NAME

crontab - tables for driving cron

DESCRIPTION

A **crontab** file contains instructions to the cron(8) daemon of the general form: ‘‘run this command at this time on this date’’. Each user has their own crontab, and commands in any given crontab will be executed as the user who owns the crontab. Uucp and News will usually have their own crontabs, eliminating the need for explicitly running su(1) as part of a cron command.

Blank lines and leading spaces and tabs are ignored. Lines whose first non-space character is a pound-sign (#) are comments, and are ignored. Note that comments are not allowed on the same line as cron commands, since they will be taken to be part of the command. Similarly, comments are not allowed on the same line as environment variable settings.

An active line in a crontab will be either an environment setting or a cron command. An environment setting is of the form,

```
name = value
```

where the spaces around the equal-sign (=) are optional, and any subsequent non-leading spaces in *value* will be part of the value assigned to *name*. The *value* string may be placed in quotes (single or double, but matching) to preserve leading or trailing blanks. The *name* string may also be placed in quote (single or double, but matching) to preserve leading, trailing or inner blanks.

Several environment variables are set up automatically by the cron(8) daemon. SHELL is set to */bin/sh*, and LOGNAME and HOME are set from the */etc/passwd* line of the crontab’s owner. In addition, the environment variables of the user’s login class will be set from */etc/login.conf.db* and *~/.login_conf*. (A setting of HOME in the login class will override the value from */etc/passwd*, but will not change the current directory when the command is invoked, which can only be overridden with an explicit setting of HOME within the crontab file itself.) If PATH is not set by any other means, it is defaulted to */sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin*. HOME, PATH and SHELL, and any variables set from the login class, may be overridden by settings in the crontab; LOGNAME may not.

(Another note: the LOGNAME variable is sometimes called USER on BSD systems... On these systems, USER will be set also).

If cron(8) has any reason to send mail as a result of running commands in ‘‘this’’ crontab, it will respect the following settings which may be defined in the crontab (but which are not taken from the login class). If MAILTO is defined (and non-empty), mail is sent to the user so named. If MAILFROM is defined (and non-empty), its value will be used as the from address. MAILTO may also be used to

direct mail to multiple recipients by separating recipient users with a comma. If MAILTO is defined but empty (MAILTO=""), no mail will be sent. Otherwise mail is sent to the owner of the crontab. This option is useful if you decide on */bin/mail* instead of */usr/lib/sendmail* as your mailer when you install cron -- */bin/mail* does not do aliasing, and UUCP usually does not read its mail.

The format of a cron command is very much the V7 standard, with a number of upward-compatible extensions. Each line has five time and date fields, followed by a user name (with optional ‘‘:<group>’’ and ‘‘/<login-class>’’ suffixes) if this is the system crontab file, followed by a command. Commands are executed by cron(8) when the minute, hour, and month of year fields match the current time, *and* when at least one of the two day fields (day of month, or day of week) matches the current time (see ‘‘Note’’ below). cron(8) examines cron entries once every minute. The time and date fields are:

field	allowed values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or use names)

A field may be an asterisk (*), which always stands for ‘‘first-last’’.

Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an ‘‘hours’’ entry specifies execution at hours 8, 9, 10 and 11.

Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: ‘‘1,2,5,9’’, ‘‘0-4,8-12’’.

Step values can be used in conjunction with ranges. Following a range with ‘‘/<number>’’ specifies skips of the number’s value through the range. For example, ‘‘0-23/2’’ can be used in the hours field to specify command execution every other hour (the alternative in the V7 standard is ‘‘0,2,4,6,8,10,12,14,16,18,20,22’’). Steps are also permitted after an asterisk, so if you want to say ‘‘every two hours’’, just use ‘‘*/2’’.

Names can also be used for the ‘‘month’’ and ‘‘day of week’’ fields. Use the first three letters of the particular day or month (case does not matter). Ranges and lists are also allowed.

The ‘‘sixth’’ field (the rest of the line) specifies the command to be run. One or more command options may precede the command to modify processing behavior. The entire command portion of the line, up to a newline or % character, will be executed by */bin/sh* or by the shell specified in the SHELL variable

of the cronfile. Percent-signs (%) in the command, unless escaped with backslash (\), will be changed into newline characters, and all data after the first % will be sent to the command as standard input.

The following command options can be supplied:

-n No mail is sent after a successful run. The execution output will only be mailed if the command exits with a non-zero exit code. The **-n** option is an attempt to cure potentially copious volumes of mail coming from cron(8).

-q Execution will not be logged.

Duplicate options are not allowed.

Note: The day of a command's execution can be specified by two fields -- day of month, and day of week. If both fields are restricted (ie, are not *), the command will be run when *either* field matches the current time. For example, ''30 4 1,15 * 5'' would cause a command to be run at 4:30 am on the 1st and 15th of each month, plus every Friday.

Instead of the first five fields, a line may start with '@' symbol followed either by one of eight special strings or by a numeric value. The recognized special strings are:

string	meaning
@reboot	Run once, at startup of cron.
@yearly	Run once a year, "0 0 1 1 *".
@annually	(same as @yearly)
@monthly	Run once a month, "0 0 1 * *".
@weekly	Run once a week, "0 0 * * 0".
@daily	Run once a day, "0 0 * * *".
@midnight	(same as @daily)
@hourly	Run once an hour, "0 * * * *".
@every_minute	Run once a minute, "*/1 * * * *".
@every_second	Run once a second.

The '@' symbol followed by a numeric value has a special notion of running a job that many seconds after completion of the previous invocation of the job. Unlike regular syntax, it guarantees not to overlap two or more invocations of the same job during normal cron execution. Note, however, that overlap may occur if the job is running when the file containing the job is modified and subsequently reloaded. The first run is scheduled for the specified number of seconds after cron is started or the crontab entry is reloaded.

EXAMPLE CRON FILE

```
# use /bin/sh to run commands, overriding the default set by cron
SHELL=/bin/sh
# mail any output to ‘paul’, no matter whose crontab this is
MAILTO=paul
#
# run five minutes after midnight, every day
5 0 * * *       $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 2:15pm on the first of every month -- output mailed to paul
15 14 1 * *     $HOME/bin/monthly
# run at 10 pm on weekdays, annoy Joe
0 22 * * 1-5            mail -s "It’s 10pm" joe%Joe,%%Where are your kids?%
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"
5 4 * * sun     echo "run at 5 after 4 every sunday"
# run at 5 minutes intervals, no matter how long it takes
@300                    svnlite up /usr/src
# run every minute, suppress logging
* * * * *       -q date
# run every minute, only send mail if ping fails
* * * * *       -n ping -c 1 freebsd.org
```

SEE ALSO

crontab(1), cron(8)

EXTENSIONS

When specifying day of week, both day 0 and day 7 will be considered Sunday. BSD and ATT seem to disagree about this.

Lists and ranges are allowed to co-exist in the same field. "1-3,7-9" would be rejected by ATT or BSD cron -- they want to see "1-3" or "7,8,9" ONLY.

Ranges can include "steps", so "1-9/2" is the same as "1,3,5,7,9".

Names of months or days of the week can be specified by name.

Environment variables can be set in the crontab. In BSD or ATT, the environment handed to child processes is basically the one from */etc/rc*.

Command output is mailed to the crontab owner (BSD cannot do this), can be mailed to a person other

than the crontab owner (SysV cannot do this), or the feature can be turned off and no mail will be sent at all (SysV cannot do this either).

All of the '@' directives that can appear in place of the first five fields are extensions.

Command processing can be modified using command options. The '-q' option suppresses logging. The '-n' option does not mail on successful run.

AUTHORS

Paul Vixie <*paul@vix.com*>

BUGS

If you are in one of the 70-odd countries that observe Daylight Savings Time, jobs scheduled during the rollback or advance may be affected if cron(8) is not started with the **-s** flag. In general, it is not a good idea to schedule jobs during this period if cron(8) is not started with the **-s** flag, which is enabled by default. See cron(8) for more details.

For US timezones (except parts of AZ and HI) the time shift occurs at 2AM local time. For others, the output of the zdump(8) program's verbose (**-v**) option can be used to determine the moment of time shift.