Technical Manual of the `Universal Server`

Contact: about (dash) universal-server (at) esperide (dot) com

Creation date: Saturday, May 2, 2020

Lastly updated: Sunday, August 1, 2021

Version: 0.0.4

Status: In progress

Dedication: Users and maintainers of the `Universal Server`.

Abstract: The Universal Server, part of the umbrella project of the same name, is a multi-service daemon in charge of the automation (monitoring, scheduling and performing) of various computer-based tasks, such as the proper management of the server itself or, in the future, of house automation.

We present here a short overview of these services, to introduce them to newcomers.

Table of Contents

Overview

We present here a short overview of the general automated services offered by our so-called "Universal Server", to introduce them to newcomers. These services are implemented by US-Main, which relies notably on US-Common. The next level of information is to read the corresponding source files, which are intensely commented and generally straightforward.
The project repository is located here.

Layer Stack

From the highest level to the lowest, as summarised here, a software stack involving the Universal Server usually is like:

- the *Universal Server* services themselves (i.e. this us-main layer)
- [optional] the *Universal Webserver*, i.e. US-Web (for web interaction)
- US-Common (for US base facilities)
- [optional] Ceylan-Mobile (for 3G connectivity, notably SMS sending, relying on the Gammu library)
- [optional] Ceylan-Seaplus (prerequisite of Ceylan-Mobile for a bridge from Erlang to the C language)
- Ceylan-Traces (for advanced runtime traces)
- Ceylan-WOOPER (for OOP)
- Ceylan-Myriad (as an Erlang toolbox)
- Erlang (for the compiler and runtime)
- GNU/Linux

The shorthand for `Universal Server` is `us`.

Facilities Provided by this Layer

These are mainly per-host administration services centralised here.
The US Sensor Manager tracks automatically many **hardware sensors**; at start-up it detects the main available ones, regarding:

- **temperatures** at various locations: the CPU socket, the CPU package and cores themselves, any APU, the motherboard, the chipset, ACPI, some disks (ex: NVME); adding in the future GPU and RAM modules is considered
- the **speed of the fans** known of the motherboard (as opposed to any case fan that would be directly connected to the power supply)
- **chassis intrusion**, should such sensors be available

(other sensors like batteries, network or USB interfaces, etc. are ignored as their measurements are mostly voltage levels)
From then, the sensor manager periodically monitors these various measurement points: it filters bogus values, detects abnormal changes and reports to the user any related issue.

Sensor Tracking

Preparing the Setup

The monitoring done by this server relies on the `sensors` executable (typically `/usr/bin/sensors`, obtained generally from a package of the same name and relying on lm-sensors). One may install the `i2c-tools` package as well for DIMM information (see R2 below).
The `sensors-detect` script must have been run once by root beforehand (select then only the default, safer options, by hitting Enter repeatedly or simply use its `--auto` option), in order to configure sensors.
Configuration is typically stored in `/etc/sensors3.conf`, and must exist prior to running the US server.

Mode of Operation of the Sensor Manager

Once the sensor manager is started, **temperatures** are periodically tracked (current, minimum reported one, maximum, and average since start) and compared to thresholds (any critical temperatures reported by the chips, and alarm ones set by our sensor manager itself).
Abnormal temperatures (that is, going above - or even below - relevant thresholds) are then automatically timestamped and reported to the user by the US logic (i.e. notified in traces with appropriate severity, and possibly sent to the user thanks to emails and/or SMS).
Similarly, any **fan** that would stop whereas not being PWM is reported, and the same applies should an **intrusion** happen.
Many sensors report bogus values; the US Sensor Manager does its best to filter them out appropriately.
This includes temperatures outside of any realistic ranges and an intrusion being reported from boot whereas it did not happen.

Temperature monitoring Temperatures are monitored based on all the sensors that are supported by `lm-sensors` (notably the motherboard and CPU ones).
In the future extra information sources could be used:

- Hard Disk Drives, thanks to hddtemp, libatasmart, udisks2 or smartmontools
- DIMM Temperature sensors (see R2)
- GPU, thanks to XNVCtrl for NVidia ones, or ADL SDK for ATI ones

Refer to R5 for further details.
Platform Controller Hub (ex: `pch_cannonlake-virtual-*`, `pch_skylake-virtual-*`, etc.) are Intel's single-chip chipsets; they tend to run hotter than CPUs.
They may be reported as autonomous first-level entries, or as measurement points of the motherboard.

Fan Control The rotation speed of the fans can be measured thanks to `lm-sensors` as well.
Note that not all fans are known of the motherboard, notably the ones that are directly controlled by the user through a button (ex: stop/low/high) will remain invisible to all programs.
Currently the sensor manager is not able to discriminate between fixed-width fans and PWM ones.
The `pulses` attribute (ex: `fan2_pulses`) tells how many of such pulses are generated per revolution of the fan.

Chassis Intrusion In this last case, prior to launching the US server, one may try to reset them; for example, as root:

```
$ ls -l /sys/class/hwmon/hwmon*/intrusion*
-rw-r--r-- 1 root root 4096 Jul 11 19:30 /sys/class/hwmon/hwmon3/intrusion0_alarm
-rw-r--r-- 1 root root 4096 Jul  8 21:46 /sys/class/hwmon/hwmon3/intrusion0_beep
-rw-r--r-- 1 root root 4096 Jul 11 19:30 /sys/class/hwmon/hwmon3/intrusion1_alarm
-rw-r--r-- 1 root root 4096 Jul  8 21:46 /sys/class/hwmon/hwmon3/intrusion1_beep
$ echo 0 >| /sys/class/hwmon/hwmon3/intrusion1_alarm
$ cat  /sys/class/hwmon/hwmon3/intrusion1_alarm
1
```

As shown, this may not succeed.

Other Related Technical Information To access information regarding a given sensor, `psensor` may be used: open the preferences of the sensor (click on its name in the main window), and select the menu item *Preferences*, and look at the *Chip* field. See this link for more information.
The `sensors` tool is reporting values found in the Linux virtual file system directory, in `/sys/class/thermal/thermal_zone*/{temp,type}` for temperatures.
Examples:

- `Package id 0` is your (first) CPU
- `dell_smm-virtual-0` is your CPU fan, managed by your system firmware
- `acpitz-virtual-0` (*ACPI Thermal Zone*) is the temperature sensor near/on your CPU socket; this sensor can be unreliable
- `coretemp-isa-0000` measures the temperature of the specific cores

See the many comments in class_USSensorManager.erl for more details.
See also the following resources:

- R1: interpreting the output of `sensors`
- R2: the `lm_sensors` documentation of Arch Linux
- R3 and R4: `lm-sensors` tips and tricks
- R5: information about `psensor`
- R6: an example of preparation/tuning of one's sensors

lm-sensors Tips and TricksLicence

The `Universal Server` is licensed by its author (Olivier Boudeville) under the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of this license, or (at your option) any later version.
This allows the use of the Universal Server code in a wide a variety of software projects, while still maintaining copyleft on this code, ensuring improvements are shared.
We hope indeed that enhancements will be back-contributed (ex: thanks to merge requests), so that everyone will be able to benefit from them.

Current Stable Version & Download

As mentioned, the single mandatory prerequisite of the Universal Server is US-Common, which relies on Ceylan-Traces, which implies in turn Ceylan-WOOPER, then Ceylan-Myriad and Erlang.
We prefer using GNU/Linux, sticking to the latest stable release of Erlang (refer to the corresponding Myriad prerequisite section for more precise guidelines), and building the Universal Server from sources, thanks to GNU `make`.
We recommend, for all Erlang-related software, to rely on rebar3.
One wanting to be able to operate on the source code of these dependencies may define appropriate symbolic links in a `_checkouts` directory created at the root of `us-main`, these links pointing to relevant GIT clones.

Using Cutting-Edge GIT

This is the installation method that we use and recommend; the Universal Server `master` branch is meant to stick to the latest stable version: we try to ensure that this main line always stays functional (sorry for the pun). Evolutions are to take place in feature branches and to be merged only when ready.
Once Erlang, Cowboy and possibly Awstats are available, it should be just a matter of executing:

```
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Myriad myriad
$ cd myriad && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-WOOPER wooper
$ cd wooper && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-Traces traces
$ cd traces && make all && cd ..

# Possibly:
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Seaplus seaplus
$ cd seaplus && make all && cd ..

$ git clone https://github.com/Olivier-Boudeville/Ceylan-Mobile mobile
$ cd mobile && make all && cd ..


$ git clone https://github.com/Olivier-Boudeville/us-common
$ cd us-common && make all

$ git clone https://github.com/Olivier-Boudeville/us-main
$ cd us-main && make all
```

Using OTP-Related Build/Runtime Conventions

As discussed in these sections of Myriad, WOOPER, Traces and US-Common, we added the (optional) possibility of generating a Universal Server *OTP application* out of the build tree, ready to result directly in an *(OTP) release*.
For that we rely on rebar3, relx and hex.
Then we benefit from a standalone, complete Universal Server.
As for Myriad, WOOPER, Traces and US-Common, most versions of the Universal Server are also published as Hex packages.
For more details, one may have a look at:

- rebar.config.template, the general rebar configuration file used when generating the Universal Server OTP application and release (implying the automatic management of Myriad and WOOPER)
- rebar-for-hex.config.template, to generate a corresponding Hex package for Universal Server (whose structure and conventions is quite different from the previous OTP elements)

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Ending Word

Have fun with the Universal Server!