at

Schedule a command to run once at a particular time, examine or delete jobs: at, batch, atq, atrm

```
Syntax
      at [-V] [-q queue] [-f file] [-mldbv] TIME          # Execute commands at a specified time.
      at [-V] [-q queue] [-f file] [-mldbv] -t time_arg   # Execute commands at a specified time.
      at -c job [job...]
      atq [-V] [-q queue]        # List the user's pending jobs, or for the superuser; list all jobs.
      atrm [-V] job [job...]     # Delete jobs, identified by their job number.
      batch                      # Execute commands when system load levels permit.

Key
   -c     Cat the jobs listed on the command line to standard output.

   -d     Is an alias for atrm.

   -f file
          Read the job from file rather than standard input.

   -l     Is an alias for atq.

   -m     Send mail to the user when the job has completed even if there was no output.

   -q queue
          use the specified queue.
          A queue designation consists of a single letter; valid queue designations range from a to z.
          and A to Z. The a queue is the default for at and the b queue for batch.
          Queues with higher letters run with increased niceness.
          The special queue "=" is reserved for jobs which are currently running. If a job is submitted to
          designated with an uppercase letter, the job is treated as if it were submitted to batch at the t
          the job. Once the time is reached, the batch processing rules with respect to load average apply.
          If atq is given a specific queue, it will only show jobs pending in that queue.

   -t time_arg
          Submit the job to be run at the time specified by the time_arg option argument, which must have
          the same format as specified for the touch utility's -t time option argument ([[CC]YY]MMDDhhmm).

   -v     Show the time the job will be executed before reading the job.
          Times displayed will be in the format "Thu Feb 20 14:50:00 1997".

   -V     Print the version number to standard error.
```

`at` and `batch` read commands from standard input or a specified file which are to be executed at a later time.

`batch` executes commands when system load levels permit; in other words, when the load average drops below 0.8, or the value specified in the invocation of atd.

`at` allows fairly complex time specifications, extending the POSIX.2 standard. It accepts times of the form *HH*:*MM* to run a job at a specific time of day. (If that time is already past, the next day is assumed.) You may also specify midnight, noon, or teatime (4pm) and you can have a time-of-day suffixed with AM or PM for running in the morning or the evening. You can also say what day the job will be run, by giving a date in the form month-name day with an optional year, or giving a date of the form *MMDDYY* or *MM*/*DD*/*YY* or *DD.MM.YY* or *YYYY-MM-DD*. The specification of a date must follow the specification of the time of day. You can also give times like now + count time-units, where the time-units can be minutes, hours, days, or weeks and you can tell at to run the job today by suffixing the time with today and to run the job tomorrow by suffixing the time with tomorrow.

For example, to run a job at 4pm three days from now: `at 4pm + 3 days`,
to run a job at 10:00am on July 31: `at 10am Jul 31`
and to run a job at 1am tomorrow: `at 1am tomorrow`.
The exact definition of the time specification can be found in /usr/share/doc/at-3.1.10/timespec.

For both at and batch, commands are read from standard input or the file specified with the -f option and executed. The working directory, the environment (except for the variables TERM, DISPLAY and _) and the umask are retained from the time of invocation. An at - or batch - command invoked from a su(1) shell will retain the current userid. The user will be mailed standard error and standard output from his commands, if any. Mail will be sent using the command /usr/sbin/sendmail. If at is executed from a su(1) shell, the owner of the login shell will receive the mail.

Installation

Most distributions include `at` by default, if you do need to install it use the package manager for your distribution e.g for apt.

```
$ sudo apt update
$ sudo apt install at
```
or
```
$sudo apt-get install at
```

Permissions

The superuser may use these commands in any case. For other users, permission to use at is determined by the files /etc/at.allow and /etc/at.deny.

If the file /etc/at.allow exists, only usernames mentioned in it are allowed to use at.
If /etc/at.allow does not exist, /etc/at.deny is checked, every username not mentioned in it is then allowed to use at.
If neither exists, only the superuser is allowed use of at.

An empty /etc/at.deny means that every user is allowed use these commands, this is the default configuration.

Environment

`SHELL` The value of the SHELL environment variable at the time of at invocation will determine which shell is used to execute the at job commands. If SHELL is unset when at is invoked, the user's login shell will be used; otherwise, if SHELL is set when at is invoked, it must contain the path of a shell interpreter executable that will be used to run the commands at the specified time.

at will record the values of environment variables present at time of at invocation. When the commands are run at the specified time, at will restore these variables to their recorded values . These variables are excluded from this processing and are never set by at when the commands are run : TERM, DISPLAY, SHELLOPTS, _, PPID, BASH_VERSINFO, EUID, UID, GROUPS. If the user submitting the at job is not the super-user, variables that alter the behaviour of the loader ld.so(8), such as LD_LIBRARY_PATH, cannot be recorded and restored by at.

Files

```
/var/spool/at
/var/spool/at/spool
/proc/loadavg
/var/run/utmp
/etc/at.allow
/etc/at.deny
```

Examples

Run the `export1.sh` script at 11pm:

```
$ echo "/home/scripts/export1.sh" | at 23:00
```

In the example above we are piping the command required to `at`, without that `at` will prompt with an `at>` prompt where you can interactively enter the command required, pressing Ctrl-D to exit when done.

We can do the same thing using a Here document:

```
$ at 23:00 <<SS64
"/home/scripts/export1.sh"
SS64
```

The job to be run can also be stored in a file and executed with -f, if the file contains multiple lines/commands then each will be launched as a separate job:

```
$ echo "/home/scripts/export1.sh" >/home/scripts/myjob.sh
$ at 23:00 -f /home/scripts/myjob.sh
```

“When the Last of the Great Auks Died, It Was by the Crush of a Fisherman’s Boot” ~

Related linux commands

cron - Daemon to execute scheduled commands.
crontab - Schedule a command to run at a later time.
Equivalent Windows command: SCHTASKS - Manage scheduled tasks.