Volume 4, Number 6 April 1992 $2.50 (*cheap*)

TUUG Lines

Technical UNIX® User Group

Newsletter of the Technical UNIX® User Group

Have You Read Any Good UNIX Books Lately?

By Richard Kwiatkowski

While reading the March newsletter's Feedback column, I noticed that someone asked for a recommendation on any good UNIX books. This reminded me of my experience and frustration when I went looking for books on UNIX. When I first found out that our new computer system was going to be UNIX based, I rushed out to the book stores in search of books on UNIX. To my surprise, there were few, if any, books to be found. It makes it difficult to compare which book is better when you have such a small selection to from which to choose, and more importantly, you know nothing about UNIX. I asked the clerk if they had any more books on UNIX; he typed into his computer terminal, and up popped a screen full of choices. Which one would I like to order? This was even more difficult than picking a book off the shelf, because you can't flip through it before you buy it. I bought two books off the shelf and ordered one more.

While I was learning Unix and I found a need to look up a specific UNIX command, I would read through all the books that I had. After a while, it became clear that the most complete book that I had was called *UNIX System V Release 4 — The Complete Reference*, by Stephen Coffin.

I would like to create and maintain a list of UNIX books for the user group. In order for me to do this, I would like you to send me your comments on the UNIX books that you have read. You can send your comments to:

Richard Kwiatkowski
838 Atlantic Ave.
Winnipeg, MB R2X 1L4

or FAX it to me at 943-4797, or call me at 589-4857. ✒

THIS MONTH'S MEETING

Meeting Location:
This month, the meeting is to be held at the Senate Chambers, room 245 in the new Engineering Building of the University of Manitoba, Ft. Garry Campus.
The meeting is set to start at the usual time – 7:30 PM, on April 14, 1992. The Senate Chambers are on the main floor, near the entrance that faces University Centre.

Meeting Agenda:
See last page for details.

INSIDE THIS ISSUE

Getting Involved, ... and Loving It!

By Gilbert Detillieux

Wow! The responses to my earlier pleas for more articles are still coming in. For the first time ever, I've got material ahead of time, ready for the next month's newsletter. A big "thank you" to all our contributors, particularly to our regular contributors. Locally written material is what makes this newsletter special.

We've had to rearrange our newsletter schedule for May. As a result, the deadline for submissions for the May edition is now April 17 (Good Friday). The newsletter is likely to be mailed out earlier than usual as well. In any case, this advanced deadline shouldn't be a problem, since I have all the material I need for the next edition.

If you have something you want to submit, go ahead anyway. If I can't fit it in for May, there's always the June edition.

As was announced at the last meeting, TUUG will have a booth at the Muddy Waters Computer Society Computer Fest, to be held at the convention centre on Sunday, April 26, 10:00 AM to 6:00 PM. By being present at this event, we hope to increase the group's visibility, and promote the recent changes, such as the name change, and the affiliation with UniForum Canada. We need volunteers to occupy the booth for that day (or for a short portion of the day). Ideally, we would like to have 6 to 8 people, each putting in about 2 hours. If you are interested in helping out for a shift, please contact Susan Zuk.

Enjoy the newsletter. Hope to see you at the April meeting, and at the Computer Fest too.

The 1991-1992 Executive

President:	Susan Zuk	(W) 788-7312
Past President:	Eric Carsted	1-883-2570
Vice-President:	Richard Kwiatkowski	589-4857
Treasurer:	Rick Horocholyn	(W) 474-4533
Secretary:	Roland Schneider	1-482-5173
Membership Sec.:	Allan Moulding	269-8054
Mailing List:	Gilles Detillieux	489-7016
Meeting Coordinator:	Eric Carsted	1-883-2570
Newsletter editor:	Gilbert Detillieux	489-7016
Information:	Susan Zuk	(W) 788-7312
		(FAX) 788-7450
(or)	Gilbert Detillieux	(H) 489-7016
		(FAX) 269-9178

Copyright Policy and Disclaimer

Our Address

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

Internet E-mail:
tuug@cs.umanitoba.ca

Group Information

The Technical UNIX User Group meets at 7:30 PM the second Tuesday of every month, except July and August. The newsletter is mailed to all paid up members one week prior to the meeting. Membership dues are $20 annually and are due at the October meeting. Membership dues are accepted by mail and dues for new members will be pro-rated accordingly.

PRESIDENT'S CORNER

You Are Invited to Participate...

By Susan Zuk, President

Can you believe that winter is over? It even feels like spring. The UNIX group is halfway through it's mandate. The accomplishments of the group have been extensive in the past 12 month. Our membership has more than doubled and new people are joining all the time. We held a successful UNIX Symposium and are in the middle of making some big decisions which could change the group quite significantly. We also have active participation in our newsletter and a very supportive membership. From the response we have received from the survey, we can see that members are interested in what the group is doing and where it will be going in the future.

We will be relaying summary information of the results of the survey in the next newsletter and will work towards implementing your suggestions. For those of you who are interested in helping the group in its quick growth path, we will be contacting you shortly. Voting has been completed on the name change as well as the affiliation to UniForum Canada. Based on the results of the vote we will proceed with changing our name to the Manitoba UNIX User Group and with the motions to become an affiliate of UniForum Canada. We will be informing you of the status in upcoming meetings and newsletters. The executive would like to thank all those who took the time to respond. The group would like to start our fall season with a large promotion of the new name and affiliation. For those who are interested in personally joining UniForum Canada, we will provide you with this information in the next issue of the newsletter.

On Sunday, April 26th, the Muddy Water Computer Society is holding a Computer Fest. Our group will be represented at this show. The show is being held at the Convention Centre from 10:00 A.M. to 6:00 P.M. If you can help out for an hour or 2 please give me a call. We will be promoting our group to the Winnipeg community and would appreciate your help.

In March I attended a conference in Toronto called the Open Systems and UNIX Executive Symposium. This symposium was comprised of Case Studies of real life situations where UNIX and Open Systems are being implemented. Companies such as the Toronto Stock Exchange, American Airlines, American Express, and the Department of National Defence spoke on their experiences. Topics which were highlighted included reasons why the move was being made to Open Systems, how it was being implemented, what to watch for when going through the process, why to make the change, who to have on your side and if it has made a difference. I will be relaying what occurred in the next issue of our newsletter.

That's all I have to say for now. Our next meeting should prove to be very interesting. I hope you are attending and can help us welcome Amdahl Canada in presenting their UNIX offering for mainframe systems. ✎

THE FORTUNE FILE

Submitted by Ken Stewart

"Almost Live" (a local comedy show in Toronto) did a sketch on Microsoft's "management style" because of the recent Buisness Week article.

- Employee of the month gets a gold watch and a small Mediterranean island.
- At board meetings, all members must dress like their favorite Star Trek character. Bill always gets to be Spock.
- Corner offices with a view are awarded to programmers who can make Mountain Dew come out of their noses at lunch meetings.
- At Christmas time, everyone gets presents from their "Secret Geek."
- Promotions are guaranteed for all executives who pick Bill first for their softball team.
- Each afternoon, everyone runs over to IBM, rings the doorbell, and hides.
- Best idea of the week gets you an hour alone with Bill's money.

Submitted from all over the place

Fake mail header found in a news posting:
X-Mac-Bust: The Best Way To accelerate a Macintoy is at 9.8m/s^2

Found in a news posting signature:
"My karma ran over my dogma."

Another signature:
"Warning: Do not drive with Auto-Shade in place. Remove from windshield before starting ignition."

Notable quote:
"If anyone disagrees with anything I say, I am quite prepared not only to retract it, but also to deny under oath that I ever said it." —T. Lehrer

Main's Law:
For every action there is an equal and opposite government program.

Ask Monsieur Poisson D'Avril

Our resident Unix expert, Monsieur Ex, who normally answers questions submitted by members, is away with a bad case of spring fever. An old friend of his, Monsieur Poisson D'Avril, fills in.

Edited by Gilbert Detillieux

Dear Editor,

I couldn't resist a short addendum (or two):

In "The Internet, UUCP, Mail, News, and All That", TUUG Lines 4(5), March 1992, Roland Schneider writes:

Say I'm visiting a friend at IBM in New York ... [all I have to do is] type "telnet rum.ee.umanitoba.ca".

Well, Roland neglects to mention that, assuming he had a friend at IBM in New York ;-) , he would first have to spend a week doing paperwork to get authorization for his packets to make it past the bit-o-matic packet dicer and slicer which lives at the gateway out of the building! Sometimes there's more to life than meets the technological eye :-(.

Further, Roland writes:

... too slow for ... X ...

Not necessarily so Roland! Say that your friend from IBM in New York is visiting your other friend at the University of Calgary, and say that your IBM friend wants to show off his latest whiz-bang X application for displaying the behavior of parallel processing systems. All he has to do (there's no authorization bureaucracy in Calgary) is telnet to his workstation in New York, set his DISPLAY environment variable to point at the X server on the workstation in Calgary, start up his application, and Voila! (Apologies to Monsieur Ex for lack of the correct accent.) It turns out that packet round trips from Calgary to New York to Calgary, as reported by ping, can average 800-1500 ms on a good night. A bit sluggish perhaps, but more than adequate for effective demonstrations of system monitoring and animation software. Ain't technology grand?

Finally, I notice that ee.umanitoba is now naming their workstations after their beverage of choice (not surprising for an ee department, but commendable nonetheless). Inspired by their progressive attitude, I'm about to rename all of my workstations after Canadian beers.

Keep up the good work out there. I follow your continued success with great delight.

Your friend at IBM in New York, - Doug K.

(soon to be dnk@oldstock.watson.ibm.com)

(or dnk@jobhunting.fired_for_abuse_of_internet.unemploymentline.gov)

P.S. The opinions expressed above are strictly those of the author, and in no way represent the views of the IBM Corporation.

M. D'Avril's reply:

While I would prefer *les bons vins Français* to Canadian beers, your *bon-vivant* spirit and patriotism (for an expatriate Canadian) are quite commendable. Perhaps we'll be able to reach you some day at dnk@moose_head.watson.ibm.com?

Glad to hear you still enjoy the newsletter. Please keep in touch.

Cher Monsieur Ex,

Votre réponse à la question «How do you combine two files into one file» est très dangereuse. Si on utilise

```
cat file1 file2 > both
```

quand les fichiers «file1» et «both» sont égaux, un désastre va suivre.

Cheers, Michael Doob.

M. D'Avril's reply:

Merci pour votre lettre. Your concern for the safety of novice users is appreciated (No one ever accused M. Ex of being overly cautious.) Novice users of the C-shell (csh) should look into the **noclobber** option to safeguard against this sort of accident.

Q. What is the difference between a 386 and a 486?
A. One hundred, last time I checked. *Mais sérieusement*, as far as the software is concerned (even for O.S. software), the two are identical. Both implement the 386 architecture and instruction set. The differences are in the hardware organization. The 486 uses more pipelining to improve speed, and has more circuitry on the chip, such as an integrated floating point coprocessor.

In short, versions of UNIX for the 386 should work fine on a 486 system.

Q. What is life?
A. *Ah, une question philosophique!* This is a tough one, so I'll ask the computer to help me with this one:

```
% whatis life
life(6)       - John Conway's game of life
```

As I've always suspected, life is merely a game. Now, if I can figure out who John Conway is, I might get somewhere:

```
% whois "John Conway"
No match for "JOHN CONWAY".
```

I didn't think it would be that easy. I guess if I were responsible for life, I wouldn't want to leave a forwarding address either. *À la prochaine, peut-être.*

***M. Ex**, a mysterious Frenchman who claims to be an old editor and an expert in UNIX, will hopefully be all better in time for next month's column.*
*Incidentally, **Poisson D'Avril** is french for "April Fool."*
***Gilbert Detillieux** is a founding member of TUUG, and a past president, yet he was naïve enough to give these two cretins space in the newsletter without checking their credentials. Must be a slow month.*

The Wonders of ElectronicMail

or Why I Hate FAX Machines

By Roland Schneider

What makes electronic mail (or e-mail) such a useful tool? Consider this: it combines some of the best features of a telephone on a PBX, an answering machine, a fax machine, and file exchange via floppy disk.

An example: A client wants to tell me that he’s run into trouble with a software package. He sends me e-mail containing a description of the problem, along with output generated by the program and the contents of a configuration file. Even if I’m away from my desk, the message will be waiting on my computer when I come back. I look at the message and electronically forward the relevant parts to the program’s author, while saving the original message so that I can track the service call. The author turns out to be on vacation, so his e-mail system automatically replies to tell me when he’ll be back, and also gives me the address of a co-worker who may be able to help. I send the same message, which was automatically saved, to the co-worker. A few hours later, he replies with a message containing a binary patch to the software, encoded into displayable characters, which I then forward to my client, along with detailed instructions describing how to extract the patch from the message and install it.

The process is the same whether the people involved are in a single building, a city, country, or almost anywhere in the world. I can use the same messaging system to make a squash date with the guy in the next room, or even to leave myself a note to buy flowers for Mother’s Day. Messages will sit in my electronic mailbox until I deal with and delete them. If a message arrives while I’m at my computer, it beeps and the little flag on the mailbox icon goes up. I can deal with the message immediately, or leave it until I finish whatever I’m doing.

How it Works

So much for trying to convince you that e-mail is wonderful. Now, how does it work? Most mail systems are split into two pieces — the user agent (UA), which you use to send and read mail, and the routing software, called the message transfer agent (MTA), which handles the actual transmission of mail to its destination. As long as everything is configured properly, users need only be concerned with the user agent, since it provides the user interface, including forwarding and message archival features. There are several to choose from, some commercial, some free, some for serial terminals, some for X-windows, etc. Most can be set up to work with whatever routing software is available.

Of course, it’s in the routing software where all the magic happens. How does a message from `czarow@eleceng.ee.queensu.ca` (a friend who's a professor in Kingston) to `rsch@ee.umanitoba.ca` (me) get to its destination? Well, that depends on how the different systems involved are hooked together. In this case, the likely route will be:

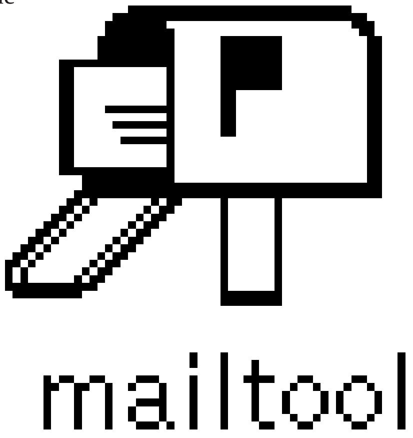
1) The message originates on `cz1.ee.queensu.ca`. This small, personal machine doesn’t know much, except that the machine `eleceng.ee.queensu.ca` knows more, so the message is sent there using SMTP (simple mail transfer protocol) over a TCP/ethernet connection.
2) `Eleceng` doesn’t know much either, so it sends the message to `qucdn.queensu.ca`, an IBM mainframe, by the same method.
3) `Qucdn`, or rather, the person administering e-mail on it, knows enough to recognize the `ee.umanitoba.ca` part of the address and contact `eeserv.ee.umanitoba.ca` directly via the Internet, although it could have just recognized `umanitoba.ca` and contacted `ccu.umanitoba.ca`, which would have forwarded the message to `eeserv`.
4) Eeserv.ee.umanitoba.ca looks up what appears to be a local address in the mail address alias database, and finds that it’s really destined for `rsch@rol.selkirk.ee.umanitoba.ca`, which isn’t on the network, but is reachable via UUCP, so it queues the message and waits for my machine to call up. (It could call me too, but I didn’t set it up that way)
5) My machine automatically polls `eeserv` for mail four times per day, and picks up the message. The flag on my mailbox icon goes up, and I have my mail — all I have to do is double click my mouse on the icon.
6) I can reply to the message by clicking on the “Reply” button, which will automatically fill in the address and subject. The path back to my friend will be similar, but not the same: from `rol` to `eeserv` to `ccu` to `eleceng`.

Some Technical Stuff

With the possible exception of the IBM mainframe, the machines in this example all run a program called sendmail. Users don’t interact with sendmail — it runs as a daemon and coordinates mail activity according to the rules in a the configuration file `sendmail.cf`. Sendmail does two things: it examines (and possibly modifies) the “`To:`” and “`From:`” addresses in a message, and then, as is appropriate for each particular destination address, either executes an external program to deliver the message, or contacts another sendmail on another machine via SMTP over a TCP link. Determining how to rewrite the addresses and which machine to contact or which program to run is the job of the rules in the `sendmail.cf` file.

Figuring out which one of millions of machines to contact is made easier by the strategy of simply contacting someone who knows more than you do. A personal machine contacts a departmental machine, which contacts a company machine, which contacts another company's machine, which contacts the appropriate departmental machine, which in turn contacts a personal machine in the department, which delivers the message to the recipient. That's why the dot-separated parts of a mail address refer to organizational entities — each mail forwarding machine only needs to determine if the address is within it's "domain", in which case it forwards down to the smaller organizational entity to the left of "its" part of the address (e.g. company to department) or, if it isn't, it forwards up. (e.g. company to regional)

The Internet-wide domain name service contains information which can be used to short-circuit this procedure in some cases — that's why `ccu.umanitoba.ca` contacted `eleceng.ee.queensu.ca` directly, instead of going through `qucdn.queensu.ca`. Note that machine names frequently form part of an e-mail address, but this isn't strictly necessary, it's just convenient since the organizational structure of machine and e-mail administration is usually similar.



The ability to execute an external program gives sendmail much added flexibility. For example, if a step in mail routing requires using UUCP, the `sendmail.cf` file specifies that the "`rmail`" program is to be run with arguments determined from the destination address. Rmail accepts the message from sendmail and queues it for forwarding via UUCP. Similar special-purpose programs can be used to forward to other mail systems in different environments, like VAXes running VMS, Macs using Quickmail, etc. You can set things up so that e-mail destined for people who don't use computers is automatically printed in the department office so that it can be put in an old-fashioned paper-type mailbox. There are even systems which allow you to send mail to a special address which causes the message to be translated into a fax image and sent to the phone number specified in the address.

Alternatives to Sendmail

Not all versions of UNIX come with sendmail. However, there is freely redistributable source available, so it is generally possible to install it. Alternatively, there is smail, also available for free, which can be used either with or without sendmail, and provides sendmail-like functionality for UUCP links. Generally, smail is simpler than sendmail, and may be appealing for that reason alone. Some vendors may provide other mail-routing facilities.

A much better alternative, which avoids much of this hodge-podge, is to use a mail system based on the X.400 mail protocol. This standardizes the format of mail addresses and messages and provides extra features like registered mail, messages containing binary information, etc. It also avoids much of the complexity of the `sendmail.cf` file by specifying only a single way to write an address. Unfortunately, X.400-based systems are not yet widespread, and I don't know of any free software implementing it. Generally, X.400 systems have gateways to non-X.400 systems, but most of the features and simplicity are lost as soon as a gateway is involved.

Why do I hate fax machines? Because they have offered an easy, although inferior, alternative to e-mail, which has delayed the development, adoption, and implementation of mail standards like X.400. Because e-mail involves sending information between computers made by different vendors, widely implemented industry-wide standards are essential. A ideal e-mail system should allow not only text, but also pictures and digitized sound to be sent, should provide for encryption and sender authentication (like a signature) for legal purposes, and should be as wide-spread as fax machines are now. Well, some day.

In the meantime, e-mail available with existing systems, although not perfect, is an indispensable tool. Like the telephone, it becomes more useful, even essential, as more and more people and companies are connected.

UNIX BITS

Files with Holes

Winnipeg streets have potholes in the spring, so can the knees of your pants, but disk files? Sure, why not?. What happens if you open a file, `write()` a few blocks, and then `lseek()` way past the end of the file and `write()` some more? What gets put in the middle? Nothing, if your version of UNIX supports files with holes. What if you try to read that nothing? You get zeros.

The nothing does not occupy any space on your disk, although `ls -l` will be fooled and will tell you that you have a huge file. Use `ls -s` to get the real story. Not many UNIX utilities understand files with holes, so if you copy a file with `cp`, it will read all those fake zeros, making the copy a filled-in version of the original, which can use a lot of disk space. The tape utility `tar` has the same problem, but `dump` does not. It's safe to rename these files with `mv` as long as you stay on the same file system.

One of the most common files with holes is the core dump file "`core`." The system just dumps out the various parts of the program's virtual address space, creating a file which may appear to be 8 megabytes long, when it actually occupies less than 200K.

Sun RPC Programming III

98.34% of All Statistics Are Made Up.

By Scott Balneaves

This time we have a slightly less trivial (but only slightly) RPC programming example. However, we'll explain the powerful protocol compiler rpcgen(1) that can create most of the "drudge work" code for you, leaving your time free to do the more interesting parts of your application.

The rpcgen(1) language specification

I won't go into all of the boring details. You can check out your manuals for these bits. However, I will gloss over the major ingredients of the language.

When you run a protocol specification language file (usually ending in a .x suffix) through the rpcgen compiler, you'll usually end up with 4 files. for a file called barfaz.x, you'll end up with: barfaz_clnt.c (the client side stubs), barfaz_svc.c (the server registration procedures), barfaz_xdr.c (the XDR stubs to translate your data) and barfaz.h (the header file for your application). We'll see how the contents of the .x file map into real C programs below.

The language itself resembles C very closely. I'll use the protocol specification file from this month's sample code as our working example:

```
/*
 * FILE stats.x:
 * RPCL Protocol definition for a
 * statistics server.  PROCS include
 * SUM, MEAN, and STDDEV.
 */
const MAX_DATA = 256;

typedef double stat_data<MAX_DATA>;

program STATPROG {
       version STATVERS {
              double SUM(stat_data)         = 1;
              double MEAN(stat_data)        = 2;
              double STDDEV(stat_data)      = 3;
       } = 1;
} = 0x20000099;
```

The first working line is pretty obvious. It sets up a symbolic constant called MAX_DATA to 256. The next line looks somewhat familiar to a C typedef, but it's not exactly the same. It defines an array of doubles (up to a maximum of MAX_DATA, 256), to be the type stat_data. You'll notice we used the <> bracket pair, instead of the normal C [] pair. You can use these, if you want, but if you do, then you will always send to the server 256 doubles of data. With the angle brackets (<>), you'll only send down the data that you actually use. In the file stats.h you'll see the definitions:

```
#define MAX_DATA 256
typedef struct {
       u_int  stat_data_len;
       double  *stat_data_val;
} stat_data;
```

You'll notice that the stat_data is now a structure with both a pointer to a double, and the length of the allocated array of doubles (in units of doubles, *not* in bytes). You don't have to worry about allocating the memory for the data values other than in your client program. The generated XDR routines will handle the transfer of this structure.

You'll remember the numbers we had to assign to any RPC procedure: The program number, the version number, and the procedure number. These are handled by the constructs:

```
program STATPROG { ... } = 0x20000099;
```

and

```
version STATVERS { ... } = 1;
```

and produce the following lines in stats.h:

```
#define STATPROG ((u_long)0x20000099)
#define STATVERS ((u_long)1)
```

You'll remember how we said that you could have multiple procedures in an RPC server? This program is an excellent example of this. We have defined three procedures:

```
double SUM(stat_data)          = 1;
double MEAN(stat_data)         = 2;
double STDDEV(stat_data)       = 3;
```

All three (SUM, MEAN, STDDEV) accept a type stat_data as a parameter, and return a double as a result. They have procedure numbers 1, 2, and 3 respectively. These result in the following definitions being placed in the stats.h file:

```
#define SUM ((u_long)1)
extern double *sum_1();
#define MEAN ((u_long)2)
extern double *mean_1();
#define STDDEV ((u_long)3)
extern double *stddev_1();
```

As you can see, rpcgen will even be so nice as to create the function prototypes we need. rpcgen will normally tack on the version number of the procedure to the end of the procedure name. Since we already defined these procedures to be version 1, we get functions sum_1, mean_1, etc.

In addition, rpcgen generates the XDR routines needed to send the stat_data structure down the network, and the server and client stubs. The server stubs basically consist of the code needed to register your service, and dispatch incoming requests to your routines. The rpcgen compiler will automatically create both a TCP and a UDP version of your routines for you, so that your clients can simply choose the transport that they want. In fact, for the server at least, all you need to do is write the actual routines themselves that are doing the actual work of your RPC server. On the client side, the stubs are created for you to make the calls. All you need to do is create the client handle (the “connection” between the client and the server) and make the calls.

The Statistics Server

Our hypothetical client/server example this time is a statistical server. It has 3 procedures. Each are passed the list of data elements, and return a double. Procedures include ones that calculate sum of the data, mean (average) of the data, and the standard deviation. What’s important here isn’t so much *what* the server is doing, but *how*the pieces fit together.

I’ll only publish here the parts that you’ll need to type in. However, take a look at the stats_svc.c, stats_clnt.c, stats_xdr.c, and stats.h files generated by the rpcgen compiler. I’ve included a simple Makefile for this example, as it’s gotten complicated enough to merit one. To compile the server, type:

```
make server
```

For the client:

```
make client
```

To start the server, use:

```
stats_svc &
```

and to run the client, type:

```
stats <machine_name>
```

where machine name is the name of the machine that the server is running on.

Next Time: RPC performance tips.

```
# Makefile:
CLIENTS = stats.c stats_clnt.c stats_xdr.c
SERVERS = stats_svc.c stats_proc.c stats_xdr.c

server :
      rpcgen stats.x
      $(CC) $(SERVERS) -o stats_svc -lm

client: stats
      $(CC) $(CLIENTS) -o stats
```

```
/*
 * FILE stats.x:
 * RPCL Protocal definition for a
 * statistics server.  PROCS include
 * SUM, MEAN, and STDDEV.
 */
const MAX_DATA = 256;

typedef double stat_data<MAX_DATA>;

program STATPROG {
      version STATVERS {
            double SUM(stat_data)        = 1;
            double MEAN(stat_data)       = 2;
            double STDDEV(stat_data)     = 3;
      } = 1;
} = 0x20000099;
```

```
/*
 * File stats_proc.c:
 * These are the statistics procedures that we
 * want.  They are relly quick, dirty, and ugly.
 * Note that with XDR data, we always deal with a
 * POINTER to the data we are working with, never
 * with the data itself.  So in some instances
 * (i.e. strings, or our list of doubles here)
 * you will be dealing with a pointer to a pointer.
 * keep this in mind, or you'll have no end of
 * core dumps to contend with.
 * in my experience, this is usually the most
 * common gotcha in rpc programming.
 */
#include "stats.h"
#include <math.h>

double *
sum_1(the_data)
stat_data   *the_data;
{
      static double     result = 0.0;
      int               ctr;
      double            *ptr;

      ptr = the_data->stat_data_val;
      for (ctr = 0; ctr < the_data->stat_data_len;
                                    ctr++)
            result += *(ptr++);
      return(&result);
}

double *
mean_1(the_data)
stat_data   *the_data;
{
      static double     result = 0.0;
      int               ctr;
      double            *ptr;

      ptr = the_data->stat_data_val;
      for (ctr = 0; ctr < the_data->stat_data_len;
                                    ctr++)
            result += *(ptr++);
      result /= (double) the_data->stat_data_len;
      return(&result);
}

double *
stddev_1(the_data)
stat_data   *the_data;
{
      static double     result = 0.0;
      int               ctr;
      double            avg, tmp, sum;
      double            *ptr;

      ptr = the_data->stat_data_val;
      avg = tmp = sum = 0;
      for (ctr = 0; ctr < the_data->stat_data_len;
                                    ctr++)
            avg += *(ptr++);
      avg /= (double) the_data->stat_data_len;
      ptr = the_data->stat_data_val;
      for (ctr = 0; ctr < the_data->stat_data_len;
                                ctr++) {
            tmp = *(ptr++) - avg;
            sum += tmp * tmp;
      }
      sum = sum /
            (double) (the_data->stat_data_len - 1);
      result = sqrt(sum);
      return(&result);
}
```

```
/*
 * File stats.c:
 * Statistical Client Program.
 *
 * This program malloc()'s an arry of doubles, and
 * statically initializes it with fixed numbers.
 * It then calls the RPC server stat_svc to obtain
 * the sum, mean, mode and standard deviation of
 * the data.
 */
#include <stdio.h>
#include <rpc/rpc.h>   /* needed by all RPC pgms */
#include "stats.h"      /* generated by rpcgen */

int
main(argc, argv)
int   argc;
char  *argv[];
{
      CLIENT            *cl;
      stat_data         my_data;
      double            *ptr;
      char              *server;
      double            *result;

      /*
       * Get name of server from command line
       */
      if (argc != 2) {
            fprintf("usage:: %s servername\n",
                  argv[0]);
            exit(1);
      } else {
            server = argv[1];
      }

      /*
       * create the client handle.  If this fails,
       * print the error and exit.
       */
      if (!(cl = clnt_create(server, STATPROG,
                        STATVERS, "udp"))) {
            clnt_pcreateerror(server);
            exit(1);
      }

      /*
       * Now that we have client handle created,
       * let's allocate our array of doubles.
       * Note that we have to know the internals of
       * the stat_data structure and set the _len
       * and _val members.
       */
      my_data.stat_data_len = 20; /* twenty #'s */
      if ((my_data.stat_data_val = (double *)
            malloc(20 * sizeof(double))) == NULL) {
            fprintf("%s: couldn't malloc array\n",
                  argv[0]);
            exit(1);
      }
      /* make a copy of the pointer */
      ptr = my_data.stat_data_val;

      /*
       * Initialize array with some static values
       * This is pretty gross, however, this is
       * just an example.  You'd probably want to
       * read these from stdin, a file, etc.
       */
      *(ptr++) = (double) 234.519;
      *(ptr++) = (double) 52.254;
      *(ptr++) = (double) 62.8736;
      *(ptr++) = (double) 186.73;
      *(ptr++) = (double) 91.736;
      *(ptr++) = (double) 132.85;
      *(ptr++) = (double) 29.483;
      *(ptr++) = (double) 76.723;
      *(ptr++) = (double) 77.5243;
      *(ptr++) = (double) 91.72;
      *(ptr++) = (double) 92.1;
      *(ptr++) = (double) 91.398;
      *(ptr++) = (double) 89.357;
      *(ptr++) = (double) 109.28;
      *(ptr++) = (double) 116.93;
      *(ptr++) = (double) 117.0;
      *(ptr++) = (double) 277.91;
      *(ptr++) = (double) 17.37;
      *(ptr++) = (double) 97.348;
      *(ptr++) = (double) 98.8;

      /* make our calls */
      result = sum_1(&my_data, cl);
      if (result != NULL)
            printf("sum is %lf\n", *result);
      else
            fprintf(stderr, "sum_1() failed\n");
      result = mean_1(&my_data, cl);
      if (result != NULL)
            printf("mean is %lf\n", *result);
      else
            fprintf(stderr, "mean_1() failed\n");
      result = stddev_1(&my_data, cl);
      if (result != NULL)
            printf("standard deviation is %lf\n",
                  *result);
      else
            fprintf(stderr, "stddev_1() failed\n");

      /* We're done!  Return success ! */
      return(0);
}
```

Agenda

for
Tuesday, April 14, 1992, 7:30 PM
Senate Chambers
245 Engineering Bldg.
University of Manitoba
Ft. Garry Campus

1. Round Table 7:30

2. Business Meeting 8:00
   a) President's Report
   b) Membership Secretary's Report
   c) Newsletter Editor's Report
   d) Treasurer's Report
   e) Meeting Coordinator's Report
   f) New Business

3. Break 8:20

5. Presented Topic 8:30
   Unix on the Mainframe - Amdahl's UTS
   Jeff Patterson, Amdahl Canada Ltd.
   The presentation will show how Unix is moving into the mainframe world and how this can affect organizations that have moved (or plan to move) to Open Systems/ Unix. The presentation will cover several interelated topics: Open Systems vs. Proprietary Systems - Today, Unix on the Mainframe, Open Systems Challenges, Open Systems vs. Proprietary Systems - The Future. While Amdahl realizes that most people don't think of a mainframe as a destination for Unix applications, there are clients, environments, applications and situations where it may be a viable alternative, and we wish to address that.

6. Adjourn 9:30

Note: Please try to arrive at the meeting between 7:15 and 7:30 pm. Thank You.

Volunteers Needed

We need about 6 to 8 people to help out with the group's booth at the MWCS Computer Fest on Sunday, April 26. Volunteers will get free admission to the Computer Fest. See page 2 for details.

TUUG Vote Results

Tuesday, March 10, 1992, 7:30pm
CIIT Building Conference Room
435 Ellice Ave.

Although we don't have minutes from last month's meeting, since there was no formal business meeting then, here are the results of the votes on the proposed name change, and the proposed affiliation with UniForum Canada.

Ballot sheets were sent to each TUUG member with the March newsletter. There were 63 paid up members at the time. A total of 26 ballots were returned, either by mail or at the March meeting.

On the question of changing the group's name to "Manitoba Unix User Group," 25 voted in favour, with one abstention. On the question of affiliation with UniForum Canada, 25 voted in favour, with one abstention.

In addition to the votes on these two motions, the ballot sheet also surveyed members to determine who would be interested in signing up for an individual membership in UniForum Canada. (There was no obligation to join by answering "yes" to this survey question.). Of the 26 respondents, 14 indicated an interest in UniForum Canada membership.

Thanks to all members who took the time to respond. Based on the results of the vote, both motions are carried. We will now proceed with a name search and name notation for the new group name (until that point, we will continue to use the name TUUG). We will also now proceed with our discussions with UniForum Canada regarding affiliation. You will be informed of these changes as they become official.

Next Month

Meeting:
Our May meeting is scheduled for Tuesday, May 12, at 7:30 PM. The presented topic is to be announced. The meeting will start off with our usual round table and business meeting. Location is also TBA.

Newsletter:
We will likely continue with our Q&A column, and RPC Programming by Scott Balneaves, next month. I also have an article on shared memory by Peter Graham, and several "filler" articles by Roland Schneider. I also hope to have a write-up on DEC's new 64-bit Alpha architecture, and specifications on it's first microprocessor based on this technology.