SWitCH – Sistemas de Computação e Redes (SCOMRED) 2020/2021 Laboratory 04

BASH programming – Algorithmics.

Tasks scheduling in Linux - CRON.

Before embracing this content, students should finish the activities and suggested exercises of the previous laboratory class.

1. Practical exercise: a BASH script for network monitoring

Create a BASH script to check hosts availability on the network, such checking will be implemented by performing ICMP echo tests (ping).

1.1. The script is to meet the following requirements:

- The script file to be created is **/root/bin/check-hosts**.
- The script must be designed to run unattended, possibly in background, therefore it shouldn't interact with stdin, stdout or stderr.
- The hosts to be checked are stored in the text file **/etc/check-hosts.conf**. Hosts may be represented by an IP address or a DNS name. Several hosts may be placed in the same line separated by spaces. Any line containing a hash signal should be ignored.
- A host availability checking is performed by sending to its address **a single ICMP echo request**, if an ICMP echo response is received within 2 seconds the test is successful and the host is regarded as being available, otherwise as unavailable.
- The script should log its actions into the file **/var/log/check-hosts.log**, this means appending text lines describing what is doing and each achieved result, bear in mind timestamps are mandatory on log files.
- Whenever a host is unavailable, a message should be sent to the **root** user in the system. For this purpose the **write** command is to be used. The write command has some outstanding limitation, the target user is required to have an active terminal session on the system, otherwise the message is lost, and also, if the user has several terminal sessions open, then the message is delivered only to one of those sessions.
- When the script is run, it will create an HTML page and store it on file **/var/www/html/check-hosts.html**. (Thus this page will be available as **https://vsX.dei.isep.ipp.pt/check-hosts.html**, where **vsX** is your virtual server's DNS name). This file's content is replaced each time the script is run, the contents must encompass at least a timestamp for when the file was generated, the list of hosts being checked and for each the status at that time (available/unavailable).

1.2. Implementation guidelines and hints.

- Use good programming practices, e.g. avoid constants spread around our code, always store them on variables on the first lines of the script, and of course this includes filenames.

- Use incremental programming, this algorithm encompasses a main loop than will go through the hosts list, then for each host several features are required to be implemented. Start by implementing the main loop, then additional features can be gradually added and tested one by one at a time.

- Regarding the test itself, you may use the **ping** command. You could focus on the command's output, but a much better option is focusing on its return value (exit code).

2. Task scheduling in Linux – the CRON service

Tasks scheduling is an important capability most operating systems make available to users, it allows a user to schedule a task execution (e.g. a program or a script) to a given time and date, and the user is not required to be logged in at the time of execution.

This is especially useful for systems administrators as they can schedule the periodic execution of unattended maintenance operations, most often at a time of low load on systems. Some examples of such operations are software updates and data backups.

CRON is the most popular tasks scheduler for Linux and other UNIX like operating systems, CRON is a service running in background (in UNIX such processes and services are commonly referred to as **daemons**).

2.1. The /etc/crontab configuration file

For every minute tick, the CRON service wakes and reads the content of the **/etc/crontab** configuration file, then it matches the current date/time with each entry on the file, for every match the corresponding task is executed.

Each line in **/etc/crontab** has at least seven attributes separated by spaces, the first five are a time/date pattern, if the current time matches that pattern, then CRON will **execute the command specified as 7th attribute using the username specified as 6th attribute**. After the 7th attribute arguments to be passed to the command may be used.

The first five attributes (time/data pattern) are:

1st – minute – legal values are integers from 0 up to 59.

2nd – hour – legal values are integers from 0 to up to 23.

3rd – day of the month – legal values are integers from 1 up to 31.

4th – month – legal values are integers from 1 to up to 12.

5th – day of the week – legal values are integers from 0 up to 7. Both 0 and 7 stand for Sunday.

For each of these attributes, a single value or a list of comma separated alternative values may be specified, a match will happen if the corresponding current date/time value is present in the list.

Instead of a comma separated list of values, for a set of consecutive values an inclusive range may be specified, with the first and last values separated by a dash.

The asterisk character represents the comma separated list of all possible values for an attribute, so it's equivalent to an inclusive range between the lowest possible value and the highest possible value.

The asterisk character (all possible values) may be associated by a forward slash with a step to create a subset of all possible values. For instance ***/20** in the minute specification is equivalent to **0,20,40**.

For a match to happen, a match on every attribute is required, so if no specific value is required for an attribute, the asterisk should be used.

Instead of the first five fields (time/data pattern), some special strings may be used, one very useful is **@reboot** which means run the task when the CRON service is started, something that happens when the server starts, but also whenever the CRON service is restarted. Nevertheless it's a good solution to execute something on the server boot.

2.2. Interpret an example of a CRON configuration file

Here is an example for a **/etc/crontab** file content:

```
*/5 * * * *  root /usr/bin/prog1 /etc/pptt.cfg

5,35 * * * * root /root/update -g all

45 2 * 8 6   root /root/make-backup
```

Try figuring out at what times each of these tasks will get executed.

2.3. Schedule the periodic execution today's class BASH script.

The hosts availability checking script, we have previously developed, may now be automatically executed from time to time.

Create a new configuration line in **/etc/crontab**, such as the hosts availability checking script is executed by the **root** user, every ten minutes, from Monday to Friday.