June 1989 $2.50
Volume 1, Number 9

Technical UNIX® User Group

newsletter of the
Technical UNIX® User Group

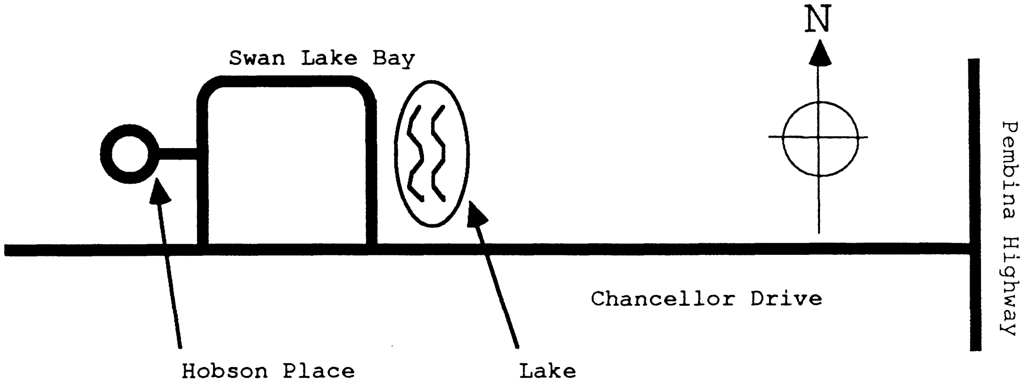
This month ...

Late Breaking News...
Next Meeting to be a Bar-B-Q
See inside for details

TUUG Barbeque Details

- Bring your own meat
- Bring your own beverage
- Bring your own lawn chair

- 22 Hobson Place
- Tuesday, June 13 - 6:30 pm



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:	Gilbert Detillieux		261-9146
Vice President:	Vacant		
Treasurer:	Gilles Detillieux		261-9146
Secretary:	Susan Zuk	(W)	786-8483
Newsletter Editor:	Darren Besler	(H)	254-3392
Membership Sec.:	Pat Macdonald	(W)	474-9870
Information:	Gilbert Detillieux		261-9146
	(or) Susan Zuk	(W)	786-8483

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

Copyright Policy and Disclaimer

This newsletter is ©opyrighted by the Technical UNIX User Group. Articles may be reprinted without permission as long as the original author and the Technical UNIX User Group are given credit.

The Technical 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.

ANNOUNCEMENT...

Meeting Location Change:

The June meeting will be a friendly get together and barbeque hosted by Kirt Marat at 22 Hobson Place, Winnipeg, on June 13. See map above for directions to get there. RSVP with Susan or Gilbert before June 9.

President's Corner

by Gilbert Detillieux, President

June is here already. It's seems it was just recently that the founding members of TUUG sat down and decided "let's get this group more organized," then held elections for the executive and started producing this monthly newsletter. But here we are already, near the end of the first year of operation of the new-and-improved TUUG! The group is now stronger than ever, and things look good for the second year. This calls for a celebration!

So, to celebrate this event, and to welcome the advent of our brief but great Winnipeg summer, we are having a barbecue in lieu of the June meeting. One of the members, Kirk Marat, has kindly offered us the use of his yard (and his basement rec-room, in case of bad weather). The location is found elsewhere in the newsletter; check the front page and the editorial page. The barbecue is on June 13, starting at about 6:30 PM. There will be no business meeting, I promise. Just good food, good company, and good times to be had by all!

You are welcome to bring along spouses, kids, and/or friends. Please call Susan Zuk or myself (phone numbers listed on editorial page) by June 9, at the latest, to let us know if you plan to attend, and how many will be coming. Everyone should bring their own favorite meat for the barbecue, as well as their favorite liquid refreshments. If you have lawn chairs, please bring them along; bring extras if you can.

We will be supplying a salad, potato salad, and a cake for desert. If you would like to bring something else for everyone to share, you are welcome; maybe let us know so we can coordinate things (and not wind up with twenty of the same thing). This will be the last time we meet before September; I hope to see you all out there. If you can't make it for supper, you are welcome to drop by later.

Now on to the next year... The October elections for the executive may seem a long way off, but there will only be one meeting before that. Nominations will be made at the September meeting, and will be open until the deadline for the October newsletter (so we can publish the list of candidates).

As a member of the executive for the past year, I can tell you how much I have enjoyed this. It has been a little hectic at times, but all-in-all, it hasn't required that much of my time, and has been the most rewarding part of my involvement with this group. Maybe some of you newer members would consider being a part of the executive; it's a good way to get to know more about the group and its members. We could sure use someone as vice-president, and someone to help out with planning future meetings. It would also be great to see more than one nomination for some of the positions, so we could have real elections, and some healthy competition! I'll leave you to think about it over the summer.

Hope to see you at the barbecue (June 13, RSVP by June 9). This past year has been a lot of fun, and I think our second year will be even better. A very warm and enjoyable summertime to all! We'll be back with another issue of the newsletter in September.

The fortune file

This month's fortune comes care of Darren Besler.

```
Although we modern persons tend to take our electric lights, radios, mixers, etc., for
granted, hundreds of years ago people did not have any of these things, which is just as
well because there was no place to plug them in.  Then along came the first Electrical
Pioneer, Benjamin Franklin, who flew a kite in a lighting storm and received a serious
electrical shock. This proved that lighting was powered by the same force as carpets, but
it also damaged Franklin's brain so severely that he started speaking only in
incomprehensible maxims, such as "A penny saved is a penny earned." Eventually he had to
be given a job running the post office.

                                        - Dave Barry, "What is Electricity?"
```

Initializing Serial Ports, Part I - Initializing Login Ports

by Gilbert Detillieux, Info West Inc.

In the February issue of this newsletter, we looked at how the UNIX system is initialized, and the role of the /etc/init program, and the /etc/inittab file. As part of that article, we saw that /etc/init initializes serial ports for logins by "respawning" a new process for each port, and running a program called /etc/getty. This month, we will take a closer look at this program.

The /etc/getty program is usually not called directly by a user, but rather is called by /etc/init, to allow logins on a serial port, by including, in /etc/inittab, a line such as follows:

```
04:2:respawn:/etc/getty tty04 9600
```

The two main duties of this program are: 1) wait for the port to become active, then determine communication parameters such as baud rate, parity, and character I/O mappings; and 2) call the login program to allow the user to log in. The second step is quite straightforward, and will not be addressed here. The first is a little more involved, and is the focus of this article. Incidentally, the name "getty" is an abbreviation for "get tty," and refers to the first step where /etc/getty gets the communication parameters for the serial port (tty).

The first argument to /etc/getty is the name of the device special file in /dev to be opened. When this device is opened, the process will usually be blocked until the port it accesses becomes active, i.e. until the terminal is powered up and placed on line, or the modem answers a call and gets a carrier. The second argument, which may look like a baud rate specification, but is actually an arbitrary string, is used as a key to look up a specific entry in the file /etc/gettydefs. Entries in this file have the following format:

```
label# initial mode # final mode #prompt#next
```

White space is allowed in the initial mode and final mode fields, and these can even be split onto more than one line. Each entry must be followed by an entirely blank line, to terminate it and separate it from the other entries. Listing 1 gives an example of an /etc/gettydefs file; this is the one that is in use on a Masscomp computer that I use.

The label field is used as a key for looking up a particular entry. Labels are arbitrary strings of letter and/or digits, but are often chosen to have some mnemonic value, such as specifying the entry's baud rate.

The initial mode field contains the initial communication parameter settings, to be used as soon as the device is opened. This is a list of words, separated by white space, that specify tty options that are to be enabled. These options are the same as those used by the stty(1) command, except they are in upper case, and the baud rate specifiers are preceded by the letter B. Not all desired options need to be explicitly specified, as many are assumed by default; the specified options are usually there to override the default action.

Once the device is set according to the initial mode, /etc/getty issues the login prompt, as specified in the prompt field. In this field, white space and newlines are significant, and are output as specified. The program then waits for input from the device.

If input is successfully read, /etc/getty assumes the device is in the right mode. It then sets the device's communication parameters according to the final mode field, then calls the login program, passing it the line of input just read as an argument list.

/etc/getty will also examine the line of input to help determine final mode settings, occasionally with undesirable results. It will examine the parity bits of incoming characters to determine parity settings. If the line was ended with a carriage return, it assumes that carriage return will always be used as a line terminator, and translates incoming carriage returns to newlines. If the line was ended with a newline (line feed) character, this input translation will not be in effect, and carriage returns will not cause any special action. Another assumption /etc/getty makes, which can often be quite annoying, is that if if the input line contained upper case characters but no lower case characters, the terminal must be an old TTY-type of terminal with no lower case. It will then translate all incoming letters to lower case, and all outgoing letters to upper case. It is therefore important to always log in in lower case.

If the initial mode in the /etc/gettydefs entry was incorrect, the displayed prompt string was probably garbled. /etc/getty then expects the user to press the BREAK key on the terminal. This will tell the program that the current entry is not the right one and it should go on to try the next one, using the "next" field as a search key. With this next field, it is possible to set up chains or loops of related entries, usually with different baud rates, and cycle through them until the appropriate one is found. For example, in the sample file, the entry with the label "300B" is chained to the "1200B" entry, which is chained to the "2400B" entry, which in turn loops back to the "300B" entry. This group of entries would be appropriate for a 2400 bps modem that also supports 1200 and 300 bps. The argument to /etc/getty would be the label corresponding to the default (or most likely) baud rate to be used with the modem.

For the meaning of the individual mode options, consult the UNIX manual sections for stty(1), ioctl(2), tty(4) and/or termio(4).

In part II, which will appear in the fall, we will look at some of the ways to initialize serial ports that are not used for logins, such as dial-out ports, printer ports, or ports for other serial I/O devices.

Listing 1: Sample /etc/gettydefs File

```
9600# B9600 SANE # B9600 SANE IXANY #login: #19200
19200# B19200 SANE # B19200 SANE IXANY #login: #9600
110A# B110 HUPCL PARENB CS7 # B110 SANE IXANY TAB3 #login: #300A
300A# B300 HUPCL PARENB CS7 # B300 SANE IXANY TAB3 #login: #1200A
1200A# B1200 HUPCL PARENB CS7 # B1200 SANE IXANY TAB3 #login: #110A
300B# B300 # B300 SANE IXANY TAB3 #login: #1200B
1200B# B1200 # B1200 SANE IXANY TAB3 #login: #2400B
2400B# B2400 # B2400 SANE IXANY TAB3 #login: #300B
```

An Extended "file" Command

by Gilles Detillieux

The "file" command on the UNIX system can be extremely useful when you have a directory full of files, many without meaningful file name suffixes, and you don't know what's what.

Unfortunately, all of the new application programs coming out introduce a lot of new file formats that the "file" command doesn't know about. When you transfer and/or store PC files on your UNIX system, you have yet more new file formats to deal with. All too often with these files, the "file" command just reports the file type as "foo: ascii text" or "bar: data".

What is needed is a way of extending the "file" command so that it knows about these new file types. On some UNIX systems, this solution has been implemented: a file called "/etc/magic" contains a description of all of the "magic numbers" and other such file identification codes in use. By editing this file, you can tell the "file" command about any new file type that exists.

For those of us who don't have this new version of the "file" command, the shell script I will describe can solve the problem. The "ftype" shell command script, shown below, can be called instead of the usual "file" command.

```
E.g.    $ ftype foo bar
        foo:    Conforming PostScript text
        bar:    TIF format image data
        $
```

The shell script saves its arguments in the variable "flist", then for each file argument, the script calls "od" to dump the file contents, in hex. The "line" command pulls out the first line of the hex dump, and discards the rest. The "set" command, at the start of the loop, sets the shell's arguments to the numbers on this first line: the file offset is $1, and $2 to $9 are the first eight 16-bit words in the file. The following case statement looks at the first four of these words, trying to match them with known patterns. (The patterns in a shell case statement can contain the *, ?, and [] wildcards, as used for filename generation.) If the file begins with any of these patterns, the script displays the corresponding file type, in the same format as the "file" command does. If no match is found, the script calls the "file" command so it can take a stab at it. With all this processing going on for each file, this shell script runs a lot slower than the "file" command.

To add new file types to this script, just add new cases to the case statement, following the example of the existing cases. Remember that with shell case statements, the first match is the one that applies. If you have two different patterns that could match the same data, place the more restrictive one first. For example, a Conforming PostScript program begins with the same two bytes ("%!") as an ordinary PostScript program; it is what follows these two bytes that distinguishes them. In some cases, it may be appropriate to include some of the first eight words ($2 to $9) in the description, for example:

```
ffaafabe*)    echo "$file:\tFoobar data (version $4)" ;;
```

I have added several locally-developed file formats to my "ftype" shell script (which I omitted from the listing). Since many of our applications don't force users to put meaningful suffixes on file names, there were a lot of "mystery files" on our system, all of which the "file" command insightfully identified as "data". This shell script has really helped clear up the confusion. I hope it can help others out there too.

```
Listing 1: "ftype" shell script
-------------------------------
:
# ftype - extended "file" command
#  - shows file types of files not recognized by "file" command
#

flist="$*"
for file in $flist
do
        set -- `od -h $file | line`
        case "$2$3$4$5" in
        1b29733*5700*)          echo "$file:\tHP PCL downloadable font" ;;
        1a0[1-8]*)              echo "$file:\tSEA ARC format archive" ;;
        303730373037*)          echo "$file:\tascii cpio archive" ;;
        677073726173203[1248])  echo "$file:\tgpsras format image data" ;;
        f001*)                  echo "$file:\tGPS format graphic data" ;;
        49492a0008000000)       echo "$file:\tTIF format image data" ;;
        252150532d41646f)       echo "$file:\tConforming PostScript text" ;;
        2521*)                  echo "$file:\tPostScript text" ;;
        626567696e203*)         echo "$file:\tuuencode format text" ;;
        1f1e*)                  echo "$file:\tpacked file" ;;
        1f9d*)                  echo "$file:\tcompressed file" ;;
        4d5a*)                  echo "$file:\tDOS executable" ;;
        f0*)                    echo "$file:\tDOS library" ;;
        80*)                    echo "$file:\tOMF object file" ;;
        4148*)                  echo "$file:\tHalo(TM) bitmapped font file" ;;
        0002000000060404)       echo "$file:\tLotus worksheet" ;;
        31b30000*)              echo "$file:\tMS-Word document file" ;;
        ff575043*)              echo "$file:\tWord-Perfect document file" ;;
        *)                      file $file ;;
        esac
done
```

Technical UNIX®User Group

Minutes From the Business Meeting
May 9, 1989

1. Minutes:

MOTION : (Kirk Marat.) The minutes from the April 11, 1989 meeting be approved.

SECONDED : (Darren Besler)

In Favour : 11 Opposed : 0 Carried

2. Membership Report:

Change Paul Hope's Address to 351 Tache Avenue.

3. Newsletter Report:

Nothing to report.

4. Treasurers Report:

The June party will require some supplies. Since there is sufficient funds TUUG can contribute a maximum of $50.00 for these supplies.

MOTION : (Darren Besler) A maximum of $50.00 can be contributed to the party fund from the TUUG account.

SECONDED : (Susan Zuk)

In Favour : 11 Opposed : 0 Carried

5. Miscellaneous:

Nominations - Elections for the next term will be held in October. Nominations for the following positions will be held in September. Think about nominating yourself or another member for these positions.

President
Vice-President
Treasurer
Newsletter Editor
Membership Secretary
Secretary
Meeting Coordinator

Muddy Waters - Muddy Waters will discuss with their executive about connecting our bulletin board to their bulletin board as well as working closer together with this group.

Autodial Modem - A future consideration for the group to buy an autodial modem for the hub communication machine The group will define the requirements at a future meeting and will put a motion to the group.

Usenet - We may be able to obtain a USENET link in the future.

June Barbeque - The June barbeque will be held at Kirk Marat's residence. Gas barbeques will be brought by Paul Hope and Richard Willacy. Pat Mcdonald will bring folding tables. Members are to bring lawnchairs, their own hamburgers, steaks, or whatever, and your own liquor. The group will supply extras. If there is rain we will move inside. Please R.S.V.P. by Friday, June 9th. Directions to Kirk's house will be located in the June newsletter.