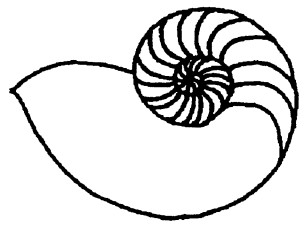


Technical UNIX®Users Group

February 1989 $2.50
Volume 1, Number 5

newsletter of the

Technical UNIX® User Group

This month ...

The President's Corner
UNIX Variations
The Init Program and The Inittab File
Agenda for February 14 Meeting

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

Song found on a local Bulletin Board

"Write in C"
(To the Melody "Let It Be" by The Beatles)

When I find my code in tons of trouble,
K&R they come to me,
Speaking words of wisdom,
Write in C.

And as the deadline fast approaches
And bugs are all that I can see,
Somewhere someone whispers
Write in C.

Write in C, write in C,
Write in C, yeah, write in C,
Don't use Pascal or LOGO,
Write in C.

I used to write a lot of FORTRAN,
For science it worked flawlessly,
Try using it for graphics,
You'll write in C.

If you just spent nearly 30 hours
Debugging some assembly,
Soon you will be glad to
Write in C.

Write in C, write in C,
Write in C, yeah, write in C,
Only WIMPS use BASIC,
Write in C.

Write in C, write in C,
Write in C, yeah, write in C,
Don't even mention COBOL,
Write in C.

(Author unknown)

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
Newletter Editor:	Darren Besler	(W) 934-5475
		(H) 254-3392
Membership Sec.:	Pat Macdonald	(W) 474-9870
Information:	Gilbert Detillieux	261-9146
	Susan Zuk	(W) 786-8483

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

Copyright Policy and Disclaimer

ANNOUNCEMENT...

Meeting Location Change:
For February's meeting, we will be gathering at UNISYS, 300 - 1661 Portage Avenue. In order to have an idea of the number of people to expect please RSVP if you are planning on coming out to the meeting. This can be done by phoning Susan Zuk at 786-8483 and leaving a message that you will be attending the meeting. This can be done up until 5:00 pm on February 14, 1989.

President's Corner

by Gilbert Detillieux, President

Greetings to all fellow snow-bound city dwellers. As I am writing this, we are in the middle of the second blizzard of the month, and I am suffering from the second cold or flu-like virus this month! Ah, but soon it will be spring, so let's all hang in there until the bugs going around are the kind you can step on. Speaking of bugs and viruses...

Virologists and wormologists take note: If last month's article on the Internet Worm and all the press hype surrounding it was not enough to keep you satisfied, you'll love the January issue of Unix Review. There were three articles about (or at least mentioning) the Internet Worm. First, there is a special report by a member of the Berkeley team that cracked the worm. Next, the C Advisor column describes some of the attacks used by the worm, and the program bugs that allowed those attacks; he also gives some hints for practicing defensive C programming. Finally, even the Devil's Advocate discusses the worm with his usual off-beat sense of humor. (Or should that be humour?)

In the same issue of Unix Review, a new column called Daemons and Dragons (cute title) talks about start-up and shut-down procedures on UNIX. You should probably avoid this article if you are just learning this stuff, as it is very misleading. The author discusses, as if they were generally applicable, many things that are particular to the brand of machine he is using. For example, the kernel is loaded from */vmunix*, rather than the more common /unix; he assumes that the start-up file */etc/rc.local* exists on all systems; and so on. Those of you who were at the January workshop know that start-up and shut-down procedures vary greatly from one system to another, and that it is hard to generalize. With that in mind, an article elsewhere in this newsletter describes the */etc/init* program and the use of inittab, as they would typically be used on a System V machine. This article will serve as a review of the main topic of the January workshop, for those who couldn't make it to that meeting.

The January workshop on system administration was very lively, like the one in November, and again, we got off course from our original topic of discussion. So it was decided that, for the February meeting, we'll be back at Unisys (300-1661 Portage, between St. James St. and Route 90) to continue the system administration workshop. The discussion should be focussed primarily on system accounting and tuning, but as usual, we'll be open to whatever other topics seem to be of interest to everyone at the meeting.

Here again is a list of some of the suggested topics that can be covered at the workshop:

1. System tuning
   a) The sar utility (system activity reporting)
   b) Kernel parameters (clists, files, buffers, etc.)
   c) Generating a new system
   d) Error reports

2. System accounting (process, login accounting, etc.)

3. Communications
   a) Serial terminals (stty, gettydefs, terminfo)
   b) Printer setup (lp, lpadmin, custom interfaces)
   c) UUCP (use, setup, HDB-UUCP)
   d) Networks (Ethernet, token ring, X.25, Usenet)

4. Security (passwords, file permissions, etc.)

Apparently, some people had trouble getting into the building at the last meeting because the security guard wasn't at the front desk. (At least, that is what happened to Darren, our editor.) For the next meeting, we will make sure we have someone in the lobby to let you all in; but if you arrive late, you may be out of luck.

Hope to see you at the next meeting (Feb. 14). A happy Valentine's day to all!

The fortune file

This months fortune comes care of Darren Besler.

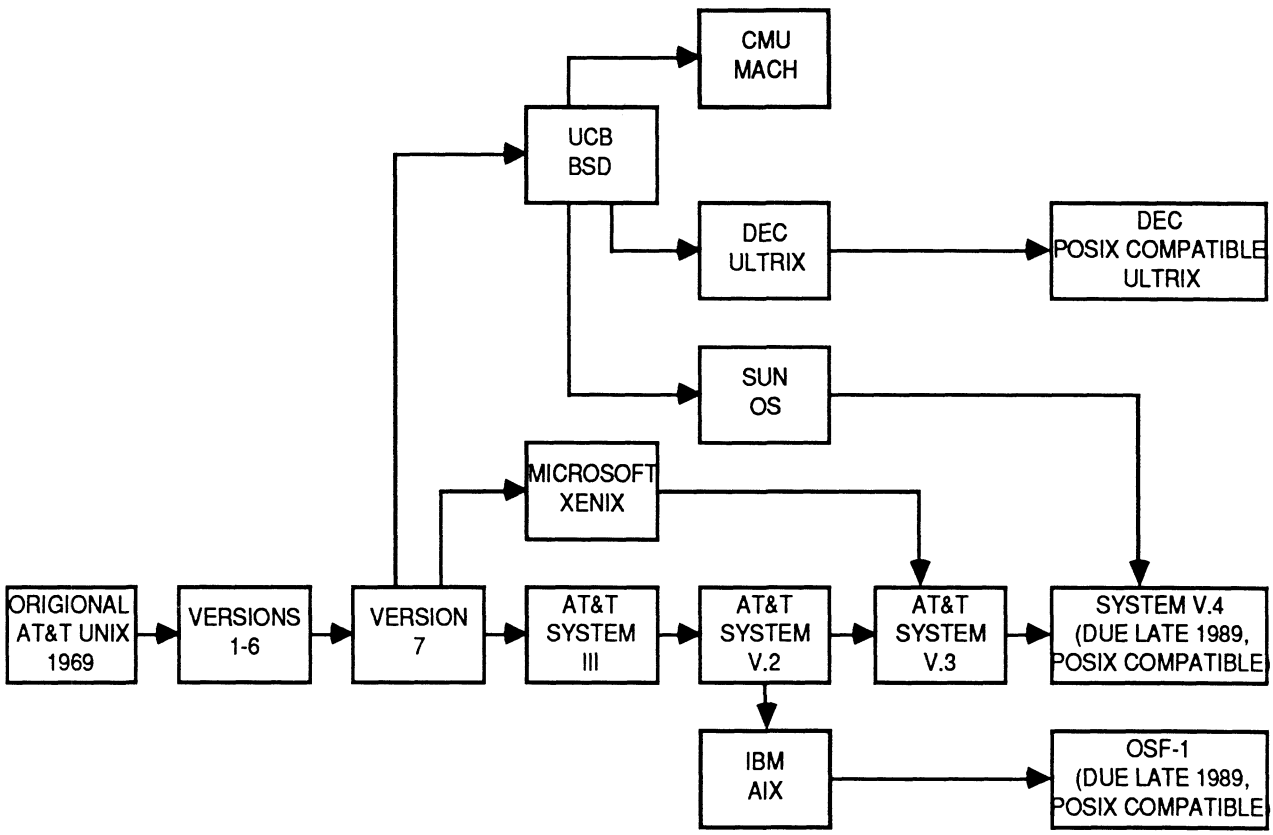
```
If not controlled, work flows to the competent person until he is submerged.
```

UNIX Variations

by Darren Besler

Since UNIX was born in that AT&T laboratory back in 1969 it has gone through many changes. Here we can mark its 20th birthday and it is just starting to be accepted as a serious contender in the operating system market place. Most operating systems would or should be retiring when they approach that age. UNIX has had time to ferment and age in the backgroud, where extremely dedicated users could initiate changes if they so desired. Even after 20 years, there still isn't a single standard for the UNIX operating system. Those people who did initiate changes, and could not agree with others, developed their own versions of UNIX. Now we have several committees fighting for a single UNIX that is a standard. Such committees are Open Systems Foundation (OSF) and Unix International Inc.

The following gives a reasonable indication of the eveolution of UNIX and where its stands today.



The Init Program and Inittab File

by Gilbert Detillieux

Start-up procedures on a UNIX system can be quite difficult to understand at first. This is partly due to the fact that there are a lot of steps, involving a lot of programs that are automatically executed. A further complication is that there is very little standardisation in this area; almost every different brand of machine has different procedures, and there may also be differences from version to version, and site to site. This article will attempt to describe some of the common elements to system start-up on all System V derived machines. This will be applicable to many BSD derived systems as well, but not to XENIX based systems, where the inittab file is not used.

As a UNIX system is started up (or booted), the first thing that happens is a small bootstrap program (usually in ROM), loads a second stage bootstrap program from the disk that contains the UNIX root file system. This bootstrap program will then load the UNIX kernel, usually found in the file */unix* on the root file system. Once the kernel is loaded and run, it will set up memory and get ready to run other processes. It will then split itself into two processes. Process 0, called the *swapper*, simply executes kernel code to reschedule the other processes from time to time. Process 1 executes a program called */etc/init*, which will be responsible for the rest of the system initialisation.

This *init* program is described in the UNIX manual as "a general process spawner." To be general and flexible, it reads instructions from a file called */etc/inittab* to find out what processes it should spawn. This is a text file that can easily be changed to suit your needs. It consists of a series of lines of the following format:

```
id:level:action:command
```

The *id* field is a unique identifier of two or more characters; only one such ID should exist for a particular run level (see below), but separate lines with the same ID can exist for different run levels.

The *level* field is a sequence of zero or more digits from 0 to 6 (this maximum may vary from system to system); each such digit is called a run level. Most lines will contain only one such digit. If a line contains more than one such digit, it will apply to each specified run level. If a line has an empty *level* field, it will apply to all run levels.

Run levels are the way init allows the system to run a different set of processes at different times. Init will be at one particular run level at a given time, and will then only run commands that are appropriate for that level. This is typically used to set up two separate modes, called single-user (or maintenance) and multi-user modes. On many systems, run level 1 is designated as single-user mode, and run level 2 as multi-user, but this can be different on other systems. For instance, the NCR Tower systems use 0 for single-user, and 1 for multi-user.

The *action* field, which follows the level field, tells init how to treat this line, usually telling it how to run the command indicated in the *command* field, which is the rest of the line. Command syntax is that of the Bourne shell; commands can continue onto another line by ending the line with a backslash, and comments can be placed after a command by using a sharp (#) character.

As an example of how an */etc/inittab* might be set up, and to explain some of the common actions, listing 1 shows a sample file, which is a simplified version of the */etc/inittab* file found on a Masscomp computer.

The first line contains the **initdefault** action. This line tells init that the default run level when init is first run is level 1. If there is no line with the **initdefault** action, init will prompt on the console for an initial run level. Once the run level is set, init will only follow the actions for lines applicable to that run level. In this example, lines containing the digit 1 in the level field, or lines with an empty level field, will be used.

The next line, with the **wait** action, tells init to run the command file */etc/rc* whenever it enters run levels 1 or 2, and wait for this to finish before going on. Normally init will start up a command, then go on to the next line without waiting. This *rc* file (short for run command) is simply a list of shell commands to perform various start-up activities before allowing logins. On many systems this is only run at level 2, but on the Masscomp, it is run at level 1 as well.

Once the *rc* command runs to completion, init will then go on to the next applicable line, the one with the **co** as the *id* field. This line is actually continued on a second line, as indicated by the backslash at the end of the first line. The combined line, with the **respawn** action, causes init to run the *env* command to set up several environment variables, then run an interactive shell (the C shell) on the console. The **respawn** action tells init to start this command, then go on to the next line without waiting; also, whenever this command terminates, init should restart (or respawn) it again.

The only other line which is applicable at level 1 is the line with the **up** as its *id* field, since it's *level* field is empty. This line starts up the */etc/update* program, to periodically update the disks, and respawn it if it terminates.

The remaining lines are used only at level 2. To get init to switch to that run level, we would issue the "*init 2*" command. In the example, this is done automatically by */etc/rc* if the start-up is clean, or it can be typed in at the console if the shell is running. (Note that if */etc/rc* changed the run level to 2, init would never get around to running the interactive shell at level 1, since it was waiting for *rc* to terminate.)

In any case, once init is at level 2, it will kill all level 1 processes that are still around, and that are not appropriate for the new level. It will then rerun the */etc/rc* command, which will this time do the appropriate start-up for level 2, such as starting up the */etc/cron* daemon process. Next, init will run the */etc/update* command, and several */etc/getty* commands.

These */etc/getty* commands will allow logins on the various terminal ports, and whenever a login or shell session terminates, a new one will be respawned. The line with the **off** action will simply be ignored for now, and any earlier process running with the **01** *id* field will be terminated. By toggling a line between **respawn** and **off** actions, we can selectively enable or disable logins on particular ports. After making any change to the */etc/inittab* file, we can force init to reread it and apply the changes by issuing the "*init q*" command. The */etc/getty* command will be covered in more detail in a future article.

The last three lines in the sample */etc/inittab* all contain the **once** action. These commands are run once each time init switches to the applicable run level, and init goes on to the next line without waiting. In the example, these lines start up the TCP/IP network daemon processes whenever level 2 is entered.

There are also many other valid actions than those used in the sample file, such as *boot* and *bootwait*, for running commands only once at boot time, and *powerfail* and *powerwait*, for running commands to perform power failure recovery. This article cannot hope to cover everything there is to say about init and inittab. To study this topic further, you should probably print out the contents of the */etc/inittab* and */etc/rc* files on your system, then read up on *init(1M)* or *init(8)*, and *inittab(5)* in your UNIX manuals.

Listing 1: Sample /etc/inittab file.

```
is:1:initdefault:
rc:12:wait:/etc/rc 1>/dev/console 2>&1 #run com
co:1:respawn:/bin/env HOME=/ LOGNAME=root \
PATH=:/bin:/etc:/usr/bin /bin/csh<>/dev/console >&0 2>&0
up::respawn:/etc/update
00:2:respawn:/etc/getty tty0 9600
01:2:off:/etc/getty tty1 9600
02:2:respawn:/etc/getty tty2 1200
04:2:respawn:/etc/getty tty4 9600
nd:2:once:/etc/net/netd 2>/usr/adm/netd.log
wd:2:once:/etc/net/rwhod 2>/usr/adm/rwhod.log
td:2:once:/etc/net/talkd 2>/usr/adm/talkd.log
```

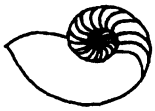
A Whereis Command

Contributed by Kirk Marat

[Ed. The following is a shell script which Kirk found on his UNIX system from Brüker Analylische Messtechnik, GMBH, Karlsrube, West Germany.]

```
# Whereis:  Descend a directory hierarchy and find "where is the file"
#
# Author:   Martin Doerr      25/8/85

case $# in
      1) find . -depth \( -name $1 -prinbt \) ;;
      3) find $3 -depth \( -name $1 -print \) ;;
      *) echo wrong usage: 'whereis f1' or 'whereis f1 in dir2'
```



Minutes From the Business Meeting
January 10, 1989

1. Minutes:

CLARIFICATION from the November Minutes:

Point 4 line read : The group name TUUG (Technical UNIX User Group) has been registered as a non-profit group and is registered for three years at which time it is renewable.

Point 4 should read : The group name TUUG (Technical UNIX User Group) has been registered by way of a name notation and is registered for three years at which time is renewable.

Point 5 line read : The official name of the UNIX group will be TUUG (Technical UNIX User Group).

Point 5 should read : The official name of the newsletter will be "Newsletter of the Technical UNIX User Group".

MOTION : (Susan Zuk) The minutes from the November 8th, 1988 meeting be approved.

SECONDED : (Peter Somers)

In Favour : 12 Opoosed : 0 Carried

2. Membership Dues:

Four new members joined today which gives us a paid membership of 22. Membership receipts (for members just joining) will be mailed out with the February newsletter.

The subject of giving our membership list out to companies or other interested parties was brought to the table by Pat Macdonald. To control this request in the future a motion was made.

MOTION : (Pat Macdonald) Membership forms will be created in time for next year's enrollment. Membership forms will include a question asking members if they would allow their addresses to be given out to companies and/or other groups.

SECONDED : (Grant Madison)

Approved : 12 Opoosed : 0 Carried

MOTION : (Paul Hope) For the interim period, the group willl give out its membership list as long as it only goes to the requesting party and no further. The recipient may only contact the members by mail (not phone or in person) and phone numbers will not be provided.

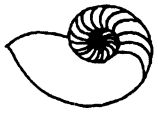
SECONDED : (Pat Macdonald)

Approved : 12 Opoosed : 0 Carried

A bulletin board was discussed and Paul Hope and Kirk Marat are to pursue the subject.

3. Newsletters:

No report was made.



Technical UNIX® User Group

Agenda

for

Tuesday, February 14, 1989
7:30pm
UNISYS
Canadian Indemnity Building
300-1661 Portage Avenue

1. Round Table 7:30

2. Business Meeting 8:00
   a) Minutes of January'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
   a) Resource Accounting
   b) System Tuning
   c) Backup/Restore Procedures

5. Adjourn 9:30