Timed Commands

CIS 2235 Linux System Administration

Running Commands in the Future

There is always a need to schedule commands to run later. There are three types:

- One-off commands:
  - "At 10:00 tomorrow, e-mail me this reminder message."
  - These are known as `at` commands.
- Exactly repeating commands:
  - "At 10 past midnight, rebuild this database."
  - These are known as `cron` jobs.
- "Close enough" repeating commands:
  - "Sometime today, refresh locate database."
  - These use the `anacron` system.

at Commands

`at` commands are defined using at:

```
$ at 16:30   # Flexible time specs. See man page.
at> ps -ef > ~/processes.txt
at> ^D
```

The parameter to `at` is the time the command should run.

`at` then prompts for the command itself.

- Command(s) exactly as they would be typed in the shell.
- Press Ctrl+D to finish.

The `at` dæmon will run the command at the specified time.

- In this example, the output of running ps -ef at 4:30 pm will be saved in the file ~/processes.txt

Details of Commands Run by at

A command executed by the at dæmon:

- Has the permissions of its owner.
- Runs in the directory it was set up.
- Has the environment in which it was set up (approximately).
- Does not run in a terminal.

Output from the command:

- Cannot be included in a terminal window.
- Will be mailed to its owner (or thrown away if there is no mailer).

Usually the output is explicitly redirected to a file.

at Execution

A command may be specified on standard input (stdin) instead of interactively.

From a file:

```
$ at 16:30 < monitor_processes.sh
```

The commands contained in the file monitor_processes.sh are run at 16:30

From stdin stream:

```
$ echo “ps -ef > psef.txt” | at 12:00
```

at Date & Time Specification

Unadorned times are in the next 24 hours:

```
$ at 09:30
```

Tomorrow can be specified explicitly:

```
$ at 17:00 tomorrow
```

A specific date can be used:

```
$ at 11:00 Nov 11
$ at 00:30 16.04.06
```

Relative times can be specified in minutes, hours, days, or weeks:

```
$ at now + 45 minutes
$ at 16:00 + 3 days
```

Managing at Commands

`atq` lists any pending at commands:

```
$ atq
12   Wed Nov  2 10:57:00 2011 a steve
14   Wed Nov  2 10:59:00 2011 a steve
15   Wed Nov  2 11:00:00 2011 a steve
16   Wed Nov  2 11:01:00 2011 a steve
```

The number at the start of each line identifies that `at` command/job

A particular at command can be displayed with -c option:

```
$ at -c 1
```

Remove an at command with atrm:

```
$ atrm 2
```

cron

https://help.ubuntu.com/community/CronHowto

"run a command at this exact (repeating) time"

If the exact time is missed, then the command doesn't run.

User-based → each user has their own cron schedule

The schedules are files stored in /var/spool/cron

- Each file is the user's name (id)
- Unix runs the command at the proper time as that user

These cron schedule files are called crontabs

User Crontabs

We don't want users editing in /var/spool/cron and then having to re-sync (HUP) the cron daemon.

Therefore, the crontab command is used by the user

- `$ crontab -e` to edit the crontab
  - The editor in the `$EDITOR` variable is invoked for this.
- `$ crontab -l` to display (list) the crontab
- `$ crontab -r` to remove the crontab

Note: the word "crontab" can refer to:

- The cron 'service'
- The schedule file (one for each user)
- The command which edits the user crontab file

crontab Format

Blank lines are ignored.

Comments are lines starting with a hash (#).

- Note: cron does not run a 'login' shell
  - .profile & .bash_profile are not executed
  - Environment variables can be set right in the crontab file
    - PATH=/usr/local/bin

standard Unix crontab has five columns and a command:

```
# m h dom mon dow command
 30 9 *   *   *   /usr/local/bin/check_logins -l
```

The command is run when the fields match the current time

crontab Numeric Formats

Each numeric column uses the following format:

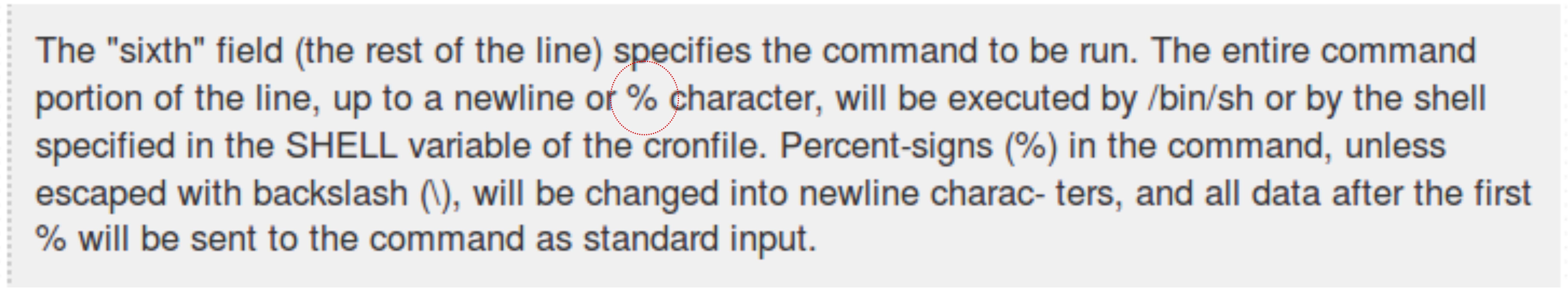
- Single number
- Star (*) → matches everything wildcard
- Range: 2 integers and a dash: 9-17 for 9am-5pm
- A range followed by a slash and a step value:
  1-30/2 for every other day
- Comma separated list: 6,8,12,15

For Linux: three-letter abbreviations can be used for month and day names:

  Mon, Tue...

  Jan, Feb,...

The 6th Column


The "sixth" field (the rest of the line) specifies the command to be run. 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 charac- ters, and all data after the first % will be sent to the command as standard input.

```
# m   h  dom  mon  dow   command
*/2   *   *    *    *    echo $(date +"\%F:\%T")  >> /home/fred/crontab.txt
```

crontab Examples

A list of multiple values for a field are specified by commas:

```
# Run at :15 and :45 past each hour:
15,45 * * * * /usr/local/bin/generate-stats-page
```

A range is specified with a hyphen:

```
# Run every half hour 09:15am-5:45pm Mon-Fri:
15,45 9-17 * * 1-5 /usr/local/bin/check-faxes
```

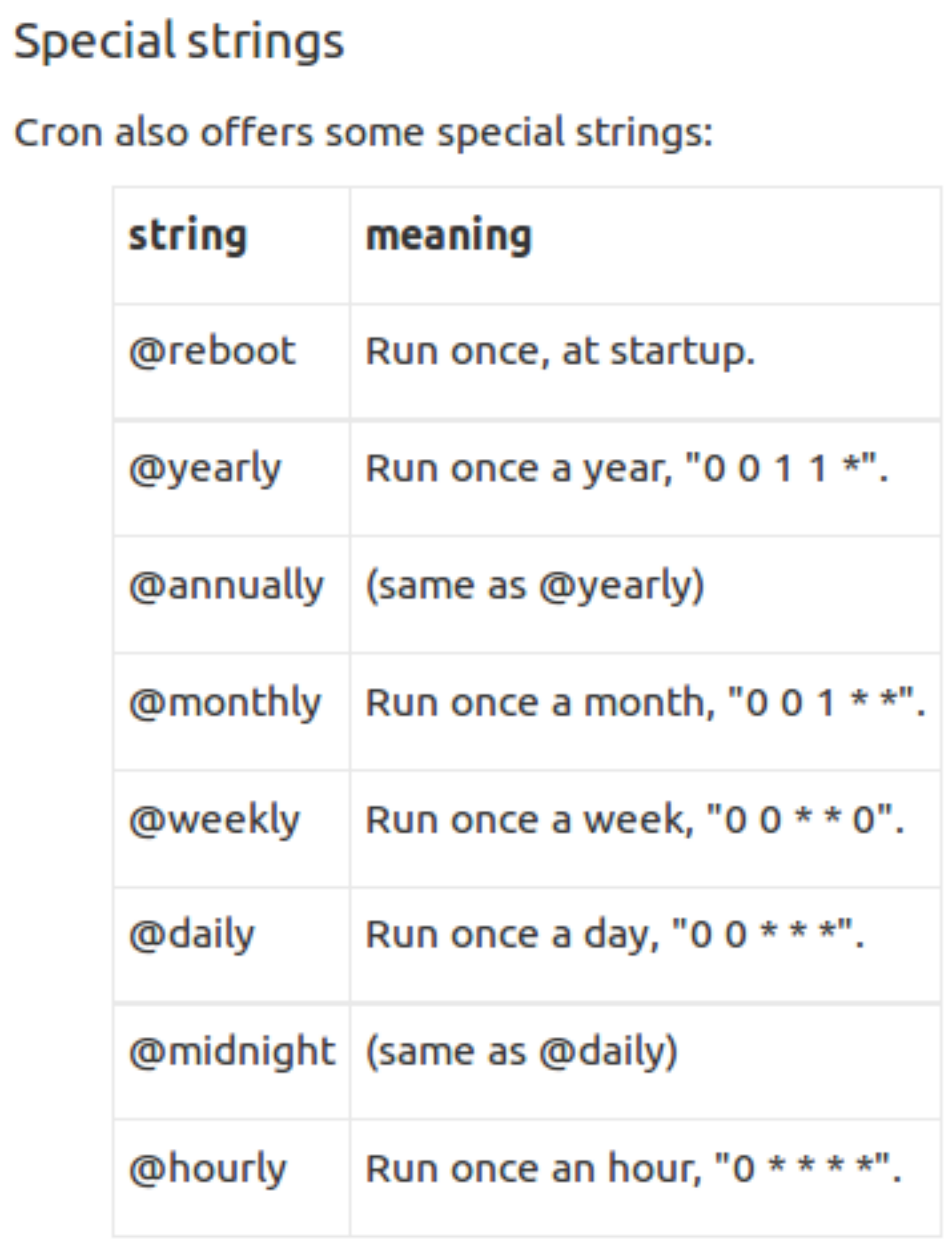
Numbers, not names, must be used for months and days in lists and ranges

i.e NO "Mon-Fri", only "1-5"

A step through a range is specified with a slash:

```
# Run every two hours 08:30am-6:30pm Mon-Fri:
30 8-18/2 * * 1-5 /usr/local/bin/check-faxes
```

Linux: a few special crontab strings



Special strings

Cron also offers some special strings:

string	meaning
@reboot	Run once, at startup.
@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 * * * *".

Vixie-cron

The "original" Unix cron service only had crontab's one-for-each-user
It was replaced with "Vixie-cron" (after author Paul Vixie)
2 new features:

1. The /etc/crontab file for system admin things
2. The /etc/cron.d dir for any file (usually from an app)

These files are run and owned by root
They look like the user's crontab, but they have an additional column – the 6th column is the user to run as

```
# Run every two hours 08:30am-6:30pm Mon-Fri:
30 8-18/2 * * 1-5 root /usr/local/bin/check-faxes
```

Cron Job Output

Cron jobs do not run in a terminal window

Generally they are administrative tasks designed not to produce any output when run successfully

Any output that is generated by a cron job is mailed:

- The recipient can be specified in the $MAILTO environment variable
- Otherwise mail is sent to the job's owner

at Command and cron Permissions

- Non-root users can be prohibited from having crontabs
  - If `/etc/cron.allow` exists then only users listed in it may have a crontab
  - file `/etc/cron.deny` blocks users from running cron
  - If *neither* file exist, then all users may have crontabs
- Permissions for running at commands are similar:
  - The files `/etc/at.allow` and `/etc/at.deny` are analogous

anacron

Problem to solve:

Computers not "on" all the time (especially laptops)

Cron jobs can be skipped because keep missing the exact time

Solution:

Anacron is designed to run a command "sometime within a given time period (hour, day, week, month)

Anacron uses an "appropriate directory" format

```
$ ls -ld /etc/cron*
```

`/etc/cron.daily` is for jobs to be run daily, etc. etc.

How Does It Work?

Several times a day, anacron checks a log to see if it has run yet for each time period: *daily, weekly, monthly*

- If it has already run, then it goes back to sleep.
- If not, then it decides to run for that time period

Anacron is set up and run by root

When it's time to run, the files are shell scripts which are run by `$ run-parts`

- `$ man run-parts` to learn more

Be careful about the names of the files in the cron directory — run-parts has some strange rules. Cron doesn't pass any flags to run-parts, so be sure you understand the rules — no extensions, so no myscript.sh

Testing anacron file names

If in doubt, test your file names:

```
sudo run-parts --report --test /etc/cron.hourly
```