successful in offering INTEL based computer users many of the same things that UNIX users have been enjoying for some time — truly useful utilities for running UNIX style systems. This article details my experiences with these tools, in case anyone out there is unwilling to forego the DOS/Windows/OS/2 compatibility of OS/2 for a true UNIX system.

OS/2's UNIX-like features

OS/2 is one of the first MS-DOS-derived operating systems to offer the following:

True multitasking and process control

Formerly, so-called multitasking on MSDOS machines took two forms — Desqview, which allowed pre-emptive multitasking of DOS applications but little dependability and protection from crashes, and Windows, which worked via non-preemptive cooperative multitasking where each application had to relinquish control of the system in turn before the next task could execute. Programs where timing was critical (such as downloading) suffered immensely when tasks monopolized CPU time.

Long file names

The new High Performance File system (HPFS) unveiled in OS/2 1.3 meant that UNIX applications could be ported far more easily, avoiding DOS's primitive CP/M-80 derived 8.3 naming convention for files and directories.

32 bit addressing

While various kludges under DOS have allowed programmers access to the new 32 bit addressing used on the 386 and greater INTEL CPUs, OS/2 2.x has made this function more easily obtainable, leading to ports of 32 bit tools for UNIX such as GNU's language products.

Software Development Tools

GNU C and C++

GNU's C and C++ tools have been ported to OS/2 (as well as DOS) and generate native 32 bit code, using a dynamic link library (DLL) to provide kernel services for the programs. Coupled with other GNU utilities, including make, info, man and others, these tools permit OS/2 users to gain the advantages of UNIX development under OS/2. Since the GCC compiler is becoming a preferred compiler across UNIX platforms as well, deployment of cross platform packages has never been easier.

EMX

Extending the GNU utilities, emx allows OS/2 programmers an inexpensive tool for developing true native 32 bit OS/2 programs, including Presentation Manager programs for OS/2's Graphical User Interface. Although the new System Object Model is currently not supported, shutting programmers out of the Workplace Shell object oriented user interface, undoubtedly future versions will add this functionality.

Electronic Mail and News

UUPC/Extended

While this program has been available for DOS and OS/2 for some time, OS/2's new extended file naming capabilities allowed more complete coverage of functions required for true UUCP functions, such as downloading files with long names.

ELM

With long filenames, UNIX software such as ELM appeared quickly, running ONLY under HPFS.

TRN

Since UUPC allows uucp file transfers, network news has become available to all. TRN simply replaces the UUPC RNEWS function to permit receipt and processing of USENET newsgroups.

Miscellaneous Utilities

Networking

Software like IBM's TCP/IP for OS/2, FTP's PCTCP for OS/2 and several good XWindow servers for OS/2 have permitted the deployment of OS/2 workstations in a UNIX environment both as character and GUI workstations, easing the transition of stand alone INTEL based PCs to a heterogeneous network of mixed architecture workstations. UNIX based software vendors such as PROGRESS have leveraged the widespread use of Windows and DOS software to create highly functional GUI applications for small systems that are clients for UNIX servers.

emacs

Since GNU C has been ported, the GNU suite of tools was soon to follow, led by a complete version of GNU emacs, which requires HPFS to run.

cron

Rather than simply execute at a certain time functions that hid in memory and grabbed control of the machine at predetermined times, a number of cron work-alikes have arisen which work almost exactly like the cron utilities under UNIX.

Shell and disk utilities

For UNIX users like myself, the availability of UNIX programs like grep, ls, rm, troff and others allows easy transition from UNIX during the daytime to OS/2 at home, as well as in the office.

Where do you get these tools?

The major site for obtaining these tools is Hobbes, which can be reached via FTP at ftp-os2.nmsu.edu. Hobbes maintains a huge collection of OS/2 1.x and 2.x software as well as ports of numerous standard UNIX utilities, including the bulk of the GNU utilities.

Conclusion

Although UNIX afficienados fervently hope that UNIX will prevail in the desktop market, non-UNIX operating systems are rapidly moving towards UNIX functionality. As more UNIX utilities are ported, many users will easily grow into a situation where UNIX is the most appropriate choice. In my own situation, where desktop workstations running DOS and OS/2 are integral parts of the movement towards client-server computing, availability of UNIX tools like TCP/IP-based telnet and ftp, emacs, uucp and elm on my OS/2 system at the office make moving between platforms effortless.

At home, of course, I have begun using LINUX. A long time OS/2 fan, I discovered that MOST of the software I use under OS/2 has been ported from UNIX, so LINUX fulfills my needs quite nicely. However, as LINUX gains the ability to run DOS and Windows software, my DOS partition will simply get smaller and smaller...

SIG Sideline

By Bary Finch, SIG Coordinator

It's time to get back at it! We all had a good Christmas vacation (I hope), including a break from our SIG meetings. Now we're ready to start up again.

We will be meeting at our usual place, ISM at 400 Ellice. It will also be the usual time of 7:30 p.m., on January 18. As usual this is the third Tuesday of the month. You can just go to the front doors of ISM on Ellice, which are locked, and you'll be let in. Please arrive after 7:15 p.m., as that's when Wolfgang will be at the front doors to let you in. If you arrive too early you may have a chilly wait!

We are arranging for a presentation this month, but we still haven't finalized the topic. The topic chosen will be from the list of topics that I showed in last months SIG Sideline. If there are any other topics that people would like to see, let me know. Or if you have a presentation that you'd like to do. We want to get as much participation by as many people as we can.

We also may not have the presence of Greg's Linux luggable, as it had disk problems severe enough to disable it. We'll see what happens by Jan.18.

We have had good success with our single SIG, but if anyone is interested in starting another SIG please contact me. You would need to have someone that will volunteer as the SIG Coordinator. This role entails coordinating the meetings as well as preparing a monthly contribution to this column. This will keep us all up to date on what's happening in the SIG.

THE FORTUNE FILE

Original Author Unknown

Submitted by Andrew Trauzzi

I was musing on similarities between Santa Claus and system administrators. Consider:

- Santa is bearded, corpulent, and dresses funny.
- When you ask Santa for something, the odds of receiving what you wanted are infinitesimal.
- Santa seldom answers your mail.
- When you ask Santa where he gets all the stuff he's got, he says, "Elves make it for me."
- Santa doesn't care about your deadlines.
- Your parents ascribed supernatural powers to Santa, but did all the work themselves.
- Nobody knows who Santa has to answer to for his actions.
- Santa laughs entirely too much.
- Santa thinks nothing of breaking into your $HOME.
- Only a lunatic says bad things about Santa in his presence.

Agenda

for
Tuesday, January 11, 1993, 7:30 PM
Samuel N. Cohen Auditorium
St-Boniface Hospital Research Centre
Main Floor, 351 Taché

1. President's Welcome 7:30
2. Round Table 7:35
3. Business Meeting 8:00
   a) Old Business
   b) New Business
4. Coffee Break 8:30
5. Presented Topic 8:45
   Hewlett-Packard discusses COSE —
   The Common Open Software Environment
   Stay tuned for details — watch muug.general for updates, as they become available.
6. Adjourn 9:30

Note: Please try to arrive at the meeting between 7:15 and 7:30, to avoid disrupting the meeting in progress.

Coming Up

Meeting:
Next month's meeting is scheduled for Tuesday, February 8, at 7:30 PM. Meeting location will be the St-Boniface Research Centre, as usual. The February meeting topic is the Internet. Stay tuned for details.

Got any ideas for meeting topics? Any particular speaker, company, or product you'd like to see at one of our meetings? Just let our new meeting coordinator, Roland Schneider, know. You can e-mail him at <rsch@muug.mb.ca>.

Newsletter:
If you are interested in a particular topic, let me know. I'm sure I could coerce you into writing an article! I could use a few articles — especially shorter ones — half a page to one page (400 to 1000 words) would be fine.

Monsieur Ex has also let me know that his mail-box has room for more of your wonderful queries again – please submit your questions to the old guy via e-mail to <m-ex@muug.mb.ca>. He may be old, but he's not ready for retirement yet!