Lecture: Automate and Schedule SystemAdministration Tasks

1) **`at`** Commands
   - **at** commands are defined using **`at <hour>`**

```
$ at 16:30
at> pstree > processes
at> <EOT>
```

   - The time the command should run is given as a parameter to **at**
   - **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 **`pstree`** at 16:30 will be saved in the file *processes*

- **at** Command Specification
  - First, add to your existing **ecscript** the following line:

```
date > /home/student/date_log.txt
```

  - A command may be specified on standard input instead of interactively from a file:

```
$ at 16:30 < /usr/games/ecscript
```

  - The commands contained in the file ecscript are run at 16:30. You can check your home directory to see if date_log.txt exist, and what it contains.

- Managing **at** Commands
  - **`atq`** lists any pending **at** commands:
  -

```
$ atq
38 2002-01-16 11:00 a
```

  - The number at the start of each line identifies that **at** command:
  - A particular **at** command can be displayed with **`at -c`**:

```
$ at -c <number>
```

- Remove an at command with **`atrm`**:

```
$ atrm 38
```

- Crontab.Crontab format

  - Crontab file is found in /etc/crontab. Open the file and check it's format:
    - Blank lines are ignored

- Comments are lines starting with a hash (#)
- Environment variables can be set:

PATH=/usr/local/bin

- Example cron job specification

```
30 9 * * * root /usr/local/bin/check_logins
```

- At 09:30
- On all days
- For the root user
- Run the command /usr/local/bin/check_logins

- Crontab Date & Time Specification
  - Order of the date and time fields:
    - Minute (0–59)
    - Hour (0–23)
    - Day of the month (1–31)
    - Month (1–12)
    - Day of the week (0–7; 0 and 7 are Sunday)

Note: Fields *almost* in ascending order.

  - The command is run when the fields match the current time
  - A field containing an asterisk (*) always matches
  - Three-letter abbreviations can be used for month and day names
  -

Example:

Run every Friday night at 17:30

```
30 17 * * Fri root /usr/games/echoscript
```

- More Complex Crontab Dates & Times
  - A list of alternative values for a field are specified by commas:
  -

Run at :15 and :45 past each hour:

```
15,45 * * * * root /usr/games/ecscript
```

  - A range is specified with a hyphen:

Run every half hour 09:15-17:45 Mon-Fri:

```
15,45 9-17 * * 1-5 root /usr/games/ecscript
```

  - Numbers rather than names must be used for months and days in lists and ranges
  - A step through a range is specified with a slash:

Run every two hours 08:30-18:30 Mon-Fri:

```
30 8-18/2 * * 1-5 root /usr/games/ecscript
```

Exercises

1.

a. Set up a command which in a couple of minutes' time will write the disk usage of the partition containing */home/* to *~/home.use*.
b. Change to your home directory. Set up a command which in ten minutes' time will write the disk usage of the partition containing the current directory to a file. Repeat the above for two more directories in other partitions, writing to separate files. Before the time is up, display the list of pending commands. Examine each one. How do they differ?

2.

a. Set up a command which will mail you a reminder message in a few minutes. Remember that output from a job run by the at dæmon will be mailed to its owner, so there is no need to invoke an explicit mail command. Check that the mail arrives. *Mutt* is a simple mail reader you can use to do this.
b. Make the xeyes command start in a minute's time, capable of opening window's on your screen. Remember that any error messages from a command failing to run will be mailed to you.
c. Make a clock appear on your screen at four o'clock this afternoon.

3.

a. Create a file containing a command which will delete all files in *~/tmp/* when it is run. Make that script be run every hour.
b. Set up a script that once a minute appends the following data to a file:
   - The current date and time
   - A count of the number of processes currently running
   - Monitor that file, and start and stop programs to affect the number reported. When you've finished, deactivate the script.

c. As a non-privileged user, set up two scripts:
   - In even-numbered minutes, write the time the system has been running to a particular file.
   - In odd-numbered minutes, delete that file.

4. Set up scripts to run as a non-privileged user at the following times; you won't be able to test most of them, but it gives you practice at getting used to the order of the fields. Add the jobs one at a time, so that your crontab will be parsed to check for errors after each one.

- **at** 09:25 every Sunday in December
- **at** 21:30 on the 1st and 15th of every Month
- Every three hours at weekends
- Half past every hour during weekdays
- The first day of every quarter
- The evening of the first and third Thursday of each month
- At 17:30 every other day

Lecture: Maintain an Effective Data Backup Strategy

a) Creating Archives with tar
   - Use the c option to create an archive
   - For example, to create an archive called *docs.tar.gz* containing everything in the *Documents* directory:
   - create first phonebook.txt file in Documents

```
$ tar czf docs.tar.gz Documents
```

   - f specifies the archive's filename
   - Must be followed directly by the filename
   - Common to use *.tar* extension
   - Any subsequent options require a hyphen
   - The z option compresses the archive with gzip
   - *.tar.gz* extension used to indicate compression
   - *.tgz* extension also popular
   - The list of files and directories to archive follows the options.

b) Listing the Files in tar Archives
   - To check that a tar file has been made correctly, use the t operation (for 'lis*t*'):

```
$ tar tzf docs.tar.gz
```

   - The **z** and **f** options work as for the **c** operation
   - To show more information about files, add the **v** (for 'verbose') option
   - Shows information similar to **ls –l**
   - Can also be specified with c to list filenames as they are added

c) Extracting Files from tar Archives
   - Use the x operation to extract files from an archive:

```
$ tar xzvf docs.tar.gz
```

   - The **v** option lists the files as they are extracted
   - To extract individual files, list them on the command line:

```
$ tar xzvf docs.tar.gz documents/phonebook.txt
```

   - Other useful options:
     - k (--keep-old-files) will not overwrite any existing files, only extracting missing ones
     - p (--preserve-permissions) will set extracted files to have the permissions they had when archived

Exercises

1.

a. Create a single file in your home directory containing a backup of all the contents of */etc/*.
b. Create another archive with the same contents, but compressed to save disk space. Compare the sizes of the two archives.

c. List the contents of each of your archives.
d. Extract the contents of one of the archives into a directory under your home directory.
e. Create a new subdirectory, and extract a single file from the archive into it.

- Your current directory must be the one into which to extract.
- You will need specfiy the path of the file to be extracted, but without a leading slash.

2. With an archive of */etc/*, extracted under your home directory:

a. Modify at least two files in your extracted copy. Re-extract one of them from the archive, losing your changes in that one file but preserving your changes elsewhere.
b. Delete some files in your extracted copy. Make tar discover which these are and re-extract them from the archive, without clobbering changes made to other files.

3.

a. Produce a list of the names of all files under */home/* which have been modified in the past day. Only include regular files in this list.
b. Create a tarball containing all files under */home/* changed modified in the past day. Why is including directories in this list not sensible?
c. Create a tarball containing all files on the system that have changed in the past day, and which are in directories you deem worthy of being backed up.
d. Set up a cron job to make a daily backup of the system.

- It should run at 18:00 every day.
- The files created should be stored under */var/tmp/backup/*.

Each day's backup should be in a file named with that day's date, such as */var/tmp/backup/2003/04-07.tgz*