Costa Rica Institute of Technology

Computer Engineering
Operating Systems Principles
Prof.: Diego Vargas
02/22/2018

Short Assignment:

Creating a Linux Daemon using C and configuring it as a service (TrackerMon)

Motivation

As part of this short assignment a Linux Daemon will be created and configured as a Service using an init script.

“A daemon is a type of program on Unix-like operating systems that runs unobtrusively in the background, rather than under the direct control of a user, waiting to be activated by the occurrence of a specific event or condition.”[1]

Objective

Implement a Linux Daemon using C that will be starting as a service in Linux run-level 5. This will help to understand how the services are set to start in the Linux Operating System. The TrackerMon service will be taking care of monitoring key resources at Operating System Level like memory, CPU and network and recording the alerts in the specified log file.

General information

- Value of this assignment: 5%
- Executables names: **TrackerMon**.
- This project can be implemented in groups of 3 people. However it will be evaluated individually, during the review session random questions will be made to any of the students.
- Any fraud will be scored with 0 and will be processed according to TEC regulations.

Functional requirements

For the implementation of this project you will have to create the following:

1. The TrackerMon Daemon. This daemon has to be created in C for Linux. It will be taking care of the following:
    - Monitor CPU usage and generate an alert that will be appended in the defined log file when the CPU threshold is equal or greater than what is specified in the configuration file.
        - Sample alert message: [CRITICAL] – CPU Usage is currently ***<put here the current***

CPU usage> which is over ***<put here the threshold defined by the user>***

- Monitor memory usage and generate an alert that will be appended in the defined log file when the Memory threshold is equal or greater than what is specified in the configuration file.
  - Sample alert message: [CRITICAL] – Memory Usage is currently ***<put here the current Memory usage>*** which is over ***<put here the threshold defined by the user>***
- Monitor the network inbound connections and report SYN floods when they are detected. Monitor the SYN connections and generate an alert when it's detected the amount of connections are bigger than the defined threshold. This alert will be appended in the defined log file when the defined threshold for SYN connections is passed.
  - Sample alert message: [CRITICAL] – SYN flood connections detected. Currently there are ***<put here the amount of connections >*** active SYN_RECV connections ***<put here the file system with issues>*** which is over the defined limit ***<put here the SYN flood connections threshold>***
- Look for system critical errors and generate an alert that will be appended in the defined log file when any of the errors are triggered. Research logger and syslog to be able to generate alerts during the testing of your application.
  - Sample alert message: [CRITICAL] – System critical error has been detected: ***<put here the critical error message you found in syslog>***
- Read the configuration file to load the thresholds for monitoring of the resources, the location of this configuration file is going to be ***/etc/trackermon/config.conf***. This file has to be reloaded in to your system every time the daemon is started by the first time or restarted (in case a change is done to the configuration file, the process would have to be restarted using the service commands in order to re-read the configuration file values). The following is a sample configuration file:

```
# By default all the trackermon events should be tracked to /var/log/messages
# if the logfile is not specified by the user

LOGFILE=/var/log/trackermon.log

# CPU threshold. The system will create an alert when this limit is >=to the
# defined threshold
CPUthreshold=75%

# Memory Threshold. The system will create an alert when this limit is >=to
# the defined threshold
Memthreshold=75%
```

number of SYN Flood (SYN_RECV) connections allowed

SYNThreshold=10

2. TrackerMon should be running as a Linux service. The init script must be located under the /etc/init.d directory (that's where all the system init scripts are located in most of the Linux Operating Systems), you will name it **trackermon (/etc/init.d/trackermon)**. With the creation of init scripts, two ways of starting/stopping this daemon are going to be implemented.

   - Start up of **TrackerMon** at boot time: Your Daemon has to be started automatically by the Operating System by implementing the corresponding Init script. This script makes possible the execution of a given service at system boot time. Depending on which Unix / Linux flavor and distribution you're handling, there are different ways to set up a service (SysVinit or Systemd). You are free to choose the Linux distribution to use for the development and implementation of the daemon, the method to use will depend on your specific Linux distribution. This script is created using shell script and should have start, stop, status and restart commands, when the Operating System is starting it will be specifically calling the **start** function of this script. The **TrackerMon** service has to start in Run-level 5.
   - Stop (gracefully) **TrackerMon**: when the Operating System is shutting down, it gracefully stops all the services that are up and running in the Operating System, run-level by run-level by calling their corresponding **stop** function from the init script. TrackerMon has to implement its stop process while the Operating System is going down as well.
   - Start, Stop, status and restart of **TrackerMon**: The daemon should be able to be stopped, started or restarted at any time from the Linux terminal (as root) as well as to show the status of the process. When the daemon starts by the first time or when it's restarted, it has to re-read(reload) the values provided in the configuration file.

Program execution example:

The TrackerMon daemon has to be able to start automatically in run-level 5 when the Operating System boots. It also has to be able to be gracefully shutdown when the Operating system is going down.

When the Operating system is finally booted, TrackerMon has to be already started and performing all the requested actions. TrackerMon can be stopped, started, show its status or restarted manually at any time by using the following commands:

- Manually stopping TrackerMon (any of the commands is valid, the command to use will depend on your Linux distribution)

```
# service trackermon stop
Stopping trackermon… done
```

or

```
#  /etc/init.d/trackermon stop
Stopping trackermon… done
```

or

systemctl stop trackermon

- Manually restarting TrackerMon (any of the commands are valid, the command to use will depend on your Linux distribution)

service trackermon restart
Restarting trackermon… done

or

/etc/init.d/trackermon restart
Restarting trackermon… done

or

systemctl restart trackermon

- Manually starting TrackerMon (any of the commands is valid, the command to use will depend on your Linux distribution)

service trackermon start
Starting trackermon… done

or

/etc/init.d/trackermon start
Starting trackermon… done

or

systemctl start trackermon

- Show the status of TrackerMon. This command should display the status of the daemon and its corresponding PID

service trackermon status
daemon: TrackerMon is running (pid 17039)

or

/etc/init.d/trackermon status
daemon: TrackerMon is running (pid 17039)

or

systemctl status trackermon
In case the daemon is no longer running, the status command should display the following message:

daemon: TrackerMon is not running

Technical requirements

- This project has to be implemented using the C programming language.
- This program will have to be implemented on GNU/Linux (gcc).
- It will be combined with shell script to interact with the init script part.

Documentation

Follow the instructions below in order to document this project. **You don't have to print the documentation**, it will have to be delivered in digital format (PDF).

In any of the sections, you could re-use some of the information in this document if required.

- Introduction
  - Do an overview of the Linux Daemons:
    - How they work in Linux.
    - How they are created.
    - What are their main functions.
    - A high level explanation of how they work.
    - Any other relevant information.
  - Do an overview of systemd and Sys-V methods.
    - How they work?
    - How are they implemented?
    - What's the difference between the two of them?
    - Why all the Linux distributions don't use the same method as standard?
  - Provide a brief specification of your program, what it does, how it works. You are free to re-use some of the information provided in this document.
- Development environment
  - Provide all the details of all the tools that have been used during the development of this project.
- Data Structures, functions and libraries
  - Provide a description of all the functions, libraries and data structures used for the development of this project.
  - Provide the details of the design of your project using UML.
- Instructions of how to use the program.
  - Provide detailed instructions of how to start, stop, restart and show the status of the

TrackerMon process.

- Student activity log:
  - Include here the details of the activities performed by each of the students in this assignment.
  - Include one line per each activity, providing details like:
    - Description of the activity.
    - Amount of time in hours/minutes spent on each activity.
    - Total amount of hours spent in the development of this project per student.
    - **Use a time-sheet format** (or table) to present this activity log.
- Project final status
  - Provide a detailed status of your project.
  - Provide details of any issues, limitations or challenges you may have faced during the development of the project.
  - Document any known issues or bugs (if any).
- Conclusions
- Suggestions, recommendations.
- References
  - Provide all the references used during the development of this project.
- Make sure your **source code** is well **documented**.
- Digital documentation:
  - Include your documentation inside the .tar.gz file that will be created with all the source code and executables of this project.
  - Upload the .tar.gz file to your shared directory in Google Drive along with the corresponding timestamp, sha1 hash and your digital signature of the document.
  - Follow the instructions in the next section with the details of the structures of the directories you must use.

Deliverables

- Source code and executables of both programs. Both programs should be in compliance with the specifications in the **Technical requirements** section.
- Documentation (sources and pdf).
- Use the following structure inside your “Tareas” directory in Google Drive:

- Tareas:
  - <carne>-tarea1.tar.gz: compressed file with the following contents:
    - Documentation: this should be a directory with both the PDF and source files (.tex or .md) for the documentation.
    - Program: this should be a directory with both programs source code and executables.
    - Additional_files: this is an optional directory, it should have any additional file you think it would be required.
  - <carne>-tarea1.tar.gz.asc
  - <carne>-tarea1.tar.gz.tsr
  - <carne>-tarea1.tar.gz.sha1

Evaluation

- Creation of the TrackerMon daemon: 50%
  - CPU monitoring: 10%
  - Memory monitoring: 10%
  - Network monitoring: 10%
  - Capture of critical events from syslog: 10%
  - Tracking of events in the corresponding log file: 5%
  - Makefile: 5%
- Init script implementation: 30%
  - TrackerMon starts at boot time as a daemon: 5%
  - TrackerMon can be manually stopped: 5%
  - TrackerMon can be manually restarted (which reloads the configuration file): 5%
  - TrackerMon can be manually started: 5%
  - TrackerMon is able to display its status and PID: 10%
- Documentation using Latex or Markdown: 20%

Extra points:

- Documentation (source code, internal and external documentation) in English: 5%

Additional considerations

The programs and documentation are in separated items in the evaluation but the following restrictions apply:

- If documentation is not delivered, you will have automatically a score of 0 in this project.
- If the source code is not compiling, you will get a score of 0. Make sure you provide a functional source code.
- The program has to be programmed in C for GNU/Linux (gcc). Failure to do this will give you a score of 0 in this assignment.

Also the take the following under consideration:

- The professor will be reviewing the documentation out of the revision session.
- During the review session, the program deliverables could be downloaded from any of the students shared directories chosen by the professor.
- Each group will have up to 20 minutes to present the project during the review session and perform the technical defense of the work. The responsibility of presenting and defending all the work relies on each of the students, so it is recommended to have everything ready and handy prior the review session.
- Every error or warning displayed during the review session considered to be part of your technical validations during the development of the programs, will be punished on your final project score with -2 points.
- Each group will be responsible of the equipment that will be used during the review session. In case of issues to take your own equipment, you will have to notify the professor with 2 days in advance (prior the review session) so the required equipment can be properly coordinated.
- The following people could participate during the review session:
    - Other professors.
    - Coordinator of Computer Engineering.
    - Assistant.

Delivery Date

March 02th of 2018 before 23:59:59 GMT-6.

Revision Date

The review of this project will be done on *Saturday March 10th of 2018* at 08:00 GMT-6. During the review you will have to download the work from your shared directory and execute your work in the Computer Laboratory (H).

References

- The Linux Information Project. “Daemon Definition”. August 2005. Available at: http://www.linfo.org/daemon.html .