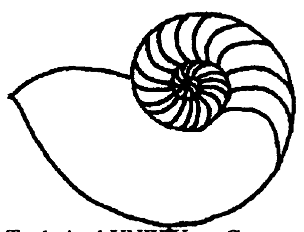
February 1990 $2.50
Volume 2, Number 5



Mr. Gilles Detillieux
Consultant
Info West Inc.
P.O. Box 130
St.Boniface MB R2H 3B4

newsletter of the Technical UNIX® User Group

This month ...

Late Breaking News...
Next Meeting to be held at
UNISYS
See ANNOUNCEMENT for details

Thoughts From The Editor

By Susan Zuk

Happy Valentine's Day!!! Are you doing something special for your Valentine on February 14th?

Last month we had our meeting at the University of Manitoba. Pat Macdonald spoke to us about the internals of NFS. The presentation was interesting and gave us an insight as to how this product works. Thank you Pat for your time and helpful presentation.

This month's newsletter is an assortment of information. Gilbert first discus[illegible]iences with UNIX communications in the [illegible]lks of how intricate communicati[illegible] with many users wanting to access their files lo[illegible]erous machines. When working with the large population at the University of Manitoba you can see how this can be a real challenge!

Our featured article is on the diff command. The diff command is used to show the differences between files. The authors give good examples on how the command works and on how to use various options. The nice part about this article is it also defines other commands with similar uses but with different options. Take a look and see how you might be able to use these commands in your environment.

We also have a special invitation from the Muddy Water Computer Society (MWCS). This group is for general computer users. The platforms dealt with by this group are mainly microcomputers. The idea of the group is people helping one another with their computer problems. MWCS is offering us the use of their bulletin board for a trial period. Derek Hay, is the President of MWCS and is also our Vice President. If you have any questions you can give him a call.

That's all I have to say for now. Take care and join us at our next meeting.

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 1 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.

The Executive

President:	Gilbe[illegible]llieux	261-9146
Vice President:	Derek Hay	943-5401
Treasurer:	Gilles Detillieux	261-9146
Secretary:	Matt Binnie	(W) 949-0190
Newsletter Editor:	Susan Zuk	(W) 788-7312
Membership Sec.:	Gilles Detillieux	(W) 788-6209
Information:	Gilbert Detillieux	261-9146
	(or) Susan Zuk	(W) 788-7312

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

Copyright Policy and Disclaimer

ANNOUNCEMENT...

Meeting Location:
The February meeting location will be provided by UNISYS, 300-1661 Portage Avenue. Upon entering the building you will then be required to sign-in. Please sign-in using "TUUG" as the agency represented.

President's Corner

by Gilbert Detillieux, President

One of the most complicated, interesting, and frustrating aspects of system administration on UNIX is the whole area of network services. This point became strikingly clear to me as I struggled to set up a network of UNIX workstations in a very short period of time recently.

As we head into the 1990's, networked UNIX systems will become the standard. Network interfaces and TCP/IP software are now available for virtually every UNIX system worth mentioning, and even come as standard equipment on many systems (for example, UNIX-based graphics workstations). Distributed computing has now become an affordable and practical alternative to a centralized computer system - each user can have a powerful desktop computer, can work without having performance too greatly affected by other users, and can still share centralized resources (files, printers, etc.).

The networking software on UNIX has been quite well thought out as far as these user services are concerned. However, little (or at least not enough) thought has been given to its impact on the system administrator. Distributed computing has lead to distributed control and authority, making it more difficult to maintain the integrity and security of all systems. Events such as the Internet worm attack make this clear, in a very alarming way - we can try to keep our systems as secure as we can, but as long as they are connected to the outside world, they are vulnerable to attack.

A less alarming, but nonetheless difficult, network administration problem that we are facing at the U of M is the whole issue of shared file volumes under NFS. Users from several departments are now or will eventually be using computers from other departments, and will want to have access to all their usual files from any machine. Since file access permission is granted based on numeric user and group IDs, these must now be standardized across campus. Software such as the Yellow Pages services provide a partial solution, since they allow a centrally administered password file to be shared among machines. However, this may not be practical in environments where administration is not necessarily centralized, and where separate password files should still be maintained.

Solutions to system administration problems created by networking will come, but it will take a fair bit of effort, and in many cases, some pretty creative thinking. It will be interesting to see if the same creative minds that gave us UNIX based networking services will now be able to solve some of the problems that have thus been created. In any case, such problems make great fodder for discussion in user groups such as TUUG.

Our January meeting topic (NFS internals) was one of the first to deal with networking issues, and it certainly won't be the last. In fact, future meetings will deal with TCP/IP services, X-Windows and Graphical User Interfaces, and other network related topics. The topic for February will likely be TCP/IP services from a user perspective. (For example, we would look at the use of Telnet, FTP, rlogin, rsh, rcp, etc.) If time permits, we will also look at how TCP/IP is set up. For this meeting, we will be back at Unisys, at the usual time (Tuesday, February 13 at 7:30PM).

I hope to see you all at the meeting. "Bon Festival!"

****

The Fortune File

Submitted by Kirk Marat

Found appended to an article in the comp.sys.nsc.32k newsgroup:

There is no problem, no matter how large or small, that cannot be solved by the appropriate application of high explosives.

How-to UNIX: Settling Your Differences The Multifaceted diff Command

By Tim O'Reilly & Dale Dougherty

Reprinted with permission from the March/April 1988 issue of CommUNIXations, published by Uniforum

With computers, it's easy to save multiple versions of the same document. What's not so easy is figuring out which one is which a few months down the line. Fortunately, to help to solve the many problems that can occur in sorting out the differences between substantially similar files, UNIX includes the *diff* family of programs. The *diff* command compares two files and prints the differences between them. *diff* uses ed-like notations (a for append, c for change and d for delete) to describe how a set of lines has changed from one file to the next. This is followed by the differing lines themselves. The < character precedes lines from the first file and > precedes lines from the second file.

To explain the output produced by *diff*, I'd like to create a simple example. Consider the two paragraphs shown in Figure 1a.

When you run *diff* on these files, the output shown in figure 1b is produced.

To understand the report, remember that *diff* is prescriptive, describing the changes needed to make the first file the same as the second file. This report specifies that the first and second lines in the first file would have to be replaced by the first line in the second file; and that the sixth line in the first file would have to be replaced with the fifth line in the second file.

Few people look at the *diff* line change notation because it is easy enough just to follow the directional arrows to see what is found in one file or another.

However, what if you want to merge the two versions of the file? In this case, the editing notation becomes important. *diff* has an option that will actually create an editing script that you can apply (with either *ed* or *ex*) to recreate file1 from file2 (the first and second files specified on the *diff* command line).

Obviously, there is no need to completely recreate the first file from the second because you could do that easily with *cp*. However, by editing the script produced by *diff*, you can come up with some desired combination of the two versions.

As you can see in Figure 1b, there is one additional phrase in each of the two files. What if you would like to merge them into one file that incorporates both edits? Try typing:

```
$ diff -e kells1 kells2 > exscript
```

This command will place the output shown in Figure 2a into the file exscript.

You'll notice that the script appears in reverse order, with the changes later in the file appearing first. This is essential whenever you're making changes based on line numbers; otherwise, changes made earlier in the file may change the numbering, rendering the later parts of the scripts ineffective.

You'll also notice that, as mentioned, this script will simply recreate version 1, which is not what you want. Rather, you

Figure 1a - Version 1

```
The Book of Kells, now one of the treasures of the Trinity
College Library in Dublin, was found in the ancient
monastery at Ceannanus Mor, now called Kells. It is a beauti-
fully illustrated manuscript of the Latin Gospels,
and also contains notes on local history.
It was written in the eigth century.
The manuscript is generally regarded as the finest example
of Celtic illumination.
```

Version 2

```
The Book of Kells was found in the ancient
monastery at Ceannanus Mor, now called Kells. It is a beauti-
fully illustrated manuscript of the Latin Gospels,
and also contains notes on local history.
It is believed to have been written in the eigth century.
The manuscript is generally regarded as the finest example
of Celtic illumination.
```

Figure 1b

```
$ diff kells1 kells2
1,2c1
< The Book of Kells, now one of the treasures of the Trinity
<College Library in Dublin, was found in the ancient
---
>The Book of Kells was found in the ancient
6c5
< It was written in the eighth century.
---
> It is believed to have been written in the eighth century.
```

want the change to line 5, but not the change to lines 1 and 2. To accomplish this, you want to edit the script so that it looks like Figure 2b.

Notice that you had to add the *w* command to write the results of the edit back into the file. Now you can type:

$ ex - kells1 < exscript

This will produce the resulting merged file shown in Figure 2c.

Using *diff* like this can get confusing, especially when there are many changes. It is very easy to get the direction of changes confused or to make the wrong edits. Just remember to do the following:

* Specify the file that is closest in content to your eventual target as the first file on the diff command line. This will minimize the size of the editing script that is produced.

* After you have corrected the editing script so that it makes only the changes that you want, apply it to that same file (the first file).

Nonetheless, because there is so much room for error, it is better not to have your script write the changes back directly into one of your source files. Instead of adding a *w* command at the end of the script, add the command 1,$p to write the results to standard output. This is almost always preferable when you are using complex editing script.

If you use this command in the editing script, the command line to actually make the edits would look like this:

$ ex - kells1 < exscript > kells3

Wishing for an Earlier Version

The *diff* manual page also points out another application of this feature of the program. Often, you find yourself making extensive changes, and then wishing you could go back and recover some part of an earlier version. Obviously, frequent backups will help. However, if backup storage space is at a premium, it is possible (though a little awkward) to save only an older version of a file and then keep incremental *diff -e* scripts to mark the differences between each successive version.

To apply multiple scripts to a single file, you can simply pipe them to *ex* rather than redirecting input:

cat script1 script2 script3 | ex - oldfile

But, how do you get your *w* (or 1,$p) command into the pipeline? Although you could edit the last script to include one of these commands, there is one more trick worth looking at because it illustrates another useful feature of the shell (one that many people are unaware of).

If you enclose a semicolon-separated list of commands in parentheses, the standard output of all commands are combined and can be redirected together. The immediate application is that, if you type:

cat script1 script2 script3; echo '1,$p' | ex - oldfile > newfile

the results of the *cat* command will be sent, as usual, to standard output, with only the results of echo piped to *ex*. However, if you type:

(cat script1 script2 script3; echo '1,$p') | ex - oldfile > newfile

the output of the entire sequence will make it into the pipeline, which is what you want.

Using Side-By-Side *diff*

If you prefer to make changes interactively, another program called *sdiff* (side-by-side diff) may be more appropriate. Its most straight forward use is to display two files in two columns

Figure 2a

```
6c
It is believed to have been written in the eighth century.
.
1,2c
The Book of Kells was found in the ancient
.
```

Figure 2b

```
6c
It is believed to have been written in the eighth century.
.
w
```

Figure 2c

```
The Book of Kells, now one of the treasures of the Trinity
College Library in Dublin, was found in the ancient
monastery at Ceannanus Mor, now called Kells. It is a beauti-
fully illustrated manuscript of the Latin Gospels,
and also contains notes on local history.
It is believed to have been written in the eigth century.
The manuscript is generally regarded as the finest example
of Celtic illumination.
```

on the screen. In a gutter between the two columns, the program displays a < if the line is unique to the first file, a > if the line is unique to the second file, and a | if the line is different in both files. Because the default line length of this program (130 characters) is too wide for most terminals, it is best to use -*w* option to specify a smaller width and truncate the display.

Figure 3 shows the results of running *sdiff* on the same two files looked at with *diff* above. Notice that *sdiff* shows every line in each file. You can use the -s option to make *sdiff* suppress identical lines and only show differences between the two files.

One of the most powerful uses of *sdiff* is to build an output file by interactively choosing between different versions of two files. To do this, you have to specify the -o option and the name of an output file to be created. The *sdiff* command then displays a % prompt after each set of differences. You can compare the different versions and select the one that will be sent to the output file. As shown in Figure 4, some of the possible responses are l to choose the left column, r to choose the right column and q to exit the program.

The file kells3 (again, referring to Figure 4) will contain all of the lines that both files had in common, plus lines one and two from kells1 (chosen with the l command).

Comparing Files, Practically

Having looked at these commands in simplified examples, now is a good time for some practical applications for comparing files. When working on a document, it is not an uncommon practice to make a copy of a file and edit the copy rather than the original. This might be done by a writer, for example, if someone other than himself were entering edits from a copy marked up by hand. The writer could use *diff* to compare the original with the revision and so examine the changes without reading the entire document.

As another example, suppose that after distributing program source code to a large number of different sites, a company (or individual) then makes a number of minor bug fixes to several of the modules. Rather than distributing the source again in its entirety, it is possible to simply distribute a set of *diff*-e scripts, or patches. Not only are these patches easier to distribute, they also protect against the possibility that the recipient of the original files has already modified them himself, and would not want to replace them altogether. As a matter of fact, patches to public domain sources are distributed in this fashion every day over Usenet.

Figure 3 - Running sdiff

```
$ sdiff -w80 kells1 kells2
The Book of Kells, now one of the    |   The Book of Kells was found in the
College Library in Dublin, was fou   <
monastery at Ceannanus Mor, now ca       monastery at Ceannanus Mor, now ca
beautifully illustrated manuscript       beautifully illustrated manuscript
and also contains notes on local h       and also contains notes on local h
It was written in the eighth centu   |   It is believed to have been writte
The manuscript is generally regard       The manuscript is generally regard
of Celtic illumination.                  of Celtic illumination.
```

Figure 4 - sdiff Plus Options

```
$ sdiff -s -w80 -o kells3 kells1 kells2
1,2c1
The Book of Kells, now one of the    |   The Book of Kells was found in the
College Library in Dublin, was fou   <
%r
6c5
It was written in the eighth centu   |   It is believed to have been writte
%l
```

The SCCS Approach

A still broader application, which was hinted at earlier, is to use *diff* to keep compact backups of a series of successive versions of a given document. As it turns out, this is precisely the function of the Source Code Control System or SCCS. SCCS is a facility for keeping track of the changes to files that take place at different stages of a software development or documentation project.

Suppose you have a first draft of a program or a manual. (This is referred to as a delta when it is saved in a special SCCS format). The second draft is, of course, based on changes to the first draft. When you make the delta for the second draft, SCCS uses *diff* to record the changes to the first draft that resulted in the second draft instead of keeping a separate copy for each draft. Only the changes, and the instructions for having an editor make them, need to be maintained. SCCS allows you to regenerate earlier drafts, which saves disk space.

SCCS is quite complex (too complex to describe here), but we recommend that you investigate it if you are working on a large, frequently revised or multiple-author programming or writing project.

Two More...

Two other programs worth mentioning are *diff3*, which allows you to compare three files at once, and *bdiff* (big file diff). *bdiff* is used on files too large for *diff*. It breaks up a large file into smaller segments and then passes each one through *diff*, and maintains line numbering as though *diff* were still operating on the single large file.

Tim O'Reilly is president of O'Reilly & Associates, Inc., a consulting firm that writes manuals and training materials.

Dale Dougherty and O'Reilly are co-authors of the Howard Sams book, UNIX Text Processing.

This article is adapted from UNIX Text Processing by Dale Dougherty and Tim O'Reilly (Howard Sams, 1987.) As a result, the discussion is slightly biased toward text processing rather than programming.

The Muddy Water Computer Society Invitation

By The Executive, Muddy Water Computer Society
Submitted by Derek Hay

The executive of the Muddy Water Computer Society wishes to extend to the Technical UNIX User Group our invitation to examine and explore our Bulletin Board System.

Muddy Water Computer Society, knowing that your group does not currently have a bulletin board, and thus has no way to effectively transfer files or messages to all users as a whole, (uucp sends only to people specified), offers a trial use for about one month, of it's facilities.

All current UNIX user group members have been logged into our system and have been assigned a password of UNIX. All you have to do is call on a 300-2400 baud modem (MNP5 available) 943-6508. 8 bit,no parity,1 stop bit. Please answer all questions at the prompts. Help can be achieved at almost any point by selecting the option that is displayed on the prompt line. After login please feel free to change your password using menu option W(rite) user data.

All UNIX members will be allowed full access to the Main board section, Unix section and the Oracle Section for one hour per 24 hour day. Please note that the Unix and Oracle sections are quite new and therefore are still quite small. We hope that through trial periods and new members we will be able to improve on the amount of files that will be available. Please upload and download any files you may wish. If a file is not found on the system but is in the file table with a description, it may be in our off-line library, and will have to requested to be placed back on the system.

Should any of the UNIX members still like to have access to our BBS after the trial period expires (approx one month) arrangements may be made for a cost reduction, or merger of services between the two groups (this will have to be an executive decision). In the mean time please enjoy the services available.

Any comments that you may have on this arrangement may be left in the UNIX conference (# 5) either as private mail or public.

The Executive
Muddy Water Computer Society
Winnipeg, Mb
943-6508 (2400 8-n-1)

Technical UNIX® User Group

Agenda

for

Tuesday, February 13, 1990

7:30pm

UNISYS

UNISYS Building

300-1661 Portage Avenue

1. Round Table	7:30
2. Business Meeting	8:00
a) Membership Secretary's Report	
b) Newsletter Report	
c) Treasurer's Report	
4. Break	8:30
5. Presented Topic	8:40
TCP/IP	
6. Adjourn	9:30