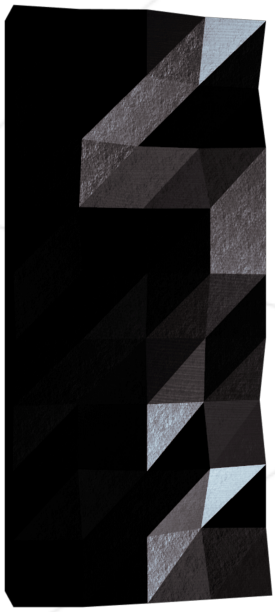

Project UNIX

Matt_daemon

42 Staff pedago@staff.42.fr

Summary: In this project, you will code a daemon.

Version: 4.1

Contents

Chapter I

Preamble

Chapter II

Introduction

A daemon – not a demon – is a computer program, a process, or a group of processes that run in the background rather than under the control of a user.

The word seems to have been introduced in 1963 by the CTSS designers at MIT as a reaction to the "dragon" designers at ITS. The retro-acronym "Disk And Execution MONitor" was created as a justification after the word spread out.

Daemons are often initiated during the OS loading and they usually respond to network requests, to hardware activity or to other programs running specific tasks. On Microsoft Windows, these functions are executed by programs called "services".

Chapter III

Objectives

This project is an introduction to the concept of daemon.

You're gonna have to code a little program - a daemon - aiming to treat and stock the messages received on a specific port, opened by this daemon.

Chapter IV

Mandatory part

Usage :

- The executable will be named `MattDaemon`.
- The program will have to run only with the root rights.
- Your program will have to run in the background, just like a real daemon.
- The daemon will have to listen to the port 4242.
- For your daemon's log, you will have to create a class called `Tintin_reporter` (you will be able to re-use it in future projects).
- Every daemon action should show in a log file `matt_daemon.log` with timestamp (in this form [DD / MM / YYYY - HH : MM : SS]) located in the folder /var/log/-matt_daemon/.
- You can create several logs if you like.
- Only one daemon instance should be able to run at once.
- When attempting to run a second daemon whereas one instance is already running, an error message indicating a creation/file opening on `matt_daemon.lock` attempt must pop.
- A `matt_daemon.lock` file must be created in /var/lock/ when the daemon starts.
- Ẁhen the daemon shuts down, the matt_daemon.lock file must be deleted.

- The program must quit with the sending of a simple "quit" character chain on the opened socket.
- Any other character chain must be registered in the log file.
- Only 3 clients can connect simultaneously to the daemon.
- When the daemon receives a signal, it must intercept it and register it in the `matt_daemon.log` file with an explicit message and quit properly.

`Of course, the signals that cannot be received won't have to be managed. If this happens, you will have to explain it during the evaluation.`

- Here is a log example:

```
# ./Matt_daemon
# ls -ali /var/lock/ | grep matt
34957 -rw-r--r-- 1 root root  0 Jan 11 14:34 matt_daemon.lock
# ps aux | grep Matt
root     6498 99.7 0.0  15172  2164 ?      Rs   14:34 43:44 ./Matt_daemon
# kill -15 6498
# ls -ali /var/lock/ | grep matt
# tail -n 7 /var/log/matt_daemon/matt_daemon.log
[11/01/2016-14:34:58] [ INFO ] - Matt_daemon: Started.
[11/01/2016-14:34:58] [ INFO ] - Matt_daemon: Creating server.
[11/01/2016-14:34:58] [ INFO ] - Matt_daemon: Server created.
[11/01/2016-14:34:58] [ INFO ] - Matt_daemon: Entering Daemon mode.
[11/01/2016-14:34:58] [ INFO ] - Matt_daemon: started. PID: 6498.
[11/01/2016-14:35:24] [ INFO ] - Matt_daemon: Signal handler.
[11/01/2016-14:35:24] [ INFO ] - Matt_daemon: Quitting.
# ./Matt_daemon
# ./Matt_daemon
Can't open :/var/lock/matt_daemon.lock
# ps aux | grep Matt
root     8062 98.7 0.0  15172  2364 ?      Rs   15:00  0:46 ./Matt_daemon
# tail -n 8 /var/log/matt_daemon/matt_daemon.log
[11/01/2016-15:00:25] [ INFO ] - Matt_daemon: Started.
[11/01/2016-15:00:25] [ INFO ] - Matt_daemon: Creating server.
[11/01/2016-15:00:25] [ INFO ] - Matt_daemon: Server created.
[11/01/2016-15:00:25] [ INFO ] - Matt_daemon: Entering Daemon mode.
[11/01/2016-15:00:25] [ INFO ] - Matt_daemon: started. PID: 8062.
[11/01/2016-15:00:26] [ INFO ] - Matt_daemon: Started.
[11/01/2016-15:00:26] [ ERROR ] - Matt_daemon: Error file locked.
[11/01/2016-15:00:26] [ INFO ] - Matt_daemon: Quitting.
```

- Here is a possible use example:

```
# ./Matt_daemon
# ps aux | grep Matt
root     3328 98.1 0.0  15168   208 ?      Rs   16:36  0:06 ./Matt_daemon
# nc localhost 4242
Salut
xd
quit
# tail -n 9 /var/log/matt_daemon/matt_daemon.log
[17/01/2016-16:36:07] [ INFO ] - Matt_daemon: Started.
[17/01/2016-16:36:07] [ INFO ] - Matt_daemon: Creating server.
[17/01/2016-16:36:07] [ INFO ] - Matt_daemon: Server created.
[17/01/2016-16:36:07] [ INFO ] - Matt_daemon: Entering Daemon mode.
[17/01/2016-16:36:07] [ INFO ] - Matt_daemon: started. PID: 3328.
[17/01/2016-16:36:43] [ LOG ] - Matt_daemon: User input: Salut
[17/01/2016-16:36:44] [ LOG ] - Matt_daemon: User input: xd
[17/01/2016-16:36:47] [ INFO ] - Matt_daemon: Request quit.
[17/01/2016-16:36:47] [ INFO ] - Matt_daemon: Quitting.
```

Chapter V

Bonus part

Bonus will be taken into account only if the mandatory part is PERFECT. PERFECT meaning it is completed, that its behavior cannot be faulted, even because of the slightest mistake, improper use, etc... Practically, it means that if the mandatory part is not validated, none of the bonus will be taken in consideration.

Bonus ideas:

- Create a graphic client to interact with the daemon.
- Add useful functions to your daemon (creation of a remote shell for instance!).
- Cypher sending and reception of the data (logically requires a client).
- Advanced log archival.
- Mail sending according to specific filters.
- Create an authentication system to connect to the daemon (through a graphic client/remote shell).
- Using advanced encryption systems (key public/private).

In case of client creation, the binary name will have to be Ben_AFK.

Chapter VI

Turn-in and peer-evaluation

- This project will only be reviewed by humans. You're free to organize and name your files as you will as long as you respect the following instructions.
- You must code in C++ (any version) and turn-in a Makefile (respecting the regular rules).
- Your classes must respect the **Coplien form**.
- You must manage errors a reasonable way. Your program should never quit unexpectedly (segmentation fault, etc).
- As usual, turn in your work on your repo `GiT`. Only the work included on your repo will be reviewed during the evaluation.
- For this evaluation, you will have to be in a VM with a Linux kernel version > 3.14. For your information, the grading scale was built with a stable 64-bit Debian 7.0.
- You can use anything you will need except libraries that will do the dirty work for you. This would be considered cheating.
- You can post your questions on the forum, Jabber, IRC, Slack...

`Of course, the daemon function is prohibited. The correct use of fork, chdir, flock (not LOCK_SH), and signal will be checked during the evaluation.`