Vol. 16 No. 1

September 2003

$2.50

MUUGlines

The Manitoba UNIX User Group Newsletter

September 9, 2003: Regular Expressions

In our September meeting Steve Moffat will be talking about Regular Expressions. Some say the "Generalized Regular Expression Parser", or grep, may be the most frequently used command in Unix. Using this simple command more effectively can save you time and give you better results every day. Whether it is grep, egrep, fgrep, perl, or any of the other programs that use or support text pattern matching, you can use Unix better if you know what Regular Expressions can do for you.

Meetings are held at the IBM offices at 400 Ellice Ave. (between Edmonton and Kennedy). When you arrive, you will have to sign in at the reception desk, and then wait for someone to take you (in groups) to the meeting room. Please try to arrive by about 7:15 PM, so the meeting can start promptly at 7:30 PM. Don't be late, or you may not get in.

Limited parking is available for free on the street, or in a lot across Ellice from IBM, for $1.00 for the evening. Indoor parking is also available nearby, at Portage Place, for $2.00 for the evening.

Call For Nominations

Hear ye, hear ye!

This is a call for nominees to participate in the election for the MUUG board. Those elected will serve from October 2003 until October 2004. The deadline to nominate yourself, or someone else, is September 23, 2003. Instructions follow.

The MUUG board is charged with coordinating the meetings and other events by the group. It's fun, and you get a role in guiding the group. All are encouraged to apply.

———————————————-

MUUG Board Elections - Call for Nominations

Every October the Manitoba Unix User Group holds its Annual Meeting, the main goals of which are to elect a new Board of Directors and to pass any special resolutions. (Aside from that, it is a regular meeting.) Any member in good standing (i.e. a paid-up member) can be nominated to run for a position.

As of this writing, the following members of the current Board have let their names stand for re-election: Gilbert Detillieux, Kevin McGregor, Doug Shewfelt and Adam Thompson.

Get Involved

To put your name on the list, do the following:

1. Ensure your membership is in good standing (i.e. do you have to renew?)
2. Find another member willing to second your nomination
3. Email your nomination to <election@muug.mb.ca>, with a CC to your seconder
4. Get your seconder to send an email to <election@muug.mb.ca> indicating their support for your nomination

In lieu of email, written correspondence will also be accepted (c/o MUUG Election Committee, P.O. Box 130, St-Boniface, Winnipeg, MB, R2H 3B4). It is important to include the following information in your nomination: Name, Title, Employer/Occupation, short (100 word) biography. Without this information, we can't put together the list of nominees to give to the rest of the members, can we?

In the event the number of nominees is fewer than the number of vacant positions, all will be accepted by acclamation. If there are more, decision will be made by secret ballot at the Annual Meeting. Nominees should familiarize themselves with the MUUG bylaws, found at http://www.muug.mb.ca/pub/bylaws/. Any questions about the election or the nomination process can be directed to Sean Walberg at the election@muub.mb.ca mailbox, or by phone at 975-5987 during business hours. Snail mail and fax information can be found at http://www.muug.mb.ca/about.html.

Again, the deadline for nominations is September 23, 2003, with the election being held at the October 7th meeting. Good luck to all!

Saving space with Screen

By Sean Walberg

If you're like me, you've got more than a few shells going on at any given time. Not a problem when you've got two 21" monitors, but it can get a bit cramped if you're remotely logged in via SSH, or, even more painful, a serial console. "screen" is a program that lets you multiplex several terminals into one, and is the topic of this article.

Stepping back a bit, when you are typing into a shell (or a program you started from the shell, such as pine), you're communicating with a Virtual Terminal, or VTY. When you log in, you're assigned a VTY, which gives you and your shell a way of passing characters to and from the rest of the system. The problem is, you can generally only run one thing at a time per shell. If you need to, you log in a second time, or, if you're even smarter, you get pretty handy with sending processes to the background. Screen's job is to create new VTYs for you, and let you switch between them from the comfort of your original VTY. Think of it like Mozilla's tabbed windows — you have one physical window, but several web pages you can cycle through.

Invoking screen is simple, just type "screen". Nothing special should happen, the screen will clear and you'll be back at the command prompt. If you're typing from within an X-Terminal, you might notice the title has changed to something like "[screen 0] tcsh", indicating that you're in window #0, and the command was tcsh (my login shell).

Commands in screen are, by default, prefixed with Control-A, the short form the man page uses is C-a. Thus, "C-a ?" means to press Control and A, followed by the question mark. Go ahead, do it now, you'll be taken to the help screen. I'll guide you through the most common features, but always remember how to get to the help screen.

Now that we're within a window (that's what the man page calls 'em, works for me), it's time to start creating more windows. A new shell can be started with "C-a c". This is window 1. To get back to the first window, try "C-a 0". Sending C-a followed by a number gets you to a specific window. To cycle through the windows one by one, the command is "C-a space", or "C-a n" (think "next"). To cycle backward, it's "C-a p" (previous).

Yet another way to start a window is to type "screen" followed by the command, from within a window. Thus, "screen pine" opens up a new window and runs pine. "C-a c" is then, basically, the same as typing "screen /bin/tcsh", or whatever your login shell is.

Whenever you exit the shell or program that started up the window, that window is closed. If you're really angry at a window and can't close it, "C-a K" (kill) will close it abruptly for you. When you exit your final window, screen exits with a small message.

Since screen manages several VTYs on your behalf, we can background the screen process and pick it up later, even from a different location. While screen is running, detach it with "C-a d". You'll be notified with a message, and returned back to your original login shell. To pick up again, run screen with the -R option (resume), and you're back. "screen -R" is a good way of running screen in the first place, as it'll pick up a detached screen session if it's there, or if not, simply start up a new one. If you decide to put screen in your startup scripts, this is a good way of running it.

Those are the basics of screen. There are a lot of fancy things that can be done, such as scrolling back (C-a ESC), splitting your screen into multiple screens (C-a S), switching between them (C-a C-I), and even cutting and pasting. The man page lists all the fun stuff you can do.

Screen is a versatile program that saves space on your display, not to mention gives you some abilities normally found only in graphical terminals. Throw it in your login script, and make sure it's always there when you need it.

Cleaning FTP

By Sean Walberg

What separates the men from the boys, at least to this crusty old Unix admin, is the ability to automate repetitive tasks. Nothing annoys me more than having to run the same commands day after day, especially when there is no thought going into it.

Script kiddies have this down to an art form. If you run a publicly accessible FTP server that allows uploads, you've probably noticed odd files magically appearing. If you've set up your server correctly, you should only see a few files. If not, you could find yourself short of hard drive space, but that's another article. The files being uploaded are for various reasons, namely to see if your site is a viable warez server, and how fast the connection is.

Either way, I'm not keen on keeping these files around, so I delete them.

If the kiddies can automate these file transfers, then I should be able to automate their deletion. Since there is valid data that could be in my incoming directory, a simple "rm -rf *" from cron won't do it, so we'll be a bit smarter.

Looking in my incoming directory, I see the following:

```
[root@poochie incoming]# ls
030727220658p 030808013101p foo.tar.gz
test
030728235109p 1mbtest.ptf space.asp
```

The directories starting with "03" aren't mine, nor are "1mbtest.ptf" and "space.asp". By hand, I'd run

```
# rm -rf 030727220658p
...
# rm space.asp
```

to get rid of them, which obviously sucks. Of course, those directories look like timestamps, so I can't count on them being the same. A shell script, and some regular expression magic will help me here.

In the top of any shell script, you must define the interpreter that is to run it. Following that is generally any user changeable parameters:

```
#!/bin/bash

BASE=/export/home2/ftp/incoming
BAD="1mbtest.ptf space.asp"
```

Here I'm running my script under bash, and I'm defining the incoming directory and some static "bad" filenames.

I should really make sure the user has enough permissions to carry on, otherwise it's just wasting everyone's time:

```
if [ ! -w $BASE ]; then
echo Must be able to write to $BASE
exit
fi
```

Here, I check to see if the base directory is writable (-w). If not (the "!"), spit out a message and quit.

I still have to build a list of stuff to delete. Even though I know the names of the files, those pesky directories are still around. The general format seems to be 12 numbers followed by a letter, which sounds like a job for regular expressions:

```
cd $BASE
DIRS='find . -type d -iregex '.*/[0-
9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9][0-9][0-9][a-z]' '
```

That's a mouthful!

First, change into the directory (you'll see why later).

The next line builds the list of directories into the $DIRS variable. In shell scripting, any time you assign something in backticks to a variable, the "something" is run by the shell, and the results assigned to the variable. For example,

```
DATE='/bin/date'
echo $DATE
```

In my command above, I'm using the `find` command to produce a list of the script kiddie's directories. I start the find at the current directory, am only looking for directories (-type d), and am passing a case insensitive regular expression to match files.

The regular expression I'm using is long but simple. I start off by matching any path (.*/), followed by 12 digits and a letter. Any time you see a regexp in square brackets, it means "anything in this set". [0-9] thus means "any digit", and [a-z] is "any letter". Had this been done in perl, ".*\/ \d{12}\w" would have sufficed, but the regular expression parser in `find` isn't as robust as perl's.

Whenever developing a shell script that involves backticks, it's helpful to have another terminal open to test the stuff in the backticks. For example, I developed the regular expression from the shell, then copied it into my shell script. It's a time saver, and leads to better code.

At this point, the list of directories to delete is in $DIRS, and the list of files to delete is in $BAD. It's a simple matter to loop through now, and delete what's not needed:

```
for i in $BAD $DIRS; do
if [ -d $BASE/$i ]; then
rm -rf $BASE/$i
fi
if [ -e $BASE/$i ]; then
rm -f $BASE/$i
fi
done
```

This loop goes through all files in both $BAD and $DIRS. If it's a directory (-d), then "rm -rf" is run on the directory. If that fails, a simple “does the file exist?” test is run (-e), and if so, the file is deleted.

Another handy tip for developing shell scripts is to ensure your script behaves properly before you let it loose. For example, I’m running some `rm` commands from an automated script. Before I set it up in cron, I added an “echo” before the “rm”, so that I had

```
echo rm -rf $BASE/$i
and
echo rm -f $BASE/$i
```

When I ran the script, it only printed off what it was going to do before it did it. Then I could verify the output to make sure it wasn’t going to destroy my filesystem overnight. Once that was done, I removed the echos, and the script was live.

So that was a look at a simple maintenance script. I threw it into my nightly maintenance tasks, and keep my incoming directory relatively clean. One more thing I can stop wasting time with.

Here’s the full script:

```
#!/bin/bash
BASE=/export/home2/ftp/incoming
BAD=”1mbtest.ptf space.asp”

if [ ! -w $BASE ]; then
echo Must be able to write to $BASE
exit
fi

cd $BASE
DIRS=‘find . -type d -iregex ‘.*/[0-
9][0-9][0-9][0-9][0-9][0-9][0-9][0-
9][0-9][0-9][0-9][0-9][a-z]’ ‘

for i in $BAD $DIRS; do
if [ -d $BASE/$i ]; then
rm -rf $BASE/$i
fi
if [ -e $BASE/$i ]; then
rm -f $BASE/$i
fi
done
```

Firebird Moves To Replace Mozilla

by Mozillazine

Mozilla Firebird has grown from its modest beginnings as an offshoot of the mainstream Mozilla project to become the centre of the Mozilla Foundation’s future strategy. In the past, development has sometimes been tumultuous: weeks of furious activity have often been followed by periods of almost no change at all and at several points the project has come close to death.

Fortunately, Firebird development has been rapid in recent weeks as the program makes the last remaining changes necessary for it to become the default Mozilla browser, which is likely to occur in the 1.6 timeframe.

A Mozilla Application Suite feature that has recently been reimplemented in Firebird is a selector for alternative stylesheets. When a page specifies one or more preferred or alternative stylesheets, Firebird displays an icon on the Status Bar. Clicking on this icon brings up a list of the preferred and alternative stylesheets, as well as a ‘Basic Theme’ item, which applies only persistent stylesheets, and a ‘No Theme’ option, which causes all external stylesheets and embedded stylesheets to be ignored.

The Options dialogue Fonts & Colors and Connection panels have been removed (their settings can now be accessed via buttons in the Web Features and General panels respectively), leaving space for Downloads and Advanced. The latter panel supplies a UI for several features that have never had one before, allowing users to configure tabbed browsing behaviour and enable/disable functionality such as smooth scrolling and automatic image resizing.

Sending Us E-Mail?

Due to the amount of e-mail MUUG receives, we’ve set up an auto-reply to give you immediate feedback, and redirect some of the e-mail to the appropriate places. Why not look at http://www.muug.mb.ca/about.html#contacts first?

Share Your Thoughts

E-mail us with your comments on the newsletter, whether it’s criticisms or commendations, and continue to send in articles or ideas for same.

If you have a How-To or other idea, and aren’t ready to give a presentation at MUUG, an article is a great alternative! If you can write better than the editor, that’s terrific; if you can’t, submit it anyway and we’ll get it into shape for publication. We know that many of you have some great ideas and lots of knowledge.

Why not share? Mailto: editor@muug.mb.ca.