NAME

crontab − table of jobs to be performed by cron

SYNOPSIS

/usr/lib/crontab
/usr/local/lib/crontab
/var/lib/crontab
/var/opt/*name***/lib/crontab** (Minix-vmd only)
/usr/spool/crontabs/*user*

DESCRIPTION

The **cron**(8) daemon runs jobs at regular intervals. These jobs are listed in **crontab** files. The format of entries in a crontab file are five fields of numbers specifying the minute (0−59), hour (0−23), day of the month (1−31), month (1−12), and day of the week (0−6 with 0 = Sunday) that a task must be executed. The task to be executed follows as a shell command.

The time numbers can be given as a comma separated list of simple numbers, ranges ("2−5" is the same as "2,3,4,5"), and repeats ("2:5" means "2,7,12,17,22" in the hour field). A repeat is cyclic affair, i.e. 2:5 and 12:5 are the same thing. A single "*" can be used in a field to indicate all valid numbers in that field, so it translates to "always". In the minute field you can use "?" for the current minute that the crontab file is loaded. It can be used in a repeat, i.e. "?:10" for every 10 minutes. This keeps machines with identical crontabs from executing tasks at exactly the same time, causing a burst of traffic if anything is done over a network.

If a given time is valid in all five fields then a command is executed. Here are a few examples that illustrate the possibilities:

```
# min hour mday mon wday command
   ?   3    *    *   *   /usr/etc/daily  # Daily system cleanup
   0   *    *    *   *   date            # Print date on the hour
  30   4    *    *  2−6  /var/etc/backup # After workdays on 4:30
   0   9   25   12   *   −u ast sing     # Andy sings on Xmas morning
   0   0   13    *   5   echo Beware!    # For the superstitious
```

The command may optionally be prefixed by **−u** *user* to specify under which user the command should be run. Commands from crontabs in the spool directory are always run under the id of the crontab's owner, the **−u** flag is ignored.

A command can be placed on the same line as the time fields, or on the next line indented by one TAB character. (A TAB, not eight spaces.) More TAB indented lines can be added for a multiline command. The tabs are removed from the command when passed to the shell. If a command is put on the same line as the time fields then percent characters are changed into newlines, this is not done for a TAB indented command. The following three entries give the same output:

```
0 12  *  *  *   echo ’Hello’; echo ’  World!’
#1
0 12  *  *  *   echo ’Hello%  World!’  #2
0 12  *  *  *        #3
        cat <<EOF    #4
        Hello
          World!
        EOF
```

Comments start with a "#" character and continue until end of line. They, excess whitespace, and empty lines are ignored. Of the comments in the example above #1 and #3 are ignored by **cron**, but #2 and #4 are not recognized as comments, but are seen as part of a command and are passed to the shell who then happens to ignore them. There is no interpretation of command characters other than the percent in a oneliner. The time fields must all be on the same line.

FILES

/usr/lib/crontab	Main Minix crontab file.
/usr/local/lib/crontab	Local jobs for all systems in an organization.
/var/lib/crontab	System specific jobs.
/var/opt/*name***/lib/crontab**	Per package jobs for Minix-vmd.
/usr/lib/packages	List of installed packages.
/usr/spool/crontabs/*user*	Per user jobs.

SEE ALSO

crontab(1), **cron**(8).

NOTES

The "?" in the minute field, the repeat field (e.g. "2:5"), TAB indented multiline commands and the **−u** option are unique to this cron implementation. This doesn't mean you shouldn't use these features, but just that you should be aware of the differences with other systems. You are even advised to use these features and avoid the percent hack for multiline commands.

Other crons allow one to specify input to a job in some way, something this cron can't. Simply use the << shell feature to do that. Other crons often choke on empty lines.

It is a common bug to use 0 for Sunday instead of 7. This cron, like most other crons out there accepts this without comment.

A job is not reissued until a previous instance of it has exited. The next time to execute is computed from the previous time it ran. If job issuing lags behind on the system time then the next time to run it is computed from the current system time.

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)