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

Fabian Astorga	Javier Sancho	Oscar Ulate
2014040808	2014159997	201229559

1 Introduction

1.1 Linux daemons

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 occurance of a specific event or condition [1].

1.1.1 How they work in Linux.

In linux exists three basic type of processes:

- Interactive: a user interaction with the operating system like a command in the terminal or interrups from the keyboard.
- Batch: Processes that are submitted from a queue and are not associated with command line, usually performing by tasks when the usage of the SO are low.
- The daemons are threated as processes, each processes in linux has a process identifier number (PID). All the processes are managed by the kernel.

Most of linux daemons init when tha system boot and dies when the SO shut down. Others daemons are started when needed and run as long as they are useful.

1.1.2 How they are created.

When a daemon starts up, it has to do some low-level housework to get itself ready for its real job. This involves a few steps:

- 1 Fork off the parent process
- 2 Change file mode mask (umask)
- 3 Open any logs for writing

- 4 Create a unique Session ID (SID)
- 5 Change the current working directory to a safe place
- 6 Close standard file descriptors

1- To make it truly autonomous, a child process must be created where the actual code is executed. This is known as forking.

2-In order to write to any files (including logs) created by the daemon, the file mode mask (umask) must be changed to ensure that they can be written to or read from properly.

3-It is recommended that you open a log file somewhere in the system for writing. This may be the only place you can look for debug information about your daemon.

4-The child process must get a unique SID from the kernel in order to operate.

5- The current working directory should be changed to some place that is guaranteed to always be there. The root directory in Linux is the most recomended.

6- Since a daemon cannot use the terminal, these file descriptors are redundant and a potential security hazard.[2]

1.1.3 How they work

Daemons are background process that run separately from the controlling terminal and just about always have the init process as a parent process ID (though theyre not required to); they typically handle things such as network requests, hardware activity, and other wait and watch type tasks. They differ from simple background processes that are spawned in the terminal because these background process are typically bound to that terminal session, and when that terminal session ends it will send the SIGHUP message to all background processes which normally terminates them. Because daemons are normally children of the init process, its more difficult to terminate them.[3]

1.2 SystemD and SysVinit methods

Both are system startup jobs that load the kernel and the posterior processes and services.

1.2.1 SysVinit

Siever [4] book says that for decades, the SysVinit was used in the Linux distributions, the traditional model divides potential system states into multiple run levels with distinct purpose. When entering in a run level N, SysVinit runs all the commands in the directory /etc/rcN.d. An issue with SysVinit is that it runs the commands sequentially, which forces commands and daemons to have a specific order, overall if they have dependencies between them, so if one command needs that a network daemon was initiated, it should be after this one.

1.2.2 SystemD

Petersen [5] describes that Fedora replaces the System V init daemon with the systemd init daemon. Whereas the System V init would start certain services when the entire system started up or shut down using shell scripts run in sequence, systemd uses sockets for all system tasks and services. Systemd sets up sockets for daemons and coordinates between them as they start up. This allows systemd to start daemons at the same time. Should one daemon require support from another, systemd coordinates the data from their sockets, so that one daemon receives the information from another daemon that it needs to continue. This parallel start up compatibility allows very fast boot times.

With those descriptions of the init daemons it is clear that the main difference is that systemd can start up more than one daemon at a time. On the other hand, System V init runs all the scripts in sequence. Another difference is that systemd is more complex because the use of sockets, System V init is more simple in that aspect, it only runs the files in order. Also with this complexity systemd manage the dependencies among daemons, System V init doesnt manage the dependencies and management should be manual.

1.3 Trackermon

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.

1.4 CPU Monitor

Monitor CPU usage and generate an alert that will be appended in the trakerMon log file when the CPU threshold is equal or greater than what is specified in the configuration file.

1.5 Memory Monitor

Monitor memory usage and generate an alert that will be appended in the TrackerMon log file when the Memory threshold is equal or greater than what

is specified in the configuration file.

1.6 Network Monitor

Monitor the network inbound connections and report SYN floods when they are detected. Monitor the SYN connections and generate an alert when its detected the amount of connections are bigger than the defined threshold. This alert will be appended in the TrakerMon log file when the defined threshold for SYN connections is passed.

1.7 Config File

This file has the threshold values for network, cpu and memory monitors. The content of the file is:

```
#
# Configuration trackermon file
#

LOG_FILE=/var/log/trackermon.log
CPU_THRESHOLD=5%
MEM_THRESHOLD=20%
NET_THRESHOLD=10%
```

2 Development environment

TrackerMon has been developed for Ubuntu 16.04, therefore, it uses systemd as the init method. It was developed using C for GNU/Linux, version 5.4.0. The source code was written in atom text editor.

For debugging and testing we used terminal commands like logger to generate the alerts that the daemon has to capture and print on the log file.

3 Data Structures, functions and libraries

All the code is written in the main file, called Trackermon.c. It contains the functions of the system. Inside the file you have all the used functions:

- struct funct_params_t: It is the structure that holds the thresholds of the parameters: memory, cpu and SYN connections.
- init_params: This function initializes the struct mencioned before.
- delete_spaces: This function is in charge of deleting spaces between words to the config.conf file.

- parse_config: This function is in charge of analize the config.conf file and set the parameters to the struct funct_params_t. It basically grabs the parameters from the disk and bring it to the memory so it can be used in the execution of the software.
- set_thresholds: Stablishes the thresholds values into the data structures needed for the algorithm.

4 How to use Trackermon

- First, download Trackermon daemon from to the computer.
- Second, copy the trackermon file to the */etc/init.d* folder with the following command in the terminal:

```
$   sudo cp trackermon /etc/init.d/
```

- The, copy the *config.conf* file to the */etc/trackermon/* folder with the following command:

```
$   sudo cp config.conf /etc/trackermon/
```

- After that, you are ready to compile Trackermon. Use the following command (Always inside the trackermon folder):

```
$   cd src
$   make
```

- Now you are ready to start de Trackermon daemon. Use the following command to start it:

```
$   cd /
$   sudo /etc/init.d/trackermon start
```

- To stop the daemon, use the command:

```
$   cd /
$   sudo /etc/init.d/trackermon stop
```

- To see the status of the daemon, use the command:

```
$   cd /
$   sudo /etc/init.d/trackermon status
```

- To restart the daemon, use the command:

```
$   cd /
$   sudo /etc/init.d/trackermon restart
```

- Finally, the log file is located in /var/log/trackermon.log

5 Student activity log

Assignment	Fabian Astorga	Javier Sancho	Oscar Ulate	Total
Daemon development	4:00	0:30	2:00	6:30
Config/init scripts	1:00	0:00	0:30	1:30
CPU monitoring	0:00	1:00	0:30	1:30
Memory monitoring	2:00	0:00	0:00	2:00
SYN Floods monitoring	0:00	1:00	6:30	7:30
Critical Msgs Monitor	1:30	4:30	0:00	6:00
Makefile, proj. management	2:00	0:00	0:00	2:00
Documentation	2:00	5:00	2:00	9:00

6 Project final status

The project was completed succesfully. All of the requirements were completed. The working group didn't know many of the features used in the system, so a lot of research was needed to complete TrackerMon.

7 Conclusions

1. Daemons and processes are very important in every operating system, and especially in linux. Therefore it is very important to understand how them and the operating system works. With a good implementation of the daemons, an engineer can create processes in the background that doen not need any intervention from an user. This is particularly important because you can interact with the system and capture events, and mount servers from the booting of the system itself.

2. A very useful and necesary tool is knowing the bash commands in GNU/Linux. The easiest way to make system calls is using bash commands. Those commands talk directly to the operating system to retrieve the required information.

3. You can not create any daemon without knowing the Linux File System. To keep the information well organized is important, and linux knows where look specific files it needs, such as congif files. This determines where to save the files of the daemon.

8 Suggestions, recommendations

1. It is recommended to implement a modular design of the code. That, because a modular design is easier to read and debug.

2. It is suggested to make a previous research of the Linux File System, GNU/Linux terminal commands and system calls to the OS.

3. It is remommended to

References

[1] The Linux Information Project. Daemon Definition. lifo.org. [Online]. Available: http://www.linfo.org/daemon.html. [Accessed: 28- Feb- 2018]

[2] Devin Watson. Linux Daemon Writting HOWTO. netzmafia.de. [Online]. Available: http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html. [Accessed: 26-Feb-2018]

[3] Aaron Krauss .How Daemons, the Init Process, and Process Forking Work. thesocietea.org. [Online]. Available: https://thesocietea.org/2016/11/how-daemons-the-init-process-and-process-forking-work/. [Accessed: 2-Mar-2018]

[4] Siever, E., Figgins, S., Love, R., Robbins, A. (2009) Linux in a Nutshell. Sebastopol, CA: O'Reilly.

[5] Petersen, R. (2016) Fedora Linux Servers with Systemd. Alameda, CA: Surfing Turtle Press.