Manage Jobs with cron and at

As a SLES 11 administrator, you will find that there are many tasks that need to be carried out on a regular basis on your Linux system. For example, you may need to update a database or back up users' data in the `/home` directory. While you could run these tasks manually, it would be more efficient (and more reliable) if you were to configure the Linux system to run them automatically for you.

You need to know how to

- "Schedule Jobs with cron" on page 385
- "Schedule Jobs with at" on page 390

Schedule Jobs with cron

One option for doing this is to use cron. The cron daemon allows you to schedule jobs that will be carried out for you on a regular schedule.

In this objective, the following topics are addressed:

- "crontab File Syntax" on page 385
- "Defining System Jobs" on page 386
- "Defining User Jobs" on page 389

crontab File Syntax

The cron daemon is activated by default on SLES 11. Once a minute, it checks to see if any jobs have been defined for the current time.

The cron daemon uses files called crontabs. These files list jobs and when they are to be run. Several crontab files exist on a Linux system. They are used to run system jobs. Users on the system can also define their own crontab files.

Each line in a crontab file defines a single cron job. There are six or seven fields in each line, separated by whitespace characters (spaces or tabs). The first five fields define when the cron job should be run. The sixth field in the line specifies the command to be run. If system jobs are defined, the sixth field contains the name of user to be used to run the job, and the seventh field specifies the command to be run.

The cron daemon can run any command or shell script. However, no user interaction is available when the command or shell script is run.

The first five fields in the crontab file use the following syntax:

Table 5-9 *Crontab Time Fields*

Field Number	Field Label	Range
1	Minutes	0–59
2	Hours	0–23
3	Day of the Month	1–31
4	Month	1–12
5	Weekday	0–7 (0 and 7 represent Sunday)

The following are guidelines for configuring these fields:

- If you want a job to run every minute, hour, day, or month, type an asterisk (*****) in the corresponding field.
- You can include several entries in a field in a list separated by commas.
- You can specify a range with start and end values separated by a hyphen.
- You can configure time steps with ***/n*** (where ***n*** represents the size of the step).
- You can specify months and weekdays using the first three letters of their names (for example, MON, TUE, JAN, FEB). The letters are not case sensitive. However, when you use letters, you cannot use ranges or lists.
- Numbers representing the weekdays start at **0** for Sunday and run through the entire week consecutively, with **7** representing Sunday again.

  For example, **3** is Wednesday and **6** is Saturday.
- If the first field starts with a **-** character and the job belongs to root, no log entry is written to the `/var/log/messages` file

The following is an example of a cron job entry:

```
*/10 8-17 * * 1-5 fetchmail mailserver
```

In this example, every 10 minutes (***/10**) between 8:00 AM and 5:50 PM (**8-17**), from Monday to Friday (**1-5**) the `fetchmail` command is run to fetch incoming emails from the **mailserver** server.

For system jobs, the user who has the permissions to run the command must also be specified. Enter the username between the time definition (the first five fields) and the name of the command (which now becomes the seventh field).

Defining System Jobs

The cron daemon can be configured to run scheduled system jobs. You can define system jobs in the `/etc/crontab` file. This file is shown below:

```
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
#
# check scripts in cron.hourly, cron.daily, cron.weekly, and
cron.monthly
#
-*/15 * * * *   root  test -x /usr/lib/cron/run-crons && /usr/lib/
cron/run-crons >/dev/null 2>&1
```

The job pre-defined in `/etc/crontab` runs the scripts contained in the following directories at the intervals indicated:

Table 5-10 *The /etc/cron.**time-interval** Directories*

Directory	**Interval**
/etc/cron.hourly	Jobs that run on an hourly basis.
/etc/cron.daily	Jobs that run on a daily basis.
/etc/cron.weekly	Jobs that run on a weekly basis.
/etc/cron.monthly	Jobs that run on a monthly basis.

NOTE: While you can add lines to `/etc/crontab`, you should not delete the default lines.

NOTE: For a detailed description of the syntax for `/etc/crontab`, enter `man 5 crontab` at the shell prompt.

In the default configuration, only the `/etc/cron.daily/` directory contains scripts, as shown below:

```
da1:~ # ls -l /etc/cron.*ly
/etc/cron.daily:
total 32
-rwxr-xr-x 1 root root  587 Feb 20 19:50 logrotate
-rwxr--r-- 1 root root  948 Feb 20 23:01 suse-clean_catman
-rwxr--r-- 1 root root 1693 Feb 20 23:01 suse-do_mandb
-rwxr-xr-x 1 root root 1875 Sep  1  2003 suse.de-backup-rc.config
-rwxr-xr-x 1 root root 2059 Sep  8  2003 suse.de-backup-rpmdb
-rwxr-xr-x 1 root root  566 Jul 23  2004 suse.de-check-battery
-rwxr-xr-x 1 root root 1314 Jul 27  2005 suse.de-clean-tmp
-rwxr-xr-x 1 root root  371 Sep  1  2003 suse.de-cron-local

/etc/cron.hourly:
total 0

/etc/cron.monthly:
total 0

/etc/cron.weekly:
total 0
```

These shell scripts may get overwritten if you update your system. Any modifications you made to these files will be lost if these files are updated. Therefore, we recommend that you add your own customized scripts to `/root/bin/cron.daily.local` (which is sourced by `/etc/cron.daily/suse.de-cron.local`) because this file is not overwritten when you update your system.

The `/usr/lib/cron/cron-runs` script called from the `/etc/crontab` file not only ensures that the scripts in the `/etc/cron.`***`time-interval`*** directories are run at the prescribed intervals, but also that jobs are run later if they cannot be run at the specified time.

For example, if a script could not be run because the computer was turned off at the scheduled time, the script is automatically run later using the settings in `/etc/crontab`. This is true only for jobs defined in `cron.hourly`, `cron.daily`, `cron.weekly`, or `cron.monthly`.

The information about the last time jobs were run is provided by the existence of the corresponding file in the `/var/spool/cron/lastrun/` directory. The time stamp of the empty files (e.g. `cron.daily`) in this directory is the time the script was run.

To add a system cron job, complete the following:

1. Open a terminal session and switch to your root user account.
2. Open `/etc/crontab` in a text editor.
3. Scroll to the bottom of the file and insert your cron job.

   For example, suppose you wanted to regularly update a database on your system using a script in `/usr/local/bin/` named `updb`. You need this script to be run every hour from 8:00 AM to 6:00 PM, Monday through Friday. You would add the following line to the file:

```
0 8-18 * * 1-5 root /usr/local/bin/updb
```

4. Save your changes to the file and exit the text editor.

In addition to putting cron jobs directly in `/etc/crontab`, you can also create individual crontab files for system jobs in the `/etc/cron.d/` directory. These files must use the same syntax format as `/etc/crontab`. However, be aware that jobs defined in files in `/etc/cron.d` are *not* run automatically at a later time if they can't be run at their scheduled time.

Defining User Jobs

In addition to system jobs, the cron daemon can also run jobs for individual users. Users can define their own crontab files in the `/var/spool/cron/tabs/` directory. These crontab files contain a schedule of jobs each user wants run. These files always belong to the root user.

Users create and maintain their own crontab files using the `crontab` command. The following options can be used with the crontab command:

Table 5-11 *crontab Command Options*

Option	Description
crontab -e	Creates or edits jobs. The vi editor is used.
crontab *file*	Replaces any existing crontab file for the current user with the specified *file*.
crontab -l	Displays current jobs.
crontab -r	Deletes all jobs.

For example, suppose you want to define a cron job that copies the contents of your user's home directory to an external USB drive mounted in `/media/USB.001` every night at 5:05 PM Monday to Friday. You would need to do the following:

1. Open a terminal session.
2. At the shell prompt, enter `crontab -e`.

   The existing crontab file is opened in the vi editor, or, if no crontab file exists yet, a blank crontab file is created for your user using the vi editor.
3. Press `i`, then enter the following:

```
5 17 * * 1-5 cp -r ~/* /media/USB.001
```

4. Press Esc, then enter `:wq`.

You can specify which users are allowed to create cron jobs and which aren't by creating the following two files:

- **/etc/cron.allow.** Users listed in this file can create cron jobs.
- **/etc/cron.deny.** Users who are *not* listed in this file can create cron jobs.

By default, the /etc/cron.deny file already exists with its own entries, including the following:

- guest
- gast

If the /etc/cron.allow file exists, it is the only file evaluated; /etc/cron.deny will be ignored in this situation. If neither of these files exists, only the root user is allowed to define user cron jobs.

Schedule Jobs with at

If you want to schedule a job to run *one* time only in the future (instead of scheduling it on a regular basis with cron) you can use the at command. To use at, you must first verify that the at package has been installed and that the atd service has been started.

You define an at job at the command prompt by entering at ***launch_time***, where ***launch_time*** is the time (12:34, for example) when you want the job to begin.

Then you enter the commands you want at to run one line at a time at the **at>** prompt. When you finish entering commands, you save the job by pressing Ctrl+d.

The following is an example of creating a job with the at command:

```
geeko@da1:~> at 21:00
warning: commands will be executed using /bin/sh
at> /home/geeko/bin/doit
at> mail -s "Results file of geeko" geeko@da1 < /home/geeko/results
at> <EOT>
job 4 at 2009-08-27 21:00
```

You can also enter the commands you want to be executed by at in a text file. If you do this, then you need to enter at -f ***file launch_time*** at the shell prompt, where ***file*** is the path and file name of the file.

The following table lists some other commonly used at commands and options:

Table 5-12 *The atq and atrm Commands*

Command	Description
atq	Displays day, time and username for scheduled jobs (including job numbers, which are needed to delete a job)
atrm *job_number*	Deletes a job (using the job number)

As with cron, you can restrict access to the atd daemon. Two files determine which users can run the at command:

- **/etc/at.allow.** Users entered in this file can define jobs.
- **/etc/at.deny.** Users who are **not** listed in this file can define jobs.

These files are text files you can modify or create. By default, the `/etc/at.deny` file already exists with its own entries, as shown below:

```
da1:~ # cat /etc/at.deny

alias
backup
bin
daemon
ftp
games

...
```

If the `/etc/at.allow` file exists, only this file is evaluated. If neither of these files exists, only the user root can define at jobs.