September 1989 $2.50
Volume 1, Number 10

Technical UNIX® User Group

newsletter of the Technical UNIX® User Group

This month ...

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

Your article could go here!

Become an active member of the Technical UNIX User group and submit your newsletter article today.

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

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

Copyright Policy and Disclaimer

ANNOUNCEMENT...

Meeting Location Change:
The September meeting location will be provided by UNISYS Canada Inc., Suite 300-1661 Portage Ave (UNISYS Building). 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

We are back. After a two month break, we are back for another year. Despite the break, here I am again, rushing to beat a deadline with this column. It hardly seems like we had a break at all! I hope most of you had more time to enjoy the summer than I did.

Anyway, we are back and ready for business this fall. The executive met during the summer to plan this newsletter, and to give some thought to the next couple of meetings. As most of you are by now aware, September's meeting (Tuesday Sept. 12) is when we will have nominations for next year's executive; elections will be held at the October meeting. Nominations will remain open until the October newsletter deadline, so that the list of candidates can be published then, so you all have until about September 23 to think about getting involved. I hope we will have a good turnout at the September meeting and that we can get a full slate of candidates for the election. Please consider putting your name in; it's not a whole lot of work (especially if we get a lot of you involved), and it can be very rewarding.

The October meeting is the start of the new year for the group, which means (of course) time to renew your membership. The fee remains $20 for the whole year, and goes mainly toward postage and stationery cost for the newsletter. I can't think of any other user group with such a low membership fee; this shouldn't break the bank for anyone, and I hope you will all renew promptly, so we can be in good shape for the new year. If you are not a member yet, take a look through this newsletter and consider what it's worth to you. We have lots of informative articles throughout the year, and lots of really up to date information. Where else can you get all this in this city, and for only $20 a year?

We also have some new projects in the works for the new year. One of them is to set up a UUCP "network" for any interested members, so that we can exchange messages, news items, and newsletter articles (hint, hint). Another thing we want to do is develop a liaison with the Muddy Waters User Group in town, which has a UNIX conference on its Bulletin Board, and some public domain UNIX software. They are interested in our group, and a working relationship would be beneficial to both groups.

Finally, one other idea that we have in the works is the formation of a special interest group (SIG) within TUUG which would cover UNIX industry-related issues (such as standards, organizations, market trends, etc.) as opposed to the more technical and user-oriented issues dealt with at our regular meetings. This Industry SIG, which would have its own meetings, would be a part of the TUUG membership, and would hopefully contribute to the newsletter (meeting minutes, at least). This would allow the group to meet the needs of a larger segment of the local UNIX community than it does now, and would strengthen our membership; all of which means better service to you, our members. The exact focus and goals of this SIG are of course very vague right now, and would be defined by those interested in being a part of it. Anyone interested in knowing more should contact me; we will have a meeting in late September or October, to get the SIG started, once there is sufficient interest in such a thing.

It looks like our September meeting will be held at Unisys again. The presented topic, after the business meeting, will be the ever-popular System Administration Workshop. Within that framework, we will cover whatever subjects you all seem to be interested in seeing. Hope to see you all out there!

The fortune file

This month's fortune comes care of Susan Zuk and the Joke Network.

```
... A programmer is a person who passes as an exacting expert on the
basis of being able to turn out, after innumerable iterations, an
infinite series of incomprehensive answers calculated with
micrometric precisions from vague assumptions based on debatable
figures taken from inconclusive documents and carried out on
instruments of problematical accuracy by persons of dubious
reliability and questionable mentality for the avowed purpose of
annoying and confounding a hopelessly defenseless department that
was unfortunate enough to ask for the information in the first
place.
```

Initializing Serial Ports, Part II - Initializing Non-login Ports

By Gilbert Detillieux, Info West Inc.

In part I of this article, which appeared in the June 1989 newsletter, we looked at how login ports are initialized using /etc/getty, and how entries in /etc/gettydefs are defined. In this part, we will look at a few ways to initialize serial ports for other uses, such as for communication with other systems, accessing printers, and so on.

To initialize communication parameters such as baud rate, parity, etc., on a serial port that is not enabled for login, it is not sufficient to simply use the stty command and expect the mode to stay the way you set it. This is because UNIX always re-initializes a closed serial port when it is first opened by a process. This initialization sets up some default communication parameters which are not always correct for the equipment connected to the port. By using stty, the mode you set up is only "remembered" as long as the port is open, i.e. as long as some process is accessing it.

There are several ways to get around this problem. Many solutions depend on what applications will be accessing the port.

Dial-out Ports

When using a serial port to dial-out to another system, or to connect directly to another system, using UUCP and/or CU, the port's baud rate is specified in a device specification file in /usr/lib/uucp (L-devices for old UUCP, Devices for HDB). For example to specify a particular port for dial-out at 1200 bps, for HDB UUCP, we might add the line:

```
ACU tty9 - 1200 hayes
```

Other communication parameters, such as parity, stop-bits, and handshaking, are hard-coded into the UUCP programs, and cannot be specified. The corresponding port on the system being called should be set up for login with compatible parameters.

Printer Ports for LP Spooler

To initialize a serial printer port used by the LP Spooler, the most straightforward way is to add an stty command to the interface program for that destination. This interface is simply a shell script, found in the /usr/spool/lp/interface directory, and is invoked by the LP scheduler whenever a job is to be printed to that destination. The scheduler first makes sure the port is free, then opens it before calling the interface. Thus a simple stty command in the interface is sufficient to set up the port for this run. For example:

```
stty 9600 cs7 parenb ixon opost onlcr <&1
```

This sets up the port for 9600 bps, 7 data bits, even parity, and XON/XOFF protocol for output flow control. It also enables the translation of output newlines to CR/LF pairs. The redirection at the end is required since the interface is called with the destination port open as the standard output, which must be redirected to the standard input of stty.

This may be sufficient for the way ports are typically used on UNIX systems. But what about ports that are used by programs to communicate with some other piece of hardware? What about printer ports that are to be used directly, rather than by a spooler? What about non-LP spoolers that don't allow you to customize the interface? Well, there are still a couple of tricks that can be used.

Using /etc/getty on Printer Ports

One trick that is used on some UNIX systems is to initialize serial printer ports the same way as login ports, using /etc/getty. The reasoning is quite simple: /etc/getty initializes the port using parameters from a text file that can be customized, prompts the user to login (with a customizable prompt), then waits for a response. Entries can be set up in /etc/gettydefs, described in part I, for initializing printer ports to the correct mode by specifying the initial mode; the final mode can be left blank; the normal login prompt is replaced with something more appropriate for a printer, such as a form feed:

```
9600p# B9600 BRKINT IGNPAR ISTRIP ICRNL
 IXON OPOST ONLCR CREAD ISIG ICANON CS7
PARENB   CLOCAL   ##^L#9600p
```

The port is then enabled with an /etc/inittab entry such as:

```
lp01:2:respawn:/etc/getty  lp01  9600p
```

Since the printer is unlikely to respond to the login, /etc/getty will just wait forever, keeping the line open, and keeping the initial mode set up for the port. This works fine in many cases, but the form feed issued as a "prompt" can be quite annoying, especially when printing on pre-printed, numbered forms.

The above solution also will not work for ports used for bidirectional communication, since /etc/getty is waiting to read anything that comes into the port. What is needed is a simple program to replace part of the task normally accomplished by /etc/getty.

The /etc/initty Program

The shell script shown in Listing 1 is a simple program that will initialize non-login serial ports in the same way as /etc/getty, but will work even on ports used bidirectionally, and without producing any extraneous output. Place the script file in /etc, or any directory you wish, and enable execute permission, at least for root. To invoke it in /etc/inittab, simply use it as you would have used /etc/getty for non-login ports:

```
lp01:2:respawn:/etc/initty   lp01   9600p
```

Invoked in this way, it will open the port /dev/lp01, set its mode according to the initial mode for the "9600p" entry in /etc/gettydefs, then go into an infinite loop reading and discarding any input it receives. For bidirectional communication ports, the reading and discarding of input can be suppressed by specifying the "-nr" (no read) option as a third argument:

```
t05:2:respawn:/etc/initty  tty05  9600  -nr
```

The program will still go into an infinite loop, simply to keep the port open with the specified mode, but no reading or writing is done, leaving the port free to be used by other programs.

The program works as follows. It first opens the port specified by the first argument. If the port is not ready for communication, i.e. there is no "carrier detect" signal, the open will be suspended until the port is ready.

Next, the stream editor, sed, is used to find the entry in /etc/gettydefs whose label is the one specified by the second argument, extract the initial mode field of this entry, and put it into a form usable by stty (translated to lower case, and the initial letter B removed from numeric baud rate). Next, stty is invoked with the output of sed as its argument list. The first two options to stty, "-parenb cs8", simply set up some initial defaults, and ensure that stty will receive at least some arguments, even if there is no output from sed.

Finally, the program goes into one of two infinite loops, based on the third argument. It will either continuously read input lines and do nothing with them, or go to sleep for a long time. In either case, it keeps the port open indefinitely, with minimal impact on system performance.

This solution is admittedly a "band-aid" solution. The best approach to take is to specify the port initialization directly to whatever application (communication program, print spooler, etc.) will be using the port. However, in cases where this is impossible, or impractical, this little program has really come in handy.

****

Listing 1: The /etc/initty Program

```
# initty  --  tty initializer for non-login serial ports
#
#   usage:                sh /etc/initty dev mode [-nr]
#
#   where:
#                               dev    is device name in /dev for port
#                               mode   is a mode in /etc/gettydefs, as for getty
#
#  - opens port for file descriptors 0, 1 and 2
#  - sets up mode by searching /etc/gettydefs for mode, and running stty
#  - waits in a loop, reading & discarding input
#  - if arg 3 is '-nr', sits in idle loop and does no reading

{                         # braces needed to read entire script before changing stdin
exec  <  /dev/$1  >  /dev/$1  2>&1        # open device

/bin/stty  -parenb  cs8  '/bin/sed  -n  "/^$2#/{
                               s/^$2#\([^#]*\).*/\1/
                               s/B\([0-9][0-9]\)/\1/
                               y/ABCDEFGHIJKLMNOPQRSTUVWXYZ/abcdefghijklmnopqrstuvwxyz/
                               p
                               q
                          }"  </etc/gettydefs'

case  "$3"  in
-nr)                      # Pause without reading from device
                          while : ; do /bin/sleep 32767; done
                          ;;
*)                        # Wait for input, discard it
                          while read x; do : ; done
                          ;;
esac
exit
}
$
```

Preparing Your Computer For UUCP

By Susan Zuk, UNISYS

As most of you know, TUUG is in the first stages of setting up a network. Members will be able to communicate with each other by means of a UUCP (UNIX-to-UNIX Communication Program) connection. At the present time the executive members are linked together. Anyone who is interested should contact Gilles or Gilbert Detillieux for the appropriate phone number, user name and password. You will need to have dial-out capabilities as the hub machine's modem is set-up as a slave and therefore has autoanswer capability.

To help in making your connection to the network the following procedure explains the requirements for setting up UUCP. This article will explain the files and file contents for the UNIX 5.2 version of UNIX. A future newsletter will explain the procedure used in the HoneyDanBer method of UUCP found in UNIX version 5.3.

The files used for UUCP are found in various places in the system. Below is a list of the various directories and the information they contain:

/usr/bin - uucp programs
/usr/lib/uucp - uucp set-up files, administration files and programs
/usr/spool/uucp - log files, temporary files holding transferring data, debugging information and communication flags
/usr/spool/uucppublic - data files awaiting transfer in or out commands. The default mailbox of uucp.

When preparing the system to become a UUCP link you must check the computer's system name. Type the following command to display the appropriate information:

```
type : uname -n
response : tuug1
```

In this case, tuug1 is the name which your kernel is using. You want to have a unique system name for the same reason you have a unique home address. You want your mail to arrive in your "mailbox"!

After having found your current system name, you can either change it or leave it as is. Changing the system name may require remaking the kernel or using the command *"uname -S newname"*. Check the system manuals for the appropriate procedure.

Whether you are changing the name or not, you must check the file /usr/lib/uucp/SYSTEMNAME to make sure the current name is listed. Edit the file with the proper name if it is wrong.

The following files, found under the directory /usr/lib/uucp, must be edited to include the correct information: L-devices, L.sys, L.cmds, USERFILE.

L-devices - This file contains the list of all lines that are directly connected to other systems. The file contains attributes of the lines and whether the line is a permanent connection or is used with a dialer. The format is:

```
type line call-device speed protocol
```

```
e.g.    DIR ttya 0 1200
```

Type will be either DIR or ACU depending on whether an automatic dialing modem or a direct connection is made. The keyword DIR indicates a connectingcommunications line. **Line** is the device name in /dev which is connected to the modem. The **call-device** field is ignored. The 0 is used as a place holder. **Speed** refers to the line speed of the connection. **Protocol** is an optional field that needs only be filled if the connection is for a protocol other than the default terminal protocol.

L.sys - This file lists and describes all systems uucp knows about. It gives UUCP login and password information on how to access other systems, what to expect and what to send.

```
system-name time device class phone login
```

```
e.g. tuug2Any ttya 1200 ttya"" ATZ\r\c OK
ATD,4564933\\r\c CONNECT \r\c ogin:-\r\c-ogin:-
\r\c-ogin: nuucp
```

The **system-name** is the name of the remote system (1 - 6 characters). **Time** indicates the days of the week and times the remote system may be called, (e.g. MoTuTh0800-1730). The word "Any" may be used to specify any time. The word "none" is used to indicate no call out, the local system is then a slave. The **device** field is the tty device name used for the call. The word "slave" would be used to indicate the local system is to be polled. **Class** refers to the line speed. The **phone** field normally will match the device field. It may contain the phone abbreviation used in the L-dialcodes file and/or the appropriate modem control information and dial number. The modem control information and phone number may be incorporated into the login field. **Login** contains login information and is given as a series of fields and subfields in the format : expect [send expect] send . The expect string looks for the data the remote system is sending (eg. login prompt and password prompt), the send substring is sent if the prior expect string is not read, the send string is the data to be sent back when the expect string is read. The process continues until the expect send pairs are used up.

USERFILE - This file defines users and remote system that can access this your system.

```
login,sys pathnames

e.g.nuucp,tuug1 /usr/spool/uucppublic
    zuksue,tuug2 /usr/spool/uucppublic,/usr/acct
```

The login field refers to the login name of a user on the remote system. **Sys** is the system name of the remote computer. **Pathnames** are the pathnames where the user is allowable access.

L.cmds - This file contains a list of commands which may be executed from a remote system.

```
e.g.  rmail
      uucp
      lp
      ps
      who
```

The files listed above are to be owned by uucp. If this is not the case then use the chown command while in superuser mode.

```
e.g.  chown uucp L.sys
```

The next set of files must be verified in order for uucp to function properly. The file inittab, found under the directory /etc should be of the form:

```
terminal-id  init-code respawn-mode  port-program device baudrate

e.g.for outgoing calls : ta:1:off:/etc/getty ttya 1200
    for incoming calls: ta:1:respawn:/etc/getty ttya 1200
```

Your /etc/passwd file should have two entries such as this :

```
e.g. uucp:8jr8j9gj49guf9:5:1:uucp:/usr/spool/uucppublic:/usr/lib/uucp/uucico
.... nuucp:Yuijy784j7j8o:5:1:uucpuser:/usr/lib/uucp:/bin/sh
```

Line 1 allows uucp commands to be executed and line 2 allows a user called nuucp to log into the remote system if permission is granted to him/her by the remote system.

The device being used for the connection must be owned by uucp and must have read and write permissions set.

```
e.g.  ls -l /dev/ttya
      chmod  666  /dev/ttya
      chown  uucp  /dev/ttya
```

You are now ready to try and send information to a remote system. The next command listing will tell the computer to send a file called junk to the system tuug2 under the directory /usr/uucp/uucppublic. The first line queues the command by using the -r option. This allows the system to be forced to send the file with a command like on line 2, or wait for the system to be polled by another machine, or wait for a crontab file to make the request for communications. The crontab file is discussed later. The second line forces the file to be sent to system tuug1 and also generates debugging code (-x9) which lets you see if there is a problem with your transmission.The *2> debugfile* places the information received from the command into the file called debug instead of the screen.

```
uucp  -r  /junk  tuug2!/usr/spool/uucppublic
/usr/lib/uucp/uucico  -r1  -x9  -stuug2  2>  debugfile
```

If you want to send a mail message type the following:

```
mail  tuug1!zuksue
```

After you have typed the mail command, the terminal will wait for you to type your message and press control D to end the message. If you have auto-dial facilities the message will be sent to zuksue on remote machine tuug1, otherwise you will have to dial out manually to the remote machine to have your message sent. If the message cannot be sent it will wait until it is either polled or a later transmission begins.

If you are having trouble making a connection to the remote system check the following:

* The directory /usr/spool/uucp for any LCK..[systemname], [devicename], or [filename] files. If there are any remove them any try your connection again
* The permissions on the device and files listed previously
* The modem connection - dial the remote system's number to make sure the connection is being made
* The L.sys file for correct syntax
* Make sure the remote system is working
* Remove any unecessary STST.[systemname] files (read before removing)
* Make sure your uucp partitions are not full

A final word. If you have an auto-dial modem you may want to have your system check for your mail or send your mail at certain times of the day or week automatically. Edit your uucp crontab file in the following way:

```
15 0,12 * * 0-3,5-6 /usr/lib/uucp/uucico -r1 -stuug1 >/dev/null 2>&1
```

This line states that at 12:15a.m. and p.m. every day except Thursday, the modem will dial-up the machine called tuug1 and send and receive any waiting mail.

You are now ready to set-up your uucp connection.

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 the executive about connecting our bulletin boards 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.

Usernet - We may be able to obtain a USERNET 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 lawnchairs. Members are to bring lawnchairs, their own hamburgers, steaks, or whatever, bring your own liquor. The group will supply the rest. If there is rain it will be inside. Please R.S.V.P. by Friday, June 9th. Directions to Kirk's house will be located in the June newsletter.

Technical UNIX® User Group

Agenda

for

Tuesday, September 12, 1989

7:30pm

UNISYS

Canadian Indemnity Building

300-1661 Portage Avenue

1. Round Table	7:30
2. Business Meeting	8:00
a) Minutes of May's Meeting	
b) Membership Secretary's Report	
c) Newsletter Report	
d) Treasurer's Report	
3. Break	8:30
4. Presented Topic	8:40
System Administration Workshop	
5. Adjourn	9:30