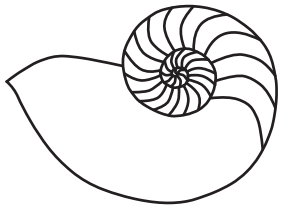
Volume 6, Number 9 September 1994 $2.50



MUUG Lines

Manitoba UNIX® User Group

Newsletter of the Manitoba UNIX® User Group

Pushing the Research Envelope

By Andrew Trauzzi

A friend of mine recently received his doctorate in mathematics — no easy feat. He attempted to explain his thesis to me, but I was lost after the words "It was based on..." I congratulated him and asked when he expects his research to be put to practical use. He was amused by this question and said that most mathematical research does not find a purpose until after the researcher's death!

This contrasts sharply with another friend's research within the computer science field. Basically, his research on object-oriented databases is being put to use *before* he has had a chance to complete it. While quite gratifying, this 'demand' for knowledge has put quite a strain on the academic departments across the continent (and probably the world).

Corporations in today's computer market need an 'edge'. If they can market and deliver a product that appears to be on the cutting edge of technology, then they will do everything in their power to exploit their advantage. This includes stretching, bending, and manipulating (among other various distortions) the truth. I am sure many of you have used products that were released prematurely. Many of these products contain techniques and procedures that were recently developed, or (possibly even worse) developed in house. Even after a number of releases, these products still act as if they were in ßeta release — mainly because that actually are.

Last year, Ken Barker gave an excellent presentation to the MUUG group on object-oriented databases (OODMBS). Basically, his presentation described OODBMS, pointed out some shortcomings that needed to be resolved, and then stated that object-oriented databases do not exist yet. Many people named actual products that claimed that they were OODBMS, and he pointed out that they were merely a relational database with an object front-end. Hopefully, Ken will be speaking to us this year, and give us an update on OODBMS.

OODBMS are an excellent example of the 'tail wagging the dog.' Industry and consumers demanded a product that Universities and researchers had not finished refining. In time, the various issues will be resolved, but not without companies exploiting whatever they can.

I include the C++ FAQ each month because I think that object technology will play an even larger role in software development than it already does. However, software developers need to be aware of the hype surrounding OOP and the huge and possibly unrealistic productivity gains reported by users of OOP. A recent study by Index Summit reports that 42 percent of North American respondents are looking into OOP compared to 6 percent involved with OOP production development. Corporate information systems operations have been slower than developers to adopt OOP because OOP tools enable programmers to create strong applications, but the tools can be difficult to utilize.

This month's speaker is Tim Siemens of Online Business Systems Inc. (one of our corporate sponsors). Tim will be presenting on object technologies and the impact that they have on modern software development.

This Month's Meeting

Meeting Location:
Our next meeting is scheduled for Tuesday, September 13, 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
C++ Q & A 4
UNIX Q & A 5
XKeyCaps Review 6
Ask Monsieur 7
SIG Sideline 8
September 13th Meeting Agenda 8

Printing Courtesy of Xerox Canada Ltd.

RAMBLINGS

Welcome Back!

By Andrew Trauzzi

I hope that everyone had an enjoyable summer! The days are becoming shorter and the nights a little cooler which can only mean that Winter is around the corner!! (sorry) For those of you that would like something to do on those cold, dark nights, here is a schedule of newsletter submission deadlines. This schedule is tentative and may be subject to change. In any case, welcome back to another MUUG year, and I hope to see you at the September meeting.

MUUG Lines Newsletter Deadlines for 1994-1995

			Deadlines		MUUG Monthly
Issue Number	**Month**	**Articles**	**Final Copy**	**Mailing**	**Meeting Date**
Volume 6, No. 10	October	September 17	September 24	October 1	October 11, 1994
Volume 7, No. 01	November	October 15	October 22	October 29	November 8, 1994
Volume 7, No. 02	December	November 19	November 26	December 3	December 13, 1994
Volume 7, No. 03	January	December 17	December 24	December 29	January 10, 1995
Volume 7, No. 04	February	January 14	January 21	January 28	**February 7, 1995** (1st Tues.)
Volume 7, No. 05	March	February 18	February 25	March 4	March 14, 1995
Volume 7, No. 06	April	March 18	March 25	April 1	April 11, 1995
Volume 7, No. 07	May	April 15	April 22	April 29	May 9, 1995
Volume 7, No. 08	June	May 20	May 27	June 3	June 13, 1995
Volume 7, No. 09	September	August 19	August 26	September 2	September 12, 1995
Volume 7, No. 10	October	September 16	September 23	September 30	October 10, 1995

Mailing deadline is always 10 days prior to meeting date, to ensure sufficient advance notice of motions, etc. All copy (ads, columns, meeting notices, etc.) must be submitted by noon on the final copy deadline date, which is one week prior to the mailing deadline. Articles and any other material that isn't time sensitive must be submitted by the articles deadline, which is one week prior to the final copy deadline.

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) 986-3898
Publicity Director	Rory Macleod	(W) 488-5168
Past President	Susan Zuk	(W) 989-3530
Information:	Bary Finch	(W) 934-2723
		(FAX) 934-2620
(or)	Andrew Trauzzi	(W) 986-3898
		(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 1994 / 1995 Program Begins!

By Bary Finch

Welcome back from your summer off! I realize that is just a summer off from MUUG, and not that you all get two months of vacation. It was good to see that we had more of a usual Winnipeg summer, with some good temperatures, and a lot less rain than the last couple of years. But back to "business", and what MUUG is doing for the 1994 / 1995 program.

For this coming year we have a number of exciting presentations for you. Over the summer the MUUG board met and reviewed the results of the survey that we had you fill out during the May meeting. This was your chance to let us know which topics you really wanted to see.

We had a good number of the membership participate, which resulted in a representative opinion of what you want to see in the upcoming months. The board went through the list and chose topics that were the most popular. It was a great checkpoint against what the board "thought" was popular, and what you really wanted to see.

So we have set up a program for 1994 / 1995 that tentatively contains all of the following topics: O-O (Object-Oriented technologies), Network Management (this was the number one topic), WWW (World Wide Web), Mosaic, Firewalls, ATM (that's Asynchronous Transfer Mode, not "how to break into Automated Teller Machines" [*see May's front page, for that! — ed.*]), X-stations, X-server software for PCs, MBnet, and EDI (that's Electronic Data Interchange, not some female entertainer married to a "Steve").

The development of the year's program is one of the most important roles for the MUUG board. It enables us to map out what we're doing each month, and with the addition of the survey results, get a program that both the board and the membership is happy with.

There will be some change to the board this year, as several of the people currently on the board are moving on to other commitments. So we will have a number of new board members, and possibly a number of changes in the roles of the people that remain on the board, as people may want to try another board role.

The new members to the board will be decided on over September, based on asking people that have expressed an interest in participating on the board. Once all the new members have been found, and their roles have been decided on, the entire new board will be introduced to the general membership at our Annual General Meeting (AGM). The AGM will be held during the October meeting, as usual.

In case you're wondering, the meetings will again be held at the St.Boniface Research Center Auditorium. Our gracious host will again be Paul Hope, who arranges for MUUG to use these facilities, and makes sure all the needed audio visual equipment is in place.

Our meeting times will also remain the same, as the second Tuesday of every month, from September to May. June is of course our Annual BBQ, and the Christmas meeting will again be a wine and cheese held in the St.Boniface Research Center atrium. Start time for all our meetings will again be 7:30 p.m.

However, there will be one big change in this schedule in February. We needed to change our meeting from the second Tuesday, as that is February 14, Valentine's Day. The board felt that even though we have very interesting meetings, bringing your loved one along wouldn't be an appropriate way to spend the Valentine's Day evening. So February's meeting will be held on February 7, the FIRST Tuesday of the month.

As a final note, I want to express my disappointment at not being able to attend the first meeting of the season. Unfortunately I will be away on vacation. This was a climate based decision, so there was no real choice.

So I won't see you in September, but I look forward to seeing all of you at the October 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

ONLINE BUSINESS SYSTEMS

TANDEM COMPUTERS

C++ Q&A

By Marshall P. Cline

This month's column examines inheritance and incremental programming. The complete C++ FAQ is now available in a book format — Addison-Wesley Publishers 0-201-58958-3 $32.25.

Question 45: What is inheritance?

Inheritance is what separates abstract data type (ADT) programming from OOP. It is not a 'dark corner' of C++ by any means. In fact, everything discussed so far could be simulated in your garden variety ADT programming language (ex: Ada, Modula-2, C [with a little work], etc). Inheritance and the consequent (subclass) polymorphism are the two big additions which separate a language like Ada from an object-oriented programming language.

Question 46: Ok, ok, but what is inheritance?

Human beings abstract things on two dimensions: part-of and kind-of. We say that a Ford Taurus is-a-kind-of-a Car, and that a Ford Taurus has parts such as Engine, Tire, etc. The part-of hierarchy has been a first class part of software since the ADT style became relevant, but programmers have had to whip up their own customized techniques for simulating kind-of (usually in an ad hoc manner). Inheritance changes that; it adds 'the other' major dimension of decomposition.

An example of 'kind-of decomposition', consider the genus/species biology charts. Knowing the internal parts of various fauna and flora is important for certain applications, but knowing the groupings (kinds, categories) is equally important.

Question 47: How do you express inheritance in C++?

By the '`: public`' syntax:

```
class Car : public Vehicle {
//          ^^^^^^ — ': public' is
//          pronounced 'is-a-kind-of-a'
//...
};
```

We state the above relationship in several ways:

- Car is 'a kind of a' Vehicle
- Car is 'derived from' Vehicle
- Car is 'a specialized' Vehicle
- Car is the 'subclass' of Vehicle
- Vehicle is the 'base class' of Car
- Vehicle is the 'superclass' of Car (this not as common in the C++ community)

Question 48: What is 'incremental programming'?

In addition to being an abstraction mechanism that makes is-a-kind-of relationships explicit, inheritance can also be used as a means of 'incremental programming'. A derived class inherits all the representation (bits) of its base class, plus all the base class' mechanism (code). Another device (virtual functions, described below) allows derived classes to selectively override some or all of the base class' mechanism (replace and/or enhance the various algorithms).

This simple ability is surprisingly powerful: it effectively adds a 'third dimension' to programming. After becoming fluent in C++, most programmers find languages like C and Ada to be 'flat' (a cute little book, 'Flatland', aptly describes those living in a two dimensional plane, and their disbelief about a strange third dimension that is somehow neither North, South, East nor West, but is 'Up').

As a trivial example, suppose you have a Linked List that is too slow, and you wish to cache its length. You could 'open up' the List 'class' (or 'module'), and modify it directly (which would certainly be appropriate for such a simple situation), but suppose the List's physical size is critical, and some important client cannot afford to add the extra machine word to every List. Another option would be to textually copy the List module and modify the copy, but this increases the amount of code that must be maintained, and also presumes you have access to the internal source code of the List module. The OO solution is to realize that a List that caches its length is-a-kind-of-a List, so we inherit:

```
class FastList : public List {
      public:
      //override operations so the cache
      //stays 'hot'
      protected: int length;
      //cache the length here
      };
```

Question 49: Should I pointer-cast from a derived class to its base class?

The short answer: yes — you don't even need the 'cast'.

Long answer: a derived class is a specialized version of the base class ('Derived is-a-kind-of-a Base'). The upward conversion is perfectly safe, and happens all the time (a ptr to a Derived is in fact pointing to a [specialized version of a] Base):

```
void f(Base* base_ptr);
void g(Derived* derived_ptr) { f(derived_ptr); }
// perfectly safe; no cast
```

(note that the answer to this question assumes we're talking about 'public' derivation; see below on 'private/protected' inheritance for 'the other kind').

Question 50: `Derived* -> Base*` works ok; why doesn't `Derived** -> Base**` work?

A C++ compiler will allow a `Derived*` to masquerade as a `Base*`, since a Derived object is a kind of a Base object. However passing a `Derived**` as a `Base**` (or otherwise trying to convert a `Derived**` to a `Base**`) is (correctly) flagged as an error.

An array of Deriveds is-NOT-a-kind-of-an array of Bases. I like to use the following example in my C++ training sessions:

'A Bag of Apples is *NOT* a Bag of Fruit'

Suppose a '`Bag<Apple>`' could be passed to a function taking a `Bag<Fruit>` such as '`f(Bag<Fruit>& b)`'. But '`f()`' can insert *any* kind of Fruit into the Bag. Imagine the surprise on the caller's face when he gets the Bag back only to find it has a Banana in it!

Here's another example I use:

'A ParkingLot of Car is-NOT-a-kind-of-a ParkingLot of Vehicle'

(otherwise you could pass a `ParkingLot<Car>*` as a `ParkingLot<Vehicle>*`, and the called fn could park an Eighteen-Wheeler in a ParkingLot designed for Cars!)

These improper things are violations of 'contravariance' (that's the scientific glue that holds OOP together). C++ enforces contravariance, so you should trust your compiler at moments like these. Contravariance is more solid than our fickle intuition.

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".

UNIX Q&A

Originally Compiled by Ted Timar

Submitted by Andrew Trauzzi

Question 1: Why doesn't redirecting a loop work as intended? (Bourne shell)

Take the following example:

```
foo=bar
while read line
do
    # do something with $line
    foo=bletch
done < /etc/passwd
echo "foo is now: $foo"
```

Despite the assignment "`foo=bletch`" this will print "`foo is now: bar`" in many implementations of the Bourne shell. Why? Because of the following, often undocumented, feature of historic Bourne shells: redirecting a control structure (such as a loop, or an "`if`" statement) causes a subshell to be created, in which the structure is executed; variables set in that subshell (like the "`foo=bletch`" assignment) don't affect the current shell, of course.

The POSIX 1003.2 Shell and Tools Interface standardization committee forbids the behaviour described above, i.e. in P1003.2 conformant Bourne shells the example will print "`foo is now: bletch`".

In historic (and P1003.2 conformant) implementations you can use the following 'trick' to get around the redirection problem:

```
foo=bar
# make file descriptor 9 a duplicate of
# file descriptor 0 (stdin); then connect
# stdin to /etc/passwd; the original stdin
# is now 'remembered' in file descriptor 9;
# see dup(2) and sh(1)
exec 9<&0 < /etc/passwd
while read line
do
    # do something with $line
    foo=bletch
done
# make stdin a duplicate of file descriptor 9,
# i.e. reconnect it to the original stdin; then
# close file descriptor 9
exec 0<&9 9<&-
echo "foo is now: $foo"
```

This should always print "foo is now: bletch". Right, take the next example:

```
foo=bar
echo bletch | read foo
echo "foo is now: $foo"
```

This will print "`foo is now: bar`" in many implementations, "`foo is now: bletch`" in some others. Why? Generally each part of a pipeline is run in a different subshell; in some implementations though, the last command in the pipeline is made an exception: if it is a builtin command like "read", the current shell will execute it, else another subshell is created.

POSIX 1003.2 allows both behaviours so portable scripts cannot depend on any of them.

Question 2: How do I run '`passwd`', '`ftp`', '`telnet`', '`tip`' and other interactive programs from a shell script or in the background?

These programs expect a terminal interface. Shells makes no special provisions to provide one. Hence, such programs cannot be automated in shell scripts.

The '`expect`' program provides a programmable terminal interface for automating interaction with such programs. The following expect script is an example of a non-interactive version of `passwd(1)`.

```
# username is passed as 1st arg, password as 2nd
set password [index $argv 2]
spawn passwd [index $argv 1]
expect "*password:"
send "$password\r"
expect "*password:"
send "$password\r"
expect eof
```

`expect` can partially automate interaction which is especially useful for telnet, rlogin, debuggers or other programs that have no built-in command language. The distribution provides an example script to rerun rogue until a good starting configuration appears. Then, control is given back to the user to enjoy the game.

Fortunately some programs have been written to manage the connection to a pseudo-tty so that you can run these sorts of programs in a script.

To get expect, email "`send pub/expect/expect.shar.Z`" to `<library@cme.nist.gov>` or anonymous ftp same from `<ftp.cme.nist.gov>`.

Another solution is provided by the `pty 4.0` program, which runs a program under a pseudo-tty session and was posted to `comp.sources.unix`, volume 25. A pty-based solution using named pipes to do the same as the above might look like this:

```
#!/bin/sh
/etc/mknod out.$$ p; exec 2>&1
( exec 4<out.$$; rm -f out.$$
<&4 waitfor 'password:'
      echo "$2"
<&4 waitfor 'password:'
      echo "$2"
<&4 cat >/dev/null
) | ( pty passwd "$1" >out.$$ )
```

Here, '`waitfor`' is a simple C program that searches for its argument in the input, character by character.

A simpler pty solution (which has the drawback of not synchronizing properly with the `passwd` program) is

```
#!/bin/sh
( sleep 5; echo "$2"; sleep 5; echo "$2") |
pty passwd "$1"
```

Question 3: How do I check the exit status of a remote command executed via "rsh" ?

This doesn't work:

```
rsh some-machine some-command || echo "Command failed"
```

The exit status of 'rsh' is 0 (success) if the rsh program itself completed successfully, which probably isn't what you wanted.

If you want to check on the exit status of the remote program, you can try using Maarten Litmaath's '`ersh`' script, which was posted to `alt.sources` in January, 1991. `ersh` is a shell script that calls rsh, arranges for the remote machine to echo the status of the command after it completes, and exits with that status.

XKeyCaps

Making Sense of X Window Key Mapping

By Gilbert Detillieux

X Window provides a very powerful and flexible way of redefining the mapping for any key on the keyboard. Unfortunately, the documentation that describes this is scattered in several obscure locations that are likely to be missed by the casual user. Also, the utility provided to work with these mappings, xmodmap, is quite primitive and tedious to work with.

Fortunately, there is an excellent little X Window client application, available as free contributed software, that makes it easy to understand and work with this stuff. It's called XKeyCaps, and it's a must for anyone who's going to spend time modifying keyboard mappings.

When you call up the program, it will try to take a guess at the type of keyboard you've got on your X server. It knows about a large number of these, and presents a nice dialog box that lets you select the right type, in case it couldn't. (You can also compile in a reasonable default, and specify the type you want as a resource or command line option.) Once the keyboard type has been selected, it displays a graphical representation of that keyboard.

What You See

Each displayed key is labeled with the key-top labels you'd expect to see on it, and the hexadecimal keycode value is also shown. (At the lowest level, each key in X is assigned a numeric keycode, which is usually a one-byte number that the keyboard's hardware will generate whenever that key is pressed or released.) As you move the cursor over the displayed keys, several status lines at the top indicate how the key is mapped to keysyms (a symbolic representation of the key, which most X applications will make use of to decide what the key's action should be), and how it is mapped to ASCII codes, if appropriate. You can toggle the modifier keys (e.g. Shift and Ctrl) by clicking on their graphical representation, or by pressing and holding the actual modifier keys. The ASCII value displayed for the other keys will then be modified accordingly.

The program also gives you the option to 'type' into a selected window, by clicking on the key's graphical representation rather than by using the keyboard itself. (This doesn't seem particularly useful, unless the key's mapping has been changed in a non-obvious way.) More useful is the ability to edit the mapping for any key. By holding the right mouse button over the graphic of a particular key, you get a pop-up menu for that key that lets you edit the keysyms for the key, exchange two keys, duplicate the key somewhere else, disable the key, or restore its default mapping.

What You Get

Editing key mappings in this way is fairly easy, but can be time consuming if there are lots of mappings to change. Changes take effect immediately, so you can test things as you go, and back out of a change fairly easily. However, you wouldn't want to go through a manual process like this too often. Fortunately, the program can write to its standard output a series of commands that are suitable as input to xmodmap. Unfortunately, the program doesn't give you the option of directly saving this output to a named file. (You would either invoke the program with output redirected to a file, or just select the output and paste it in to an editor window.)

This ability to produce xmodmap input files is very useful, and saves you having to code such files by hand. Unfortunately, the files are specific to the keyboard type you selected, since they contain keycode commands. (It would be fairly easy to edit such files to use keysym commands instead, which would be more general.)

The man page does warn of the dire consequences of changing key mappings with the wrong keyboard type selected, fortunately. The man page also does a good job of explaining not only how the program works, but gives a lot of background information on keysyms, keycodes, and modifier mappings. This is shamelessly copied from the X Protocol document and InterClient Communications Conventions Manual, as they do indicate. This is a good thing, since it would unlikely ever be read by mere mortals otherwise.

How To Build It

Building XKeyCaps is fairly straightforward if you've got X11R4 or newer, and have the xmkmf script and imake program. The notes on building it are fairly skimpy, but if you've built X clients before, this shouldn't be a problem. Before building, you might want to edit the Imakefile, and define a reasonable default keyboard type, since the program's ability to guess the right type is pretty weak. (It's not the program's fault — that sort of information just isn't always readily available.)

There was one snag with building the program that you're also likely to encounter. At one point, the compilation fails with the following error:

```
cc -O2 -pipe -I./kbds -target sun4 -c  all-kbds.c
"./kbds/sgi5de-r5-map.h", line 17: warning:
undeclared initializer name XK_dead_circumflex
"./kbds/sgi5de-r5-map.h", line 17: illegal initial-
ization
"./kbds/sgi5de-r5-map.h", line 17: cannot recover
from earlier errors: goodbye!
Compilation failed
*** Error code 1
make: Fatal error: Command failed for target 'all-
kbds.o'
```

☛

The problem is that one of the keyboards that the program tries to define, the SGI German keyboard, relies on keysym definitions that may not exist on other systems. (That was the case with the standard X11R5 distribution.) The solution is to edit the file "`kbds/all-kbds.h`", search for lines containing the string "`sgi5de`" and remove (or comment out) all references to this. This won't likely have any adverse effects on the program's usefulness, unless you've got a few German keyboards kicking around.

After that, the compilation and installation should be a breeze. I did have a problem with the program dumping core on an unaligned memory reference when I tried to run it on a DECstation (with a MIPS R3000 processor). That may have had more to do with the way X11R5 was built on that machine than with the program itself (since the program dies within one of the Xt library calls). The program ran without any problems on a Sun SPARC architecture.

XKeyCaps Summary

Name: ..... XKeyCaps
Description: ..... X11 client program to graphically display and edit the X keyboard mapping.
Archive Location: . `export.x.org:/contrib/applications/xkeycaps-2.24.tar.Z`
Archive size: ..... 290025 bytes
Approx Space: ..... 2.5 MB
Time to Install: ..... 10 minutes.
Pros: .. • very handy utility if you've ever had to struggle with xmodmap
• ability to write xmodmap input files very useful
• well written, comprehensive man page
Cons: . • didn't compile correctly due to undeclared keysyms; had to delete definitions for SGI German keyboard
• xmodmap files only written to standard output, rather than allowing a named file
• xmodmap files always contain keycode, rather than keysym, commands; not portable to other keyboard types

Ask Monsieur X

A column in which our resident UNIX expert answers questions submitted by members.

According to Al Maki:

> Cher Monsieur,
> I have a question regarding the Internet, rather than
> Unix itself. My wife is trying to contact a woman in
> Pennsylvania doing a dissertation on a topic
> similar to my wife's. She has been told to use
> "compose" on the mainframe (remember them?)
> with an address of blahblah@psuvm. Now, my wife
> doesn't know how to use compose but she is familiar
> with using elm on "ccu". I assume that "psuvm" is an
> abbreviation which compose knows how to expand,
> but if she doesn't want to use the main frame, how can
> she find out what the true, expanded address is?
> A bientot.
> —
> Al Maki <`almaki@muug.mb.ca`>
> Winnipeg, Manitoba

Cher Monsieur Maki,

Well, many of these mainframe systems (which Monsieur Ex affectionately calls "les vieux dinosaures") have traditionally been connected to a network called BITNET. I am assuming that the brief address you give is such an address (a BITNET node ID is a maximum of 6 characters). Such an address is usually available for e-mail from the Internet by simply adding the ".bitnet" domain name to the end of the address, e.g. `blahblah@psuvm.bitnet`. This sort of address will usually be forwarded to an Internet to BITNET gateway that will know what to do with it.

As for "`compose`", Monsieur Ex would assume that it is just a command or facility on the mainframe for composing e-mail to send out on the network. An equivalent command is available within Elm, or your favorite e-mail package.

Au revoir, M. Ex.

***Monsieur Ex**, a mysterious Frenchman who claims to be an old editor and an expert in UNIX, awaits further mental challenges! BTW: autobogotiphobia is the fear of becoming bogotofied — to make or become bogus!*

Nominations for MUUG Board Elections

By Gilbert Detillieux

Well, October and the MUUG Annual Meeting are fast approaching. One of the most important events at the annual meeting is the election of a new board of directors.

If you feel you would like to contribute to the group by running for a board position, please don't hesitate to do so. Below is the current list of nominees, followed by instructions on nominating others.

Bary Finch	Systems Eng. Rep.	IBM
Rory Macleod	Solutions Consultant	Xerox
Roland Schneider	Computer Consultant	Niche Technology
Brad West	Systems Engineer	Freshwater Fish
Andrew Trauzzi	Programmer/Analyst	City of Winnipeg
Doug McLean		SHL Systemhouse
Doug Shewfelt	System Designer	City of Winnipeg

A total of eight people will be elected. If you want to be nominated, or nominate someone else, send a letter to the group's mail-box or deliver it in person to a member of the election committee. The letter must contain the name, title, and employer of the nominee, along with a short (100 word) biography, and must contain the signatures of the nominee and one other member. The letter must be received no later than September 27, 1994. This is 14 days prior to the annual meeting. If you have any questions about the election, please give Andrew Trauzzi a call at 986-3898 during business hours.

SIG Sideline

By Brad West, SIG Coordinator

I hope everyone had fun, safe summer and is ready for the up and coming year. We have a exciting year planned ahead. Our Special Interest Group (SIG) is set up to combine interest in System Administration as well as Linux to be pursued to a greater depth then allowed by MUUG's regular meeting schedule. Our meetings are held the third Tuesday of each month at 7:30 PM at ISM (400 Ellice Avenue). This years meeting format will consist of a short topic presentation followed by a round table discussion. Some of the topics we are hoping to present this year are: Setting up Archie/ Veronica/Gopher in Linux, Mail gateways, Setting up a machine for general security, the ins and outs of using free software, PPP/Slip, shell programing, setting up X in Linux, Perl, and backups; example of "tars", backup strategies.

If anyone is interested in being a guest speaker at a SIG meeting or you have a specific topic of interest, let me know. I can be reached by email <bwest@muug.mb.ca> or my work phone is 983-0336. There is no specfic topic for next month's meeting — in the event that a topic is not found the round table format will be followed. The next meeting is scheduled for Tuesday, Sept 20, at 7:30 PM. This meeting will again be held at ISM, 400 Ellice Avenue, behind Portage Place. Our host is Wolfgang von Thuelen. He will be waiting in the lobby as of 7:15 PM to let everyone in. Hope to see you at the September meeting!

The Fortune File

Submitted By John Fenske

Written By jelson@condor.cs.jhu.edu (Jeremy Elson)

This was inspired by the recent file making its rounds on the Net describing how to shoot yourself in the foot in a variety of programming languages. Now, the madness is extended to operating systems.

Unix: You shoot yourself in the foot.
DOS: You keep running up against the one-bullet barrier.
MS-Windows: The gun blows up in your hand.
Windows NT: The gun is so huge and unwieldy that you have to keep swapping it from one hand to the other.
OS/2: The gun and the bullet aren't speaking to each other any more.
Mac Finder: It's easy to shoot yourself in the foot — just point and shoot.
AIX: You can shoot yourself in the foot with either a .38 or a .45.
IRIX: The Terminator shoots you in the foot. A T-Rex bites your other foot.
SVR4: The gun isn't compatible with your foot.
Minix: You learn how to shoot yourself in the foot with a Saturday Night Special.
Linux: Generous programmers from around the world all join forces to help you shoot yourself in the foot for free.
HURD: You'll be able to shoot yourself in the foot Real Soon Now.
VM/CMS: IBM shoots you in the foot.

Agenda

for
Tuesday, September 13, 1994, 7:30 PM
Samuel N. Cohen Auditorium
St-Boniface Hospital Research Centre
Main Floor, 351 Taché

1.	President's Welcome	7:30
3.	Business Meeting a) Old Business b) New Business	7:35
5.	Presented Topic This month, Tim Siemens of Online Business Systems Inc. Will be discussing object-oriented issues within corporate environments.	7:45
4.	Coffee Break and Informal Discussion	9:00

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, October 11, at 7:30 PM. Meeting location will be the St-Boniface Research Centre, as usual. The March meeting topic is security. 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!

VMS: \FOOT\ ambiguous: supply more toes.
AMIGA-DOS: The gun works pretty well, except that few people use one and it's impossible to find bullets.
Mach: The bullets work pretty well, but they don't make guns for it any more.
Cray: You shoot yourself in the foot with an Uzi.
MasPar: You shoot all of your friends' feet simultaneously.