Volume 7, Number 7 May 1995 $2.50

# MUUG Lines

Manitoba UNIX® User Group

Newsletter of the Manitoba UNIX© User Group

## Why Have a Firewall?

***By David Bonn, co-founder, Mazama Software Labs***

Maintaining reasonable system security on each and every machine on a local area network may well be impossible. Even in the best of cases, this is a dauntingly complex administrative task. Installation of new software will quietly introduce a security hole — and this problem isn't restricted to obvious security-related programs either. Many large applications use networking services in surprising ways, and it is in general impossible to predict how even the most innocuous piece of software might affect a local area network — this is especially true of software for personal computers. From an administrative standpoint, it is also very unlikely that the person responsible for overall security will have significant control over what software is being ran on all workstations.

A firewall deals with this problem by assuming that all machines inside the network can trust one another. What a firewall does is provide a single entry point that can be closely monitored for security problems. A firewall can run on inexpensive hardware that can be administratively controlled by those responsible for security.

The ideal firewall is quite simple. It only has four essential requirements:

- It should be very easy to set up and require little maintenance.
- It should provide accurate and secure logging of interesting firewall events.
- It should not interfere or even be noticeable to legitimate users of the local network, even when they access resources outside the local network.
- It should keep unauthorized people outside the local network from entering the local network.

Quite a few existing solutions to the firewall problem exist. Like all solutions, there are quite a few tradeoffs. In general, the more convenient solutions are much more expensive.

Writing packet filtering rules is very, very hard. Having good support tools that make it easier to write correct packet filtering rules is very important.

Most commercial routers have some kind of packet filtering capability built in. If you have access to the router you are using, this might be a good solution for you.

Router-based packet filtering hasn't been very satisfactory in practice. Some of the reasons for this include:

- Router-based firewalls are hard to configure.
- It is rare to be able to filter on all properties of IP packets.
- Many routers attempt to optimize filter rules.
- These optimizations often break over-specified, but working, filter rules.
- Many people rent their routers from their internet service provider and don't have access to the router for configuration purposes.
- If you upgrade routers, or purchase a different brand of router, you'll have to do your configuration all over again.

There are many freeware, shareware, and commercial solutions to firewalls — watch this space for reviews. ➻

## This Month's Meeting

**Meeting Location:**
Our next meeting is scheduled for Tuesday, May 9, 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

Printing Courtesy of Xerox Canada Ltd.

# Is Programming A Lost Art?

***By Andrew Trauzzi***

I remember the good old days back in '89 when we used to code. Yup, don't think I'll ever see times like that again. Now all you young whipper snappers have your integrated packages, your code generators, and whatnot. Phooey! Why I remember one project way back when...

Sounds convincing doesn't it? How many years off is this scenario? Twenty or thirty? Try right now!

Many companies are sick and tired of shelling out hundreds of thousands of dollars on computer systems just to keep up with their competitors. Not only do companies need new computer hardware every five years, they need custom software that takes advantage of the new hardware's capabilities. The cost of custom software usually exceeds the cost of new hardware by a long shot. How did the computer industry manage to work itself into this state?

Let's go back to the 60s. Say a company bought a mainframe in 1965 — what could they expect to have to purchase in the near future? Well, they would have someone write custom software for the original machine, and they would have to purchase an operating system. For the next five to eight years, the company purchases upgrades to their mainframe and operating system. That's it. Sure, the upgrades were expensive, but it only took a few employees a day to install them. Contrast that with buying a hundred new PCs and associated software.

The price is roughly about the same, but you have to have a team of eight people installing the computers, and two full-time staff constantly baby-sitting the machines. The staff becomes confused because of the new operating system, and they all have to go on courses. On top of all that, the custom software you had written four years ago must be completely rewritten for the new platform! The company can live with retraining staff, because it's relatively inexpensive and boosts productivity. The new custom software is deemed too expensive, so the company buys a pre-written package and fires half of its programmers.

These days, in-house programmers don't really program — they script, they integrate, and they customize. It's not the number of languages you know, it's the number of packages you know. There's something else — the clerk next to you is rather handy with macros, so they move him to your position. He may not have any formal training, but he costs $15,000 less per year, and whines a lot less about not being able to program anymore. ➜

## The 1993-1994 Executive

| | | |
|---|---|---|
| President: | Bary Finch | (W) 934-1690 |
| Vice-President: | Rob Wright | (W) 488-5175 |
| Treasurer/Secretary: | Doug Shewfelt | (W) 986-3748 |
| Publicity Director | Rory Macleod | (W) 488-5168 |
| Programs: | Doug McLean | (W) 985-1643 |
| Mailing List: | Roland Schneider | 1-204-785-9179 |
| Membership Sec.: | Roland Schneider | 1-204-785-9179 |
| Newsletter editor: | Andrew Trauzzi | (W) 986-3898 |
| Past President | Susan Zuk | (W) 989-3530 |
| Information: | Bary Finch | (W) 934-1690 |
| | | (FAX) 934-1308 |
| (or) | Andrew Trauzzi | (W) 986-3898 |
| | | (FAX) 586-8567 |

## Copyright Policy and Disclaimer

## 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

## It Seems to be Spring

***By Bary Finch***

Well it seems to finally be warming up around here, and the highest technology item on people's minds is a rake. However, we still have one more meeting to entice you indoors with.

Unfortunately I missed our last meeting, due to being out of town. I gather the presentation by The North West Company was well received. The presenter, David Hodge, Director, Technology Services, gave a good overview of not only what The North West Company is, but also how it is using UNIX to enable their business. Maybe I should say "our" business, as I do in fact work for The North West Company, and have a great familiarity with how we are using UNIX. I'd better as I'm the UNIX analyst!

With being away, I of course had to have an alternate host for the meeting. Thanks once again to Rob Wright, Vice-President, for hosting in my absence.

In the ongoing saga of the MONA refund cheques, the first group of you will have received your refund cheques by now. If any of you have questions about this matter, please call me or email me.

As for the rest of you whom are awaiting a refund cheque, you won't have to wait much longer! The new cheque order has now arrived, and I will be processing the next and final batch of refund cheques over the next week. There are about 50 more of you to receive refund cheques, so it will take a bit to get all the letters printed and the cheques signed. But we'll get them out as soon as possible.

The other members that are still awaiting refunds are the UniForum members that never received memberships, due to UniForum Canada closing up. Now that the new cheques have arrived, I will be sending out refunds to each of you, along with a membership form for the US based UniForum organization. This will allow you to apply the refund (in Canadian dollars) towards the US UniForum membership (in US dollars) if you so desire.

I am still awaiting the forms from UniForum that show what the affiliate requirements are for MUUG. Once MUUG's board reviews these, we will decide if we want to become an affiliate of UniForum or not. Stay tuned.

I must say I'm a little disappointed in the lack of people that want to help out with the SIG. There were so many people that enjoy attending the meetings, and no-one is stepping up to the responsibility of hosting the SIG meetings. It isn't a lot of effort for someone that's already attending the meetings. Make that commitment, and help us out by becoming the SIG Coordinator.

On a happier note, we all have the annual MUUG BBQ to look forward to as our June meeting. We are still setting the location of this event, but I'm sure it will turn out as well as it always has. More details will be coming in the next newsletter.

That's all the rambling (oops! sorry Andrew) I've got for this month. See you at the next meeting!

CORPORATE SPONSORS

## The Manitoba UNIX User Group

**gratefully acknowledges the generous support of the following**

## Corporate Sponsors

IBM

digital™

HEWLETT PACKARD

Great-West Life Assurance Company

THE DOCUMENT COMPANY XEROX

TANDEM COMPUTERS

ONLINE BUSINESS SYSTEMS

sun microsystems

# C++ Q&A

*By Marshall P. Cline*

*This month's C++ Q&A continues last month's comparison of C++ and Smalltalk. The complete C++ FAQ is now available in a book format — Addison-Wesley Publishers 0-201-58958-3 $32.25.*

**Question 86: What is 'static typing', and how is it similar/ dissimilar to Smalltalk?**

Static (most say 'strong') typing says the compiler checks the type-safety of every operation *statically* (at compile-time), rather than to generate code which will check things at run-time. For example, the signature matching of fn arguments is checked, and an improper match is flagged as an error by the *compiler*, not at run-time.

In OO code, the most common 'typing mismatch' is invoking a member function against an object which isn't prepared to handle the operation. Ex: if class 'X' has member fn `f()` but not `g()`, and 'x' is an instance of class `X`, then `x.f()` is legal and `x.g()` is illegal. C++ (statically/ strongly typed) catches the error at compile time, and Smalltalk (dynamically/weakly typed) catches 'type' errors at run-time. (Technically speaking, C++ is like Pascal [*pseudo* statically typed], since ptr casts and unions can be used to violate the typing system; you probably shouldn't use these constructs very much).

**Question 87: Which is a better fit for C++: 'static typing' or 'dynamic typing'?**

The arguments over the relative goodness of static vs dynamic typing will continue forever. However one thing is clear: you should use a tool like it was intended and designed to be used. If you want to use C++ most effectively, use it as a statically typed language. C++ is flexible enough that you can (via ptr casts, unions, and #defines) make it 'look' like Smalltalk.

There are places where ptr casts and unions are necessary and even wholesome, but they should be used carefully and sparingly. A ptr cast tells the compiler to believe you. It effectively suspends the normal type checking facilities. An incorrect ptr cast might corrupt your heap, scribble into memory owned by other objects, call nonexistent methods, and cause general failures. It's not a pretty sight. If you avoid these and related constructs, you can make your C++ code both safer and faster — anything that can be checked at compile time is something that doesn't have to be done at run-time, one 'pro' of strong typing.

Even if you're in love with weak typing, please consider using C++ as a strongly typed OOPL, or else please consider using another language that better supports your desire to defer typing decisions to run-time. Since C++ performs 100% type checking decisions at compile time, there is *no* built-in mechanism to do *any* type checking at run-time; if you use C++ as a weakly typed OOPL, you put your life in your own hands.

**Question 88: How can you tell if you have a dynamically typed C++ class library?**

One hint that a C++ class library is weakly typed is when everything is derived from a single root class, usually 'Object'. Even more telling is the implementation of the container classes (List, Stack, Set, etc): if these containers are non-templates, and if their elements are inserted/extracted as ptrs to 'Object', the container will promote weak typing. You can put an Apple into such a container, but when you get it out, the compiler only knows that it is derived from Object, so you have to do a pointer cast (a 'down cast') to cast it 'down' to an Apple (you also might hope a lot that you got it right, cause your blood is on your own head).

You can make the down cast 'safe' by putting a virtual fn into Object such as 'are_you_an_Apple()' or perhaps 'give_me_the_name_of_your_class()', but this dynamic testing is just that: dynamic. This coding style is the essence of weak typing in C++. You call a function that says 'convert this Object into an Apple or kill yourself if its not an Apple', and you've got weak typing: you don't know if the call will succeed until run-time.

When used with templates, the C++ compiler can statically validate 99% of an application's typing information (the figure '99%' is apocryphal; some claim they always get 100%, others find the need to do persistence which cannot be statically type checked). The point is: C++ gets genericity from templates, not from inheritance.

**Q89: How do you use inheritance in C++, and is that different from Smalltalk?**

There are two reasons one might want to use inheritance: to share code, or to express your interface compliance. ie: given a class 'B' ('B' stands for 'base class', which is called 'superclass' in Smalltalkese), a class 'D' which is derived from B is expressed this way:

```
    class B { /*...*/ };
    class D : public B { /*...*/ };
```

This says two distinct things: (1) the bits(data structure) + code(algorithms) are inherited from B, and (2) 'D's public interface is 'conformal' to `B's (anything you can do to a B, you can also do to a D, plus perhaps some other things that only D's can do; ie: a D is-a-kind-of-a B).

In C++, one can use inheritance to mean:

• #2(is-a) alone (you intend to override most/all inherited code)

• both #2(is-a) and #1(code-sharing)

but one should never **never** use the above form of inheritance to mean

• #1(code-sharing) alone (ex: D really *isn't* a B, but...)

This is a major difference with Smalltalk, where there is only one form of inheritance (C++ provides 'private' inheritance to mean 'share the code but don't conform to the interface').

*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".*

# Ethernet Q&A

***Originally Compiled by Marc A. Runkel***

*Submitted by Andrew Trauzzi*

*This is the first in a short series of articles briefly explaining some Ethernet concepts. ATM may be hot, but it's more likely that you will be installing or using an Ethernet network.*

**Question 1: What is a baseband network?**
A baseband network is one that provides a single channel for communciations accross the physical medium (e.g., cable), so only one device can transmit at a time. Devices on a baseband network, such as Ethernet, are permitted to use all the available bandwidth for transmission, and the signals they transmit do not need to be multiplexed onto a carrier frequency. An analogy is a single phone line such as you usually have to your house: Only one person can talk at a time — if more than one person wants to talk everyone has to take turns.

**Question 2: Ok, so what is a broadband network?**
Simplisticly, it is the opposite of a baseband network. With broadband, the physical cabling is virtually divided into several different channels, each with its own unique carrier frequency, using a technique called "frequency division modulation". These different frequencies are multiplexed onto the network cabling in such a way to allow multiple simultaneous "conversations" to take place. The effect is similar to having several virtual networks traversing a single piece of wire. Network devices "tuned" to one frequency can't hear the "signal" on other frequencies, and visa-versa. Cable-TV is an example of a broadband network: multiple conversations (channels) are transmitted simultaneously over a single cable; you pick which one you want to listen to by selecting one of the frequencies being broadcast.

**Question 3: What is an OSI Model?**
The Open Systems Interconnect (OSI) reference model is the ISO (International Standards Organization) structure for the "ideal" network architecture. This Model outlines seven areas, or layers, for the network. These layers are (from highest to lowest):

7.) *Applications:* Where the user applications software lies. Such issues as file access and transfer, virtual terminal emulation, interprocess communication and the like are handled here.

6.) *Presentation:* Differences in data representation are dealt with at this level. For example, UNIX-style line endings (CR only) might be converted to MS-DOS style (CRLF), or EBCIDIC to ASCII character sets.

5.) *Session:* Communications between applications across a net-work is controlled at the session layer. Testing for out-of-sequence packets and handling two-way communication are handled here.

4.) *Transport:* Makes sure the lower three layers are doing their job correctly, and provides a transparent, logical data stream between the end user and the network service s/he is using. This is the lower layer that provides local user services.

3.) *Network:* This layer makes certain that a packet sent from one device to another actually gets there in a reasonable period of time. Routing and flow control are performed here. This is the lowest layer of the OSI model that can remain ignorant of the physical network.

2.) *Data Link:* This layer deals with getting data packets on and off the wire, error detection and correction and retransmission. This layer is generally broken into two sub-layers: The LLC (Logical Link Control) on the upper half, which does the error checking, and the MAC (Medium Access Control) on the lower half, which deals with getting the data on and off the wire.

1.) *Physical:* The nuts and bolts layer. Here is where the cable, connector and signaling specifications are defined.

There is also the undocumented but widely recognized ninth network layer:

9.) *Bozone (a.k.a., loose nut behind the wheel):* The user sitting at and using (or abusing, as the case may be) the networked device. All the error detection/correction algorithms in the world cannot protect your network from the problems initiated at the Bozone layer.

**Question 4: What does an ethernet packet look like?**
See the information below, as described in the National databook. The ethernet packet preamble is normally generated by the chipset. Software is responsible for the destiantion address, source address, type, and data. The chips normally will append the frame check sequence.

| | | |
|---|---|---|
| Preamble | 62 bits | A series of alternating 1's and 0's used by the ethernet receiver to acquire bit synchronization. This is generated by the chip. |
| Start Of Frame Delimiter | 2 bits | Two consecutive 1 bits used to acquire byte alignment. This is generated by the chip. |
| Destination Ethernet Address | 6 bytes | Address of the intended receiver. The broadcast address is all 1's. |
| Source Ethernet Address | 6 bytes | The unique ethernet address of the sending station. |
| Length or Type field | 2 bytes | For IEEE 802.3 this is the number of bytes of data. For Ethernet I&II this is the type of packet. Types codes are > 1500 to allow both to coexist. The type code for IP packets is 0x800. |
| Data — 46 bytes to 1500 bytes | | Short packets must be padded to 46 bytes. |
| Frame Check Sequence | 4 bytes | The FCS is a 32 bit CRC calculated using the AUTODIN II polynomial. This field is normally generated by the chip. |

The shortest packet is: 6 + 6 + 2 + 46 = 60 bytes The longest packet is: 6 + 6 + 2 + 1500 = 1514 bytes

## SIG Sideline

*By Andrew Trauzzi*

**The SIG group needs a new co-ordinator! If you are interested in taking an active role in MUUG activities, please contact the board at <board@muug.mb.ca>.**

The next SIG meeting will be May 16, 1995, at 7:30 PM. As usual, the meeting will be at ISM, 400 Ellice Avenue (behind Portage Place).

### The Fortune File

Sing this one to Michael Jackson's "Beat it"

You're processing some words when your keyboard goes dead,
Ten pages in the buffer, should have gone to bed,
The system just crashed, but don't lose your head,
Just BOOT IT, just BOOT IT.

Better think fast, better do what you can,
Read the manual or call your system man,
Don't want to fall behind in the race with Japan,
So BOOT IT,

Get the system manager to
BOOT IT, BOOT IT,
Even though you'd rather shoot it.
Don't be upset, it's only some glitch.
All that you do is flip a little switch.
BOOT IT, BOOT IT,
Get right down and restitute it.
Don't get excited, all is not lost.
CP/M, UNIX or MS/DOS
Just BOOT IT, boot it, boot it, boot it...

You gotta have your printout for the meeting at two,
The system says your jobs at the head of the queue,
Right then the thing dies but you know what to do,
BOOT IT.

You always get so worried when the system runs slow,
And when it finally crashes, man you feel so low,
But computers make mistakes (they're only human you know)
So BOOT IT,

Call the local guru to
BOOT IT, BOOT IT,
Go ahead re-institute it.
If you're not lucky, get the book off the shelf,
But if you are, it'll do it itself.
BOOT IT, BOOT IT,
Then go find the guy who screwed it!
Operating systems are built to bounce back,
Whether it's a Cray or a Radio Shack.

BOOT IT! BOOT IT!

## Agenda

***for***

***Tuesday, May 9, 1995, 7:30 PM***
***Samuel N. Cohen Auditorium***
***St-Boniface Hospital Research Centre***
***Main Floor, 351 Taché***

| | | |
|---|---|---|
| 1. | President's Welcome | 7:30 |
| 2. | Business Meeting | 7:35 |
| 3. | Short Topic | 7:40 |
| 4. | Coffee Break and Informal Discussion | 7:50 |
| 5. | Feature Topic | 8:00 |

Doug McLean will present the MS Word HTML editor, and Oliver Friedrichs of Freenet will present security. Note that both topics are tentative.

### Coming Up

**Meeting:**
Next month's meeting is scheduled for Tuesday, June 13, at 7:30 PM. Meeting location is TBA because the meeting topic is the annual MUUG barbeque!

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, Doug McLean, know. You can e-mail him at <dmclean@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!