January 1990 $2.50
Volume 2, Number 4

Technical UNIX®User Group

newsletter of the Technical UNIX® User Group

This month ...

Late Breaking News...
Next Meeting to be held at the
UNIVERSITY OF MANITOBA
See ANNOUNCEMENT for details

Thoughts From The Editor

By Susan Zuk

Happy New Year!!! Hope the holidays passed joyously and you are all ready for a new year and decade.

This is the month which Uniforum is holding its annual UNIX Show. It is being held in Washington, D.C. from January 23-25. The theme this year is "UNIX in the 90's - Decade of the User". Uniforum 1990 is offering 20 all-day tutorials, 36 marketing and industry-oriented conference sessions, eight technical presentations, daily keynote addresses, and 16 free UNIX system workshops. More than 250 major UNIX system vendors will be displaying and demonstrating their newest products and services. If anyone is interested in more information you can call me. If anyone is attending, let us know so you can tell us all about it!

This month's newsletter includes a Christmas jingle complements of a special Joke network. Just remember, Ukrainian Christmas is the first weekend in January so this article is right on time!

For those of you confused by crontabs, we have found an interesting article which discusses a way of automating the scheduling of tasks. A shell script lets the user fill-in various time, date and command queries instead of having the user know what information to put in a specific file. It also allows any user to create their own crontab commands without leaving the directory in which they are located. Don't they call that being "user-friendly"? Give the programs a try and tell us how you like them.

Now is about the time to complete my editorial and allow you to continue reading. Enjoy yourself and I'll see you next week at the meeting. Remember the location has changed to the UNIVERSITY of MANITOBA, Main Floor, Senate Chambers, Engineering Building.

Bye for Now!!!

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:	Derek Hay	943-5401
Treasurer/Membership:	Gilles Detillieux	261-9146
Secretary:	Matt Binnie	(W) 949-0190
Newsletter Editor:	Susan Zuk	(W) 788-7312
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 January meeting location will be provided by The **University of Manitoba**, Senate Chambers, Main Floor Engineering Building (Fort Garry Campus), South of University Centre.

President's Corner

by Gilbert Detillieux, President

Well, here we find ourselves not only starting a new year, but a new decade. The 1980's have certainly seen some interesting changes in the uses and popularity of UNIX.

As the decade began, I got my first exposure to UNIX, on a PDP-11/45 running Version 6 (the version before the first portable UNIX, Version 7, which started the whole "movement"). At that time, UNIX was well known and well liked in most universities, but virtually unheard of in the business world.

By the middle of the decade, UNIX was starting to catch on in business, and opportunities looked really promising for those with UNIX expertise. Our company, INFO WEST, incorporated at that time, and we decided that this expertise would be our edge, but unfortunately, in Winnipeg most business and MIS people were still asking "What's UNIX?"

As the decade draws to a close, UNIX looks healthier than ever, and is well known enough to be getting coverage in newspapers and business magazines. Where UNIX is headed in the next decade is not completely clear, but it is certainly headed up! I expect that by the mid-90's, what isn't UNIX will either be UNIX-derived (Mach, AIX, and others to follow) or UNIX-inspired (many companies are talking about adding POSIX compatibility to their non-UNIX systems). Thus the 1990's should be an interesting decade for UNIX watchers.

With that in mind, our local industry psychic, Jean U. Nixen, has come out of a coma again just long enough to make some predictions for the decade ahead. Here we go again...

1991: DEC's wish that the OSF and UNIX International will "get married soon" is realized; unfortunately, a divorce soon ensues.

1993: IBM's success with its new line of RISC processors prompts it to devise new industry-specific RISC architectures. The Financial Instruction Set Computer And Logic (FISCAL) architecture is introduced, with built-in FIFO and LIFO storage, Internal Register Store (IRS), Global Instruction Cache (GIC), and Register-Register Storage Processor (RRSP). Revenue Canada decides against FISCAL technology because Global System Traps (GST's) are not implemented.

1995: "OS/Who?"

1996: NeXT Inc. decides to stop providing discounts to academic institutions. Infuriated by that move, Steve Jobs resigns and goes on to form LaST Computer Technologies.

1999: LaST introduces it's revolutionary new system for the 2000's, which features optical computing logic, 1000 MIPS, 1500 MFLOPS, 4500 KLIPS (1000's of Logical Inferences Per Second), 2TB (Tera-Bytes) optical main memory, and 200TB removable optical storage. IBM buys them out; Steve Jobs resigns.

2000: The financial world grinds to a halt, as a quarter century's worth of computer software fails to calculate the correct date.

Now, back to the present, and some more serious matters.

Our current membership secretary, Pat Macdonald, regretfully informed us that he would be resigning his position. On behalf of all the membership, I would like to thank Pat for having done a great job for the past year; I would also like to thank our treasurer, Gilles Detillieux, for assuming the responsibilities of membership secretary for the remainder of the year.

Pat will remain a member, however, and has agreed to do a presentation for the next meeting. Thus, our January meeting will not be at the usual location, but at the University of Manitoba; specifically, the Senate Chambers, Main Floor, Engineering Building (just south of University Centre), Fort Garry Campus. This meeting will be January 9, at 7:30PM, and will follow our usual format: round table discussion, business meeting, then presented topic. The presented topic will deal with sockets and IPC for TCP/IP networks - it will be quite technical, but should be very useful for any programmers out there interested in networks.

I hope to see you all at the meeting. I wish you all success for the new year and the new decade!

Solving Those Puzzling Quirks of UNIX Systems Use

By Steven List and Bruce Stewart

Reprinted from /usr/group CommUNIXations September/October 1987

The UNIX operating system provides two facilities for scheduling activities such as performing system backups, preparing reports or building software. Both scheduling facilities use the *cron* mechanism. The *at* facility schedules a *single* occurence of any activity. For example, you can specify that you want to create a report beginning "at 10PM tonight" or "at noon on Thursday" or even "now". The system will perform the request once, at the specified time. The *crontab* facility schedules the *regular* occurence of an activity. Activities may be scheduled once a year (using a specific date), once a minute or for any interval between. Although both mechanisms keep track of scheduled activities, *crontab* is more appropriate for scheduling the same activity repeatedly, whereas *at* is the method of choice for scheduling a single occurence of an activity. The *at* facility is also appropriate for scheduling activities that occur regularly, but change significantly between executions.

But How?

This column discusses using *crontab* and *cron* to schedule activities. We will examine a shell script that uses *crontab* to schedule a regular activity. The script is suitable for non-technical users or for those with no experience with *cron* and *crontab*, as well as those users who just want a more convenient method of using the resident UNIX scheduling facilities. We will also briefly discuss the role that the system administrator must play in setting up and monitoring the *crontab* mechanism.

The Prerequisites?

Several files are required in order to use the *at* and *crontab* scheduling facilities. All are found in the directory */usr/lib/cron*. Typically, these files restrict or allow access to the *at* and *crontab* schedulers, and are controlled by the system administrator. Each file is listed and described in Figure 1.

Using Crontab

The *crontab* facility relies on a series of *crontab* files to schedule activities. A *crontab* file consists of one or more commands to be executed and the scheduling information for each command. Each user authorized to use the *crontab* facility has a *crontab* file in the /usr/lib/cron/crontab directory. Entries in the *crontab* file follow a special format, and readers are referred to the *crontab* and *cron* manual pages for a discussion of the fields in each *crontab* entry. The shell script examined here prompts the user for the values to be entered in each field, validates the input and creates a new entry in the *crontab* file.

Figure 3 contains the main logic of the shell program. The *eval* command is used to create a menu in which each choice corresponds to a field in a *crontab* entry. The user is prompted to select a field to be modified. For each field selected, the value to be placed in the field is edited by the *valnum* shell function (Figure 2). The function actually begins the shell script because shell functions must be defined before being

Figure 1 - Essential Files

at.allow	This file is a list of users permitted to use the *at* facility. Note that if there is no at.allow file and no at.deny file, only the superuser is allowed to use at. Also note that if there is only an at.deny file (see below), the manual indicates that anyone not listed in that file may use at. The implication is that if there is an empty at.deny file, anyone may use at.
at.deny	This file is a list of users who are not allowed to use the *at* facility. As indicated above, this file may be used to provide global access.
cron.allow	Similar to *at.allow*, this file is a list of users permitted to use the *cron* facility.
cron.deny	Similar to at.deny, this file denies specified users the use of the *cron* facility. This file or the *cron.allow* file, should generally be set up as there is more danger in allowing users to schedule regular events than specific events, although not too much.
.proto	This file is required and contains a prototype header file for *at* jobs.

used. If the selection is to update the *crontab* entry, the entered values are appended to a file in the */usr/spool/cron/crontabs* directory. The *crontab* command is then invoked with this file as input to create a new *crontab* entry.

Still referring to Figure 3, the *true* and *false* commands are used to control the *while* loops. Both of these commands are shell scripts and live in */bin*. *true* is a shell script with no commands and a default exit status of zero. It does nothing successfully. *false* is a shell script with an explicit exit status of 255. It does nothing unsuccessfully. Both commands are useful for controlling loops designed to repeat indefinitely.

The && list separator is also used to assign the value from false to the shell variable FAILED. The statement that executes the *valnum* function includes the construction "&&FAILED=false". The && list separator informs the shell to execute the statement following the && only if the statement preceding the separator succeeds. In this example, if the *valnum* function returns zero (success), the variable FAILED is set to false and the loop terminates.

Finally, IFS (internal field separator) is used to parse the expressions passed to the *valnum* function. The existing IFS is first saved and then set to "-,". Whether the string passed to *valnum* is a range (elements separated by a dash) or a list (elements separated by a comma), the expression can be validated against the minimum and maximum values allowable for the appropriate field. IFS is then restored to its previous value.

Suggestions and Improvements

This script is only a beginning. Although it helps users create a crontab file, it cannot be used to edit an existing crontab file. Output from the script is always appended to a crontab file; no provisions is made to change an entry once it has been written. It also doesn't offer the user a graceful way to exit the script. Once invoked, the script assumes that a user will eventually want to write a crontab file. And although the help files give some guidance for completing the scheduling information in the *crontab* entry, the contents of the command field are necessarily left to the user. Improvements in these areas would make the script both friendlier and more useful.

Quirks and Points of Interest

Our discussion of UNIX scheduling would not be complete without a brief mention of the *batch* command. As discussed in the manual page for *at*, the *batch* command is similar to "at now," but not identical. The *batch* command is actually a shell script living in */usr/bin*; it does no more than invoke *at -qb*. The "-qb" argument to *at* specifies the queue to be used for the submitted job. Presumably, the undocumented flag "-q" indicates that the letter following refers to the queue in which the submitted job is to be placed for execution.

One important note about the use of *cron*, which is also clearly documented in the manual page, is worthy of additional comment here. All output (both standard output and standard error) from commands executed by *cron* are mailed to the user. If the user wishes to have something else happen, then either or both of these outputs must be explicitly handled. Also, *even if there is no output*, the *cron* facility will create an entry in the user's mail file reflecting the time at which the entry was executed unless the user redirects the output.

Another interesting point is found in the capability to redirect *standard input* into a user's commands in *cron*. This can provide very powerful means of executing commands that are normally interactive in a batch environment..

Steven List is a senior technical staff member of Benetics Corp. Bruce Stewart is an independent software consultant experienced in program development, systems administration and system design.

Figure 2: Function valnum - validate numeric field contents

```
:
# function valnum accepts a string that may be com-
# posed of digits, commas, or dashes. since cron
# does not allow both ranges (a-z) and lists (a,b,c)
# in the same field, this function will handle the
# exclusion.
#
# usage: valnum fieldtype value
#
MIN=1 HR=2 DOM=3 MOY=4 DOW=5
#
valnum ()
{
   fieldtype=$1
   fieldval=$2
   case $fieldtype in
      $MIN)   min=0 max=59 type=Minute;;
      $HR)    min=0 max=23 type=Hour;;
      $DOM)   min=1 max=31 type="Day of Month";;
      $MOY)   min=1 max=6 type=Month;;
      $DOW)   min=0 max=6 type="Day of Week";;
   esac
   shift
   dash=`expr "$fieldval" : '.*'`
   comma=`expr "$fieldval" : '.*'`
   if [ -n "$dash" -a -n "$comma" ]
   then
      echo "You cannot have both a range and a list:"
      echo $fieldval
      ret=1
   else
      OLDIFS="$IFS" IFS="-,"
      set $fieldval
      IFS="$OLDIFS"
      ret=0
      for i in $*; do if [ $i -lt $min -o $i -gt $max ]; then
            echo "Invalid value for $type: $i"
            ret=1
         fi
      done
   fi
   return $ret
}
```

Figure 3: The MKCRON Shell Script

```
:
# mkcron - create a crontab entry
#
# usage: mkcron
#
MIN=0
HR=2 DOM=3 MOY=4 DOW=5 CMD=6 UpdCron=7
Max=8
#
#
eval Menu_$MIN=\"Minute\"
eval Menu_$HR=\"Hour\"
eval Menu_$DOM=\"Day of Month\"
eval Menu_$MOY=\"Month\"
eval Menu_$DOW=\"Day of Week\"
eval Menu_$CMD=\"Command to Execute\"
eval Menu_$UpdCron=\"Update CRONTAB with this entry\"
#
Minutes='*' Hours='*' DaysOfMonth='*' DaysOfWeek='*' Months='*'
Command=/bin/date
#
Continue=true
#
while $Continue
do
   Select=0
   #
   while [ $Select -eq 0 ]
   do
      echo "\n\tPlease select which field you would like to enter\n"
      for i in $MIN $HR $DOM $MOY $DOW $CMD $UpdCron
      do
         eval echo "\\\t$i: \$Menu_$i"
      done
      echo "\n\tPlease enter the number of your choice: _\b\c"
      read Select
      if [ "$Select" -le $Min -o "$Select" -ge $MAX ]
      then
         echo "\n--> This is not a valid choice [$Select]\n"
         Select=0
      fi
   done # while !select
   #
   # process the specific field requested by the user
   #
   if [ $Select -eq $CMD ]
   then
      eval echo "\\\tPlease enter the value for the \$Menu_$Select field: "
      read Command
   elif [ $Select -eq $UpdCron ]
   then
      if [ -r /usr/spool/cron/crontabs/$LOGNAME ]
      then
         cp /usr/spool/cron/crontabs/$LOGNAME ct.$LOGNAME
      else
         touch ct.$LOGNAME
      fi
```

```
        chmod 666 ct.$LOGNAME
        echo "$Minutes $Hours $DaysofMonth $Months $DaysOfWeek $Command" >> ct.$LOGNAME
        crontab ct.$LOGNAME
        rm -f ct.$LOGNAME
        echo "\tYour crontab thus far contains the following:\n"
        crontab -l
        Continue=false
        continue
    else
        FAILED=true
        while $FAILED
        do
            eval echo "\\\tPlease enter the value for the \$Menu_$Select field: \\\c"
            read Value
            if [ "$Value" = '?' ]         # a cry for help
            then
                echo ""
                eval more Help.$Select
                echo ""
            else
                valnum $Select "$Value" && FAILED=false
            fi
        done
        case $Select in
            $MIN)           Minutes="$Value";;
            $HR) Hours="$Value";;
            $DOM) .......... DaysOfMonth="$Value";;
            $MOY) .......... Months="$Value";;
            $DOW) .......... DaysOfWeek="$Value";;
        esac
    fi
    echo "\tEntry so far:\n\t\c"
    echo "$Minutes $Hours $DaysofMonth $Months $DaysOfWeek $Command"
done # while continue
exit 0
```

The UNIX Christmas Song

Better watchout
Better !cry
Better !pout
Lpr why
Santaclaus < Northpole > Town

Cat /etc/password > list
Ncheck list
Ncheck list
Grep list naughty > nogiftlist
Grep list nice > giftlist
Santaclaus < Northpole > Town

Who | grep sleeping
Who | grep awake
Who |grep bad/good
For (goodness sake) {be good}

Technical UNIX* User Group

Agenda

for

Tuesday, January 9, 1990

7:30pm

The University of Manitoba

Fort Garry Campus

Senate Chambers

Main Floor, Engineering Building

South of University Centre

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~~ - Pat Macdonald	
NFS Internals	
6. Adjourn	9:30