Volume 6, Number 6 April 1994 $2.50

MUUG Lines

Newsletter of the Manitoba UNIX® User Group

Sound Bytes

By Arne Grimstrup and Doug Shewfelt

News of the Computer World you would rather not know about!

Phillippe Kahn of Borland was overheard to say “Look, we bought Paradox from Ansa and DBase from Ashton-Tate. If we don’t find another database manufacturer soon whose name starts with ‘A’, we’re going to have to start buying from the ‘B’s!”

In a surprise move yesterday, IBM announced that they were changing the name of the latest release of their mainframe database system from “IMS” to “Advanced IMS”. All functions associated with developing and maintaining the database will be transferred to a new subsidiary company called “Advanced IMS Ltd.”. When asked about the prospects of the aging database system, an IBM spokesperson responded “Don’t worry — we have great plans for this puppy! We know were we can make some quick money with it!”

William Gates called a sudden news conference last Saturday, where he announced “Ok, you’re right! I screwed up!” and gave ownership of Microsoft to Richard Stallman.

The Software Foundation for Law and Government has announced the release of its new “ClipperNet” architecture. The Foundation (which is funded entirely by the U.S. National Security Agency) says that the architecture is a convenient encryption system that automatically forwards a copy of all transactions to the NSA offices.

Microsoft has rounded out its financial offering by pre-announcing “Fraud for Windows.” “We think that this will open a whole net market for us,” said a Microsoft spokesperson. Also planned are “Tax Evasion for Windows”, “Grand Larceny for Windows”, and “Avoiding Anti-Trust Suits for Windows”.

Complaining that C code is too difficult to read, Kenneth Iverson has ported the Unix kernel into a single line of APL.

MIT researchers into Object Oriented Programming have discovered that methods are not inherited, but are instead socialized by their peer methods.

The SAA Association (Students Against Acronyms), had created its first annual top 5 worst computer industry acronyms of the year. Here is the list:

5. POWER — Try not to choose the acronym first!
4. WABI — First coined by Elmer Fudd!
3. WYSIWYG — Now we are really geeking out!
2. EBCDIC — Not new, but loathsome!
and finally...
1. PCMCIA — Acronyms are supposed to help you remember the words right?

This Month’s Meeting

Meeting Location:
Our next meeting is scheduled for Tuesday, April 12, 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.

RAMBLINGS

Some Quick Hacks

By Andrew Trauzzi

In keeping with April's festivities, I have compiled some of my favourite definitions from *The New Hacker's Dictionary.* If you enjoy the following quips, The New Hacker's Dictionary is available at many fine bookstores (ISBN 0-262-68079-3). Some definitions © Eric S. Raymond, 1993.

heisenbug */hi:´zen-buhg/* [from Heisenberg's Uncertainty Principle in quantum physics] n. A bug that disappears or alters its behaviour when one attempts to probe or isolate it. In C, nine out of ten heisenbugs result from uninitialized auto variables.

obi-wan error */oh´bee-won`er´er/* [RPI, from 'off-by-one' and the Obi-Wan Kenobi character in "Star Wars"] n. A loop of some sort in which the index is off by 1. Common when the index should have started from 0 but instead started from 1.

walking drives n. An occasional failure mode of magnetic-disk drives back in the days when they were huge, clunky washing machines. These old dinosaur parts carried terrific angular momentum; the combination of a misaligned spindle or worn bearings and stick-slip interactions with the floor would cause them to 'walk' across a room, lurching alternate corners forward a couple of millimeters at a time.

Walking could also be induced by certain patterns of drive access (a fast seek across the whole width of the disk, followed by a slow seek in the other direction). Some bands of old-time hackers figured out how to induce disk-accessing patterns that would do this to particular drive models and held disk-drive races.

COBOL */koh´bol/* [COmmon Business-Oriented Language] n. (Synonymous with evil.) A weak, verbose, and flabby language used by card wollopers to do boring mindless things on dinosaur mainframes. Hackers believe that all COBOL programmers are suits or code grinders, and no self-respecting hacker will ever admit to having learned the language. Its very name is seldom uttered without expressions of disgust or horror. See also fear and loathing and software rot.

lots of MIPS but no I/O adj. Used to describe a person who is technically brilliant but can't seem to communicate with human beings effectively. Technically it describes a machine that has lots of processing power but is bottlenecked on input-output (in 1991, the IBM Rios, a.k.a. RS/6000, is a notorious recent example). *(sorry Bary ;-) — ed)*

eighty-column mind [IBM] n. The sort said to be possessed by persons for whom the transition from punched card to tape was traumatic. It is said that these people will be buried 'face down, 9-edge first' (the 9-edge being the bottom of the card). The following is inscribed on IBM's 1402 and 1622 card readers:

He died at the console
Of hunger and thirst.
Next day he was buried,
Face down, 9-edge first.

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

A Glimpse of the Future

By Bary Finch

This month I have the rare treat of being able to provide some insight into the future of IBM products. I have just recently been informed of many new directions that IBM is taking. Most of these products are of great interest to the UNIX community, so I thought I’d take the opportunity to describe some of the wonderful things coming in the near future.

That Wascaly WABI

There has been huge interest in the coming availability of WABI (Windows Application Binary Interface) for all those MS Windows fans. This addresses the requirements of one of the larger desktop user “communities” out there. However there are a number of smaller “communities” that have not yet been addressed. One that IBM recognizes is long overdue for recognition is the large application base of CP/M. So I am happy to announce the direction of providing CUE (CP/M User Environment). This will give all of the people that are still big CP/M users a chance to move to the latest technology.

My 409?

Another development direction that everyone is talking about is the PowerPC. IBM’s Bruce Danforth just presented on this at our last meeting. One direction he mentioned was of the PowerPC getting into process control environments. The PowerPC chips to be used for process control are going to have a different numbering scheme than the current 601, 603, 604, 620, ... chips. I have inside info that the numbering will be “4xx”, and that one of the specific chips will be marketed with the slogan “She’s so fine, my 409.”

On the same topic of PowerPC, there will be a new group formed within the Personal Systems (PS) division of IBM, called the Power Personal Systems, or PP Systems. This new division will specialize in products for the PC marketplace, especially the notebook style of computers. In fact the PP Systems will be uninary based, rather than binary, for further weight savings.

Another direction that many of you will have heard of is the WorkplaceOS coming out as a micro-kernel based operating system to run on the PowerPC systems. This OS will likely have several personalities available to run on the the kernel, such as OS/2, or AIX. My sources indicate that there will be other specialty OS’s, based on the WorkplaceOS. One such OS will be WorkshopOS for routers, with many toolkits built in.

A Terabyte of Storage?

On the high end of our product line are the 9076-SP1 systems, where SP stands for Scalable POWERparallel. These are parallel systems that consist of up to 64 RS/6000s today. Cornell University has one that they intend to upgrade to 512 processors. The direction I have heard of is to continue up to 1024 processors. Each of these processors has local disk as they are running in parallel. In fact they could easily have 1GB of disk each. So for all 1024 of the processors they would have 1KGB of disk. Very secure!

Well that’s all the inside info I have to pass on for now. I’d just ask you to remember that this edition of the newsletter coincides with a very special time of year, and you can’t always believe what you read!

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 will look at friends and friend functions, but first we will wrap up operator overloading.

Question 20: Can I create a '`**`' operator for 'to-the-power-of' operations?

No. The names of, precedence of, associativity of, and arity of operators is fixed by the language. There is no '`**`' operator in C++, so you cannot create one for a class type.

If you doubt the wisdom of this approach, consider the following code: `x = y ** z`; Looks like your power operator? Nope. `z` may be a ptr, so this is actually: `x = y * (*z)`; Lexical analysis groups characters into tokens at the lowest level of the compiler's operations, so adding new operators would present an implementation nightmare (not to mention the increased maintenance cost to read the code!).

Besides, operator overloading is just syntactic sugar for function calls. It does not add fundamental power to the language (although this particular syntactic sugar can be very sweet, it is not fundamentally necessary). I suggest you overload '`pow(base, exponent)`', for which a double precision version is provided by the ANSI-C `<math.h>` library.

By the way: `operator^` looks like a good candidate for to-the-power-of, but it has neither the proper precedence nor associativity.

SECTION 6: Friends

Question 21: What is a 'friend'?

Friends can be either functions or other classes. The class grants friends access privileges. Normally a developer has political and technical control over both the class, its members, and its friends (that way you avoid political problems when you want to update a portion, since you don't have to get permission from the present owner of the other piece(s)).

Question 22: Do 'friends' violate encapsulation?

Friends can be looked at three ways: (1) they are not class members and they therefore violate encapsulation of the class members by their mere existence, (2) a class' friends are absorbed into that class' encapsulation barrier, and (3) any time anyone wants to do anything tricky they textedit the header file and add a new friend so they can get right in there and fiddle 'dem bits.

No one argues that (3) is a Good Thing, and for good reasons. The arguments for (1) always boil down to the rather arbitrary and somewhat naive view that a class' member functions 'should' be the *only* functions inside a class' encapsulation barrier. I have not seen this view bear fruit by enhancing software quality. On the other hand, I have seen (2) bear fruit by lowering the *overall* coupling in a software system. Reason: friends can be used as 'liaisons' to provide safe, screened access for the whole world, perhaps in a way that the class syntactically or semantically isn't able to do for itself.

Conclusion: friend functions are merely a syntactic variant of a class' public access functions. When used in this manner, they don't violate encapsulation any more than a member function violates encapsulation. Thus a class' friends and members *are* the encapsulation barrier, as defined by the class itself.

I've actually seen the 'friends always violate encapsulation' view *destroy* encapsulation: programmers who have been taught that friends are inherently evil want to avoid them, but they have another class or fn that needs access to some internal detail in the class, so they provide a member fn which exposes the class' internal details to the PUBLIC! Private decisions should stay private, and only those inside your encapsulation barrier (your members, friends, and [for 'protected' things] your subclasses) should have access.

Question 23: What are some advantages/disadvantages of using friends?

The advantage of using friends is generally syntactic. i.e.: both a member fn and a friend are equally privileged (100% vested), but a friend function can be called like `f(obj)`, where a member is called like `obj.f()`. When it's not for syntactic reasons (which is not a 'bad' reason — making an abstraction's syntax more readable lowers maintenance costs!), friends are used when two or more classes are designed to be more tightly coupled than you want for 'joe public' (ex: you want to allow class '`ListIter`' to have more privilege with class '`List`' than you want to give to '`main()`').

Friends have three disadvantages. The first disadvantage is that they add to the global namespace. In contrast, the namespace of member functions is buried within the class, reducing the chance for namespace collisions for functions.

The second disadvantage is that they aren't inherited. That is, the 'friendship privilege' isn't inherited. This is actually an advantage when it comes to encapsulation. Ex: I may declare you as my friend, but that doesn't mean I trust your kids.

The third disadvantage is that they don't bind dynamically. i.e.: they don't respond to polymorphism. There are no virtual friends; if you need one, have a friend call a hidden (usually 'protected:') virtual member fn. Friends that take a ptr/ref to a class can also take a ptr/ref to a publically derived class object, so they act as if they are inherited, but the friendship *rights* are not inherited (the friend of a base has no special access to a class derived from that base).

Question 24: What does it mean that 'friendship is neither inherited nor transitive'?

This is speaking of the access privileges granted when a class declares a friend.

The access privilege of friendship is not inherited:

- I may trust you, but I don't necessarily trust your kids.
- My friends aren't necessarily friends of my kids.
- Class '`Base`' declares `f()` to be a friend, but `f()` has no special access rights with class '`Derived`'.

The access privilege of friendship is not transitive:

- I may trust you, and you may trust Sam, but that doesn't necessarily mean that I trust Sam.
- A friend of a friend is not necessarily a friend.

Question 25: When would I use a member function as opposed to a friend function?

Use a member when you can, and a friend when you have to. Like in real life, my family members have certain privileges that my friends do not have (ex: my family members inherit from me, but my friends do not, etc). To grant privileged access to a function, you need either a friend or a member; there is no additional loss of encapsulation one way or the other. Sometimes friends are syntactically better (ex: in class '`X`', friend fns allow the '`X`' param to be second, while members require it to be first). Another good use of friend functions are the binary infix arithmetic operators. Ex: '`aComplex + aComplex`' probably should be defined as a friend rather than a member, since you want to allow '`aFloat + aComplex`' as well (members don't allow promotion of the left hand arg, since that would change the class of the object that is the recipient of the member function invocation).

FEEDBACK

A Concise Guide to UNIX Books

Compiled by: Samuel Ko (kko@sfu.ca, sko@wimsey.bc.ca)

Submitted by Andrew Trauzzi

This month's column looks at some general intermediate/ advanced UNIX books.

Unix for the Impatient Authors
Paul Abrahams and Bruce Larson
1992 ISBN: 0-201-55703-7

- **Highly Recommended**. A comprehensive and in-depth reference to Unix. " a handbook you can use both as a manual to learn UNIX and as a ready reference for fast answers to specific UNIX questions."

Unix Power Tools
Jerry Peek, Tim O'Reilly and Mike Loukides
1993 ISBN: 0-553-35402-7

- **Highly Recommended.** Simply great!!! " [It] contains literally thousands of tips, scripts, and techniques that make using UNIX easier, more effective, and even more fun." With a CD-ROM disk containing PD programs and shell scripts. The shell scripts can also be obtained by anon-ftp from `ftp.uu.net` (as `/published/oreilly/power_tools/unix/upt.mar93.tar.Z`).

Unix System V Release 4: The Complete Reference
Stephen Coffin
1990 ISBN: 0-07-881653-X

- Another good book on Unix fundamentals and related subjects.

Unix Desktop Guide to Tools
Pete Holsberg
1992 ISBN: 0-672-30202-0

- A comprehensive guide to numerous Unix utilities.

Modern Unix
Alan Southerton
1992 ISBN: 0-471-54916-9

- Covering selected topics like shells, X Window, networking.

Unix in a Nutshell
Daniel Gilly and O'Reilly staff
1992 (for System V and Solaris 2) ISBN: 1-56592-001-5

- **Highly recommended.** An excellent desktop reference to almost all Unix commands. " a complete reference containing all commands and options, plus generous descriptions and examples that put the commands in context." Also, an edition for 4.3. BSD (ISBN: 0-937175-20-X).

SSC reference cards
SSC staff
1984-93 ISBN: 0-916151-**-*

- These are some good, inexpensive reference/tutorial cards on Unix commands, Bourne shell, Korn shell, emacs, vi, C, C++, etc. e.g. the new "Unix System Command Summary for SVR4.2/Solaris 2.1" (ISBN: 0-916151-61-1). Contact Belinda Frazier `<bel@ssc.com>` or `<sales@ssc.com>` for more info.

The Design of the Unix Operating System
Maurice Bach
1986 ISBN: 0-13-201799-7

- An excellent reference on the internals of System V This book and the next one are indeed highly technical. And if you just want a short case study on Unix, consult a good operating systems text like Modern Operating Systems by A. Tanenbaum or Operating System Concepts by A. Silberschatz, J. Peterson and P. Galvin.

The Design and Implementation of the 4.3 BSD Unix Operating System
S. Leffler, M. McKusick, M. Karels and J. Quarterman
1990 ISBN: 0-201-06196-1

- An authoritative description of the design of BSD Unix ... " It covers the internal structure of the 4.3BSD system and the concepts, data structures, and algorithms used in implementing the system facilities."

Next month, we will look at some UNIX books on shells and shell programming (including the Korn shell), and several books on UNIX editors including vi and emacs.

ADVERTISEMENT — by Roland Schneider

Announcing Cork Board™, the most efficient method of inter-office communication to come along in thirty years. Are you tired of the constant hassle of obscure commands, unreliable operation and user dis-interest in your office bulletin board system? Looking for a better way for people at the office to communicate? Well, look no further! Cork Board™ features proven technology coupled with the most intuitive, easy to use user interface ever devised. Just look at these features:

- Easy one-hand, thumb-operated user interface!
- No typing skills necessary — uses evolutionary OCR technology with near 100% accuracy.
- "At a glance"® message reading is simple, effective, and leaves hands free to hold coffee and a muffin.
- Automatic layering of new messages over old ones.
- Low maintenance: requires only bi-yearly transfer of old messages to the circular file.
- Reliable: message database cannot be affected by power spikes.

For more information on this and other fine organic office products, call Tree Bark Technologies at 1-800-555-CORK. Your boss will love you for it!

UNIX Q&A

Originally Compiled by Ted Timar

Submitted by Andrew Trauzzi

Question 1: Why do I get [some strange error message] when I "rsh host command" ?

(We're talking about the remote shell program "`rsh`" or sometimes "`remsh`" or "`remote`"; on some machines, there is a restricted shell called "`rsh`", which is a different thing.)

If your remote account uses the C shell, the remote host will fire up a C shell to execute 'command' for you, and that shell will read your remote `.cshrc` file. Perhaps your `.cshrc` contains a "`stty`", "`biff`" or some other command that isn't appropriate for a non-interactive shell. The unexpected output or error message from these commands can screw up your `rsh` in odd ways.

Here's an example. Suppose you have

```
stty erase ^H biff y
```

in your .cshrc file. You'll get some odd messages like this.

```
% rsh some-machine date stty: : Can't assign
requested address Where are you? Tue Oct  1
09:24:45 EST 1991
```

You might also get similar errors when running certain "`at`" or "`cron`" jobs that also read your `.cshrc` file.

Fortunately, the fix is simple. There are, quite possibly, a whole *bunch* of operations in your "`.cshrc`" (e.g., "`set history=N`") that are simply not worth doing except in interactive shells. What you do is surround them in your "`.cshrc`" with:

```
if ( $?prompt ) then operations.... endif
```

and, since in a non-interactive shell "`prompt`" won't be set, the operations in question will only be done in interactive shells.

You may also wish to move some commands to your `.login` file; if those commands only need to be done when a login session starts up (checking for new mail, unread news and so on) it's better to have them in the `.login` file.

Question 2: How do I {set an environment variable, change directory} inside a program or shell script and have that change affect my current shell?

In general, you can't, at least not without making special arrangements. When a child process is created, it inherits a copy of its parent's variables (and current directory). The child can change these values all it wants but the changes won't affect the parent shell, since the child is changing a copy of the original data.

Some special arrangements are possible. Your child process could write out the changed variables, if the parent was prepared to read the output and interpret it as commands to set its own variables.

Also, shells can arrange to run other shell scripts in the context of the current shell, rather than in a child process, so that changes will affect the original shell.

For instance, if you have a C shell script named "`myscript`":

```
cd /very/long/path setenv PATH /something:/something-else
```

or the equivalent Bourne or Korn shell script

```
cd /very/long/path PATH=/something:/something-else
export PATH
```

and try to run "`myscript`" from your shell, your shell will fork and run the shell script in a subprocess. The subprocess is also running the shell; when it sees the "`cd`" command it changes *its* current directory, and when it sees the "`setenv`" command it changes *its* environment, but neither has any effect on the current directory of the shell at which you're typing (your login shell, let's say).

In order to get your login shell to execute the script (without forking) you have to use the "`.`" command (for the Bourne or Korn shells) or the "`source`" command (for the C shell). i.e. you type

```
. myscript
```

to the Bourne or Korn shells, or

```
source myscript
```

to the C shell.

If all you are trying to do is change directory or set an environment variable, it will probably be simpler to use a C shell alias or Bourne/Korn shell function. See the "how do I get the current directory into my prompt" in the Feb. issue of MUUGLines.

A much more detailed answer prepared by `<Thomas.Michanek@lin.infolog.se>` (Thomas Michanek) can be found at `ftp.wg.omron.co.jp` in `/pub/unix-faq/docs/script-vs-env`.

Question 3: How do I redirect stdout and stderr separately in csh?

In csh, you can redirect stdout with "`>`", or stdout and stderr together with "`>&`" but there is no direct way to redirect stderr only. The best you can do is

```
(command >stdout_file) >&stderr_file
```

which runs "`command`" in a subshell; stdout is redirected inside the subshell to `stdout_file`, and both stdout and stderr from the subshell are redirected to `stderr_file`, but by this point stdout has already been redirected so only stderr actually winds up in `stderr_file`.

If what you want is to avoid redirecting stdout at all, let `sh` do it for you.

```
sh -c 'command 2>stderr_file'
```

Question 3: How do I ring the terminal bell during a shell script?

The answer depends on your Unix version (or rather on the kind of "`echo`" program that is available on your machine).

A BSD-like "`echo`" uses the "`-n`" option for suppressing the final newline and does not understand the octal `\nnn` notation. Thus the command is

```
echo -n '^G'
```

where `^G` means a _literal_ BEL-character (you can produce this in emacs using "`Ctrl-Q Ctrl-G`" and in vi using "`Ctrl-V Ctrl-G`").

A SysV-like "`echo`" understands the `\nnn` notation and uses `\c` to suppress the final newline, so the answer is:

```
echo '\007\c'
```

GNU Review

By Peter Graham

Greetings. The deadline for submitting this article is March 19th. Can anyone guess what the date is now when I am writing it? I knew you could. :-)

I thought that this month I would start talking about some of the Gnu tools designed to aid in software development. Most people use make and SCCS (the Source Code Control System) to manage their software projects under Unix. As with other Unix programs, there are enhanced gnu products. For make, its ‘`make`’ (funny how they keep the same names...) while RCS is Gnu’s replacement for SCCS. Gnu also offers another product called CVS (a front end to RCS). Conveniently enough, there are three products and three months left in the year, so I’ll talk about RCS this month, Gnu make next month, and then close out the year by talking about CVS.

RCS — the Revision Control System

Why RCS isn’t called SCCS or GCCS I don’t know. Its likely historical but since RCS isn’t a strict superset of SCCS, its probably just as well.

So what is revision control? If you consider the process of software development (or development of any frequently modified, multi-file project) you will realize that it is a process of continual refinement. Even once a program is written, it goes through a life cycle of improvements, bug fixes, and extensions. Furthermore, a program may spawn multiple programs that are based on it. It is helpful if the “versions” of the software that result at each stage of the development process can be maintained rather than being overwritten with each software change. This permits back out to previous versions when mistakes are made, allows different versions (perhaps with different performance characteristics) of a program, and supports the concurrent development of multiple program releases. RCS helps you automate this process in a storage efficient way. It also helps in coordinating the work of multiple programmers working on a single project by ensuring that programmers don’t make conflicting updates to the code. Finally, RCS also allows versions to be merged. This is useful when code development has proceeded independently on different versions of a program resulting in versions (say for two different machine architectures) with different capabilities. By merging the code, the addition of new features from other code versions can be simplified.

All the versions of a given file/system are maintained in a tree structure which reflects their relationships. Programmers may check code in and out of the tree for modification. RCS users are asked to document their changes at check in time. This documentation is associated with each version and can be extracted automatically to produce a change history for the code.

RCS requirements

RCS maintains versions of software using “deltas”. That is, it does not store each version in its entirety. Instead, to save disk space, it stores the differences between a version and the version which precedes it. In order to manage these deltas, RCS uses the Unix ‘`diff`’ program extensively and the diff on your machine must have certain capabilities. Quoting from the installation document...

“*RCS requires a diff that supports the* `-n` *option. Get GNU diff (version 1.15 or later) if your diff lacks* `-n`*. RCS works best with a diff that supports* `-a` *and* `-L`*, and a diff3 that supports* `-E` *and* `-m`*. GNU diff supports these options.*”

If you want to use RCS, make sure you get Gnu diff. Things will work a lot smoother.

RCS installation

RCS is not a standard Gnu install (no ‘`./configure`’ step) but installation is still simple. Since there is no ‘`configure`’ to run, you have to customize the Makefile yourself manually. This is accomplished by editing the Makefile and commenting out incorrect lines and uncommenting correct ones. A section of the README file in the ‘`src`’ directory called “Makefile notes” is provided to assist with this process. Installation then requires you to run a ‘`make conf.h`’ and if this is successful, a ‘`make all`’ followed by a ‘make install’. The edits were pretty simple for my machine (I think I just had to change one incorrect pathname prefix). The ‘`make conf.h`’ will take a while to run (~5 minutes) since it is doing the source code customization normally accomplished (along with the Makefile customization) by ‘`./configure`’. The ‘`make all`’ also took about 5 minutes. Documentation is installed separately by doing a ‘`make install`’ in the ‘`man`’ directory.

RCS Components

RCS isn’t a single program. Instead, it is a collection of several programs, each of which performs a different function. Here is a list of the programs and what they do:

co	• Check Out a source file. Takes a copy of the file from an RCS archive and locks access to that file for other programmers.
ci	• Check In a source file. Places the file into an RCS archive and releases any held locks on the file.
ident	• Identify the version(s) of a source file.
merge	• Incorporate the changes made between two files into a third.
rcs	• Manage rcs file attributes.
rcsdiff	• Compare two rcs file versions.
rcsmerge	• Merge versions rather than files.
rlog	• Print log and other information about RCS files.

RCS Usage

RCS stores archived versions in a directory called RCS. So start by making an RCS directory. Then check in (‘`ci`’) all of your source files. They will disappear as you check them in and be stored for you under the RCS directory. When you wish to update a file just check it out (‘`co`’). Basic usage is as easy as that. Due to space limitations, we’ll have to leave it at that. See the man pages or ‘`rcs.ps`’ (contained in the distribution) for more information.

RCS vs SCCS

Why use RCS instead of SCCS which you probably already have? There are several reasons. First, RCS has a simpler and more intuitive user interface. It also offers enhanced capabilities including improved version identification and more flexible version selection rules (see the documentation). It is also more efficient in both space and time. It is faster than SCCS for most functions and has an improved delta scheme which decreases space utilization for the storage of versions. If this hasn’t convinced you, find an active user and ask them. RCS users are avid supporters of the software.

☞

RCS Summary

Name	RCS (Revision Control System).
Description	Tool to help manage versions and releases of software.
Archive Loc'n	`prep.ai.mit.edu: /pub/gnu/rcs-5.6.0.1.tar.gz`
Archive size	250585 bytes.
Approx Space to Install	2MB.
Time to Install (Sparc-1)	11 minutes.
Pros	• Free, small, and easy to install. • Understandable online man pages AND excellent offline paper describing the system as a whole. • Superset of SCCS in function and performance.
Cons	• Damn few I could think of, although it's obviously not as glitzy as some of the graphically based development systems which offer similar capabilities. (Mind you RCS doesn't cost $2K per seat either.)

Catch you all later. Maybe our baby sitters will be back from the sunny climes of Arizona and we'll be able to make it to the next meeting. If so, we'll see you there. Requests to <`pgraham@cs.umanitoba.ca`>.

This Month's Speaker

This month, Marlon Miller of Xerox Canada Ltd. will be presenting document management and workflow in a client server environment.

Why Would Xerox ("the document company") be speaking to MUUG?

Xerox undertook the first Object Oriented database research in the 1960's and created the first OO Language — Smalltalk, co-developed ethernet, founded the Palo Alto Research Center (PARC), created the first PC (the Alto) before IBM or Apple, created the first WYSIWYG interface (which was the impetus behid MS Windows, System 7, and OpenLook), the laser printer, the optical mouse, the first LAN, etc...

Marlon Miller is Marketing Manager of Document Management Solutions for Xerox Canada. Marlon has been involved with electronic publishing and document management for the last 15 years. Most recently at Xerox, he has focused on integrated document management solution sales combining document library services, workflow and full text retrieval. Additionally, he has significant experience with electronic document composition, SGML and high-speed demand publishing.

He was involved with implementing the ATOS system for the Air Force, the original system which helped spawn the CALS initiative and SGML publishing. He is an executive member of the Canadian SGML Users' Group and is based out of Toronto. Marlon has an undergraduate degree from Carnegie-Mellon University with additional coursework completed at UCLA.

A Day in the Life of a Grad Student

Anonymous

6:30am Wakeup and lie awake in Bed.

6:31 Realize you spent $18 on last night's dinner, means no eating out for the next 6 weeks .

6:32 Hit snooze button. Go back to sleep.

7:00 Wake up suddenly with heart in mouth when you realize you didn't hit the snooze button—you turned it off.

7:01 Fall asleep again.

7:44 Wake up with heart in mouth again.

7:45 Ready to go to school, will shave tommorrow, will eat early brunch at (Denny's/Penny's/Lenny's/Dinko's whatever cafeteria).

8:03 Arrive at school. Realize your foreign officemate arrived earlier today. Must have got more work done.

8:04 Pass by Advisor's office, chat with Secretary to find out if he is coming in today. He is, darn. Need to start work on the draft due this afternoon.

8:15 Read electronic mail.

8:20 Delete mail from students taking 74.206 regarding questions about the class.

8:30 Hate your TA job.

8:55 Depression: too much work to do today.

9:00 For jumpstart: go to Pepsi machine.

9:05 Kick Pepsi machine; promise yourself to call up the company and ask for your money back.

9:06 Wonder why they would beleive you.

9:33 Start printing out loads of stuff that may be vaguely related to your work.

9:41 Early morning stupefaction.

9:42 Mutter racist comments to yourself about your officemate.

9:43 Curse your officemate in a low tone he would not comprehend.

9:44 Feel good about him not grasping English well.

9:58 Finger everyone in the department and most people half way around the world (using the "finger" command, of course).

10:19 Feel sleepy, should not have stayed late playing tetris last night.

10:31 Momentary panic attack!!!!!!!!!!!!

10:43 Edit .plan file.

10:45 Write a shell program to edit .plan more easily

10:59 Drop in at advisor's office and borrow something you don't need & and kinda make him aware you are working hard on your project.

11:05 Perverted daydreams.

11:11 Read electronic news.

11:15 Mid-morning yawn time.

11:34 Start typing junk at a very high key-in rate to pretend you are working hard as your advisor passes by from outside.

11:35 Press the BackSpace key for one and a half minutes until all the garbage you typed in is erased.

11:37 Realize that you can type more than 256 characters per half minute.

11:41 Flirt with the new girl in the department.

11:45 Print out some slides for afternoon's draft + presentation.

11:47 Print them again, you forgot to change the date from last presentation.

11:49 Print another copy in case this one gets lost.

11:51 Completely forget about sueing the coffee-machine company.

12:15 Hunger pangs.

12:20 BigMac/Fries time. Drink a not-so-cold generic can of cola from your desk. Ch-Ching, you just saved 35 cents by buying bulk cola.

1:00 Group Meeting with advisor.

1:14 Sudden awareness of one's shallowness.

1:20 Resentment towards foriegn officemate for sucking up to your advisor.

1:30 Get reminded by your advisor that you need to do some

more work for your literature survey.
1:51 Advisor hands you the reddened copy of your draft for corrections.
1:51:02 The 49 second urge to murder advisor begins!!
1:51:52 Realize that he controls your assistantship/grade/graduation possiblity/graduation date/all job opportunities/and the rest of your life.
1:52:53 Thank him.
1:52:54 Thank yourself for not saying something stupid to your advisor.
1:53:00 Splitting headache #1.
1:59 Check electronic mail, don't reply though, you are too busy to do that.
2:06 More generic cola.
2:17 Oh No, it is my turn to cook tonite :-(
2:30 Sit through the class you were told to sit through.
2:39 Look outside the window make unrealistic plans to quit this degree program and take up a job.
2:42 Wonder why blonde girls are so pretty.
2:48 More perverted day-dreams.
2:51 Close the office door and open a few .gif files.
3:04 Sharpen pencil.
3:06 Worry about never graduating.
3:08 Time to write a letter—NOT! No time for that.
3:10 Rearrange desk.
3:30 Call up bank; see if you have any money.
3:40 Fear of losing aid next Fall.
3:41 Read latex manuals to figure out how to put &$%&% in %$^% format.
3:43 Watch the clock.
4:50 Make plans to do a all-nighter tonite.
4:55 Vow to watch only 2 TV programs.
4:58 Notice Advisor leave.
4:58:01 Sudden sense of freedom.
4:58:03 Go home for quick, short dinner break.
9:00pm Come into the office.
9:01pm The hard working grad student you are, you have to come to the office late at night to "get the work done."
9:03 Check electronic mail.
9:10 Decide it would be a good time to attack those ftp sites since network wont be loaded.
9:40 Run into "since network wont be loaded" traffic and get the pictures into your machine.
9:45 Compress all unwanted research/class directories to make space.
9:59 Back up all your pictures.
10:11 Admire pictures.
10:45 Begin work; Realize you need references.
10:46 Realize its too late today to go to the library.
10:47 Sudden feeling of having wasted the day.
10:48 Sudden feeling of possibly having to waste the night.
10:49 Decide to turn in early and come back very early tommorrow morning.
10:50 Decide to play a Tetris on the system to put yourself in a good mood.
11:15 Play game after game after game to improve your score and get on the scoreboard.
11:45 Realize that your officemate is still at number 6, two notches above you on the scoreboard.
12:20 Play until you beat your officemate into the 7th place. A sense of achievment!! Yes, today was not wasted!!
12:47 Return home to find your roommate watching David Letterman reruns on NBC. Tell him about the "hard working grad student day you had."
1:00 Discuss philosophy with roommate.
1:09 Think about becoming a philosopher and dining with 4 others. (The Dining Philosophers problem, hee hee :-) (Comp Sci joke.)
1:15 Argue with him about politics, why people prefer Japanese cars and whether it is better to set the heat to "hot" or "cold" to defrost the windshields faster.
1:49 Realize neither of you have bought milk today. Get reminded of the "too much milk problem."
2:04 Forget about getting up early. Turn the phone ringer off and go to sleep.

(repeat)

Accent Server News

ACCENTServer™ is a monthly publication of National Information Systems, Inc., (NIS) containing interesting news and views, and some hearsay from around the globe.

Free Subscriptions:
info@nis.com
Article/News Submissions:
accentserver@nis.com

YOU CAN BEAT CARPAL TUNNEL SYNDROME

One of the most talked about disorders affecting millions of computing professionals, Carpal Tunnel Syndrome, can be prevented by following these simple steps:

1. Keep your wrists and elbows flat, not angled, at the keyboard.
2. Take a short break from typing every 15 minutes.
3. Exercise your hands by rotating the wrists, stretching the fingers apart, and bending the hands slowly back and forth.
4. Type gently.
5. Sit at a 90-degree angle to your work — you shouldn't have to stretch or lean over to reach the keyboard.

SEQUOIA INTERNATIONAL ANNOUNCES MOTIF 1.2.3 FOR LINUX, BSDI, ETC.

Sequoia International, Inc. recently announced the availability of OSF/Motif 1.2.3 for these environments: Coherent 4.2, Linux 0.99, BSDI 1.0, FreeBSD 1.0.2, and NetBSD 0.9. OSF/MOTIF is one of the most widely accepted graphical user interfaces for the X Window System.

Packages contain the complete runtime and development environment for Motif 1.2.3, which includes the following:

- The Motif Window Manager (mwm)
- Shared Library (libXm) [Linux Only]
- Static Libraries (libXm, libMrm and libUil)
- Header and Include Files
- Complete On-line Manual Pages
- Source code to OSF/Motif demo programs
- Complete OSF/Motif Users Guide

For order information as well as technical details on the above products, contact:

Sequoia International, Inc.
600 W. Hillsboro Blvd, Suite 300
Deerfield Beach, FL 33441

E-Mail: info@seq.com
Phone: (305) 480-6118
Fax: (305) 480-6198

SIG Sideline

By Brad West, SIG Coordinator

Once again we had a strong turnout at our last SIG (Special Interest Group) meeting on Tuesday, the 15th. The meeting started out with a round table discussion. The topics discussed ranged from video adapter card problems in Linux to writing and modifying printcap files. Of course the discussions turned to what's new in Linux. The official release of Linux has finally reached 1.0. The release will now take two base paths, 1.01 path will be the hacker paradise version, and 1.1 path will be the stable release platform. As the hacker's version becomes stable, the release will be incorporated in the stable version. One of the next big projects being worked on in Linux is the network code.

The presentation for the evening was sendmail given by Gilles Detillieux. Gilles gave a great overview presentation of the workings of sendmail. Some of sendmail special features presented were: point to point, delivery and forget, aliasing and forwarding, mailing list, and error notification. Topics covered were: installation, sendmail components, to MX or not to MX, and the approach to sendmail setup. The pros of sendmail presented are: it is a public domain program with the source code available, it is relatively bug free, and is configurable and flexible. The cons are: its step learning curve, tricky rules and, that fact that it is an old program.

No specific presentation is scheduled for the next meeting to date, in the event that a speaker is not found, we will continue with the round table format. 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. The next meeting is scheduled for Tuesday, April 19, 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 April meeting.

Internet Corner

Compiled By Andrew Trauzzi

Question: How do I send mail to other networks?

Mail to the Internet is addressed in the form <user@domain>. (Without the '< >'). Remember that a domain name can have several components and the name of each host is a node on the domain tree. So, an example of an Internet mail address is <atrauzzi@mona.muug.mb.ca>.

There are several networks accessible via e-mail from the Internet, but many of these networks do not use the same addressing conventions the Internet does. Often you must route mail to these networks through specific gateways as well, thus further complicating the address.

Here are a few conventions you can use for sending mail from the Internet to three networks with which Internet users often correspond.

Internet user to BITNET user:
user%site.BITNET@BITNET-GATEWAY
e.g. gsmith%emoryu1.BITNET@cunyvm.cuny.edu

Internet user to UUCP user:
user%host.UUCP@uunet.uu.net
user%domain@uunet.uu.net

Internet user to SprintMail user:
/G=Mary/S=Anderson/O=co.abc/ADMD=SprintMail/C=US/
@SPRINT.COM (case is significant)

Internet user to CompuServe user:
Replace the comma in the CompuServe userid with a period, and add the compuserve.com domain name.

CompuServe user to Internet user:
>Internet:user@host
Insert >internet: before an Internet address. ☛

Agenda

for
Tuesday, April 12, 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 7:35
a) Old Business
b) New Business

5. Presented Topic 7:45
This month, Marlon Miller of Xerox Canada Ltd. will be presenting document management and workflow in a client server environment.

See the writeup on page 8 for more info.

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.
below for details).

Coming Up

Meeting:
Next month's meeting is scheduled for Tuesday, May 10, 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!

Internet user to MCIMail user:
accountname@mcimail.com
mci_id@mcimail.com
full_user_name@mcimail.com. ✒