Unix Systems Event Manager

Álvaro Villalba Navarro
Director: Juan José Costa Prats

February 15, 2012

In this initial report we will explain the problem about abstract events scheduling, handling and reaction in Unix systems that this final project intends to solve and describe the chosen solution. As the nature of the final project is time-limited we won't cover the implementation of the whole solution but the most important functionalities, so the goals of this project will be stated explicitly.
Furthermore this document will state where we are inside the time-line of the project, which is the current approach to the solution and how much work has to be done until the end of the given time. In a nutshell, which are the goals that have been achieved and which need to be achieved yet. This will be done by defining a planning divided into explained tasks with assigned deadlines, a final statement about the current work being done and a forecast about the future of the project.
The reason for this document to be written in English, which is certainly not my mother tongue, is the Free Software nature of the project. This decision is expected to help its diffusion.

Contents

1 Planning

Here we will show the planning we intend to follow in order to 'have things done' in time. It's mainly a list of tasks to perform ordered from most to less priority and assigned deadlines to sets of those tasks (or iterations):

February 16th, 2012

- Main data structure of the daemon done and working.
- Wrappers to all the third party data structures (mainly glib).

  As it's expected to improve the efficiency of the software in the future, we want to wrapper all the declarations of the third party basic data structures we use, so when we re-implement them we won't have to change any other non-related code.
- Daemon socket ready to receive events from other processes.

  The poll management will be performed by 'libevent'.
- Execute shell commands functionality.
- Dummy first version of control program.

  Its purpose is to send events, so we can test the daemon.

After this deadline what we have are the very most basic and principal functionalities of the project. This is, a daemon with some states machines defined in it that receives event messages for those states machines from a socket and executes shell commands if assigned to the transitions. There's no way to read the states machines from anywhere yet so it has some test hard-coded state machines.

February 29th, 2012

- Transition tabs.

  Define syntax.

  Parser function.

  Saver function.

  Define files location by user.

Now we have implemented one of the lacking features of the basic version with more priority.

March 9th, 2012

- Control program with the following functionalities:

  Add transition.

  Generate event.

  Return all the states machines in a known graph syntax.

  f.e. DOT.

With this step done, we have the first basic but functional version of the project.

March 16th, 2012

- Shared library.

  This is expected to be easy and fast to finish, because it has been already implemented for the control program.
  'syslog.h' and 'libudev.h' style.

- Internal events.

  Implement a port to manage internal daemon events.

March 30th, 2012

- Propagation rules data structure.

  This must be an update of the main data structure to also check propagation rules or instead, creating a new parallel data structure.

- Propagation rules files.

  Define syntax.

  Parser function.

  Saver function.

  Define files location by user.

April 13th, 2012

- Daemon socket ready to receive TCP/IP event messages.

- Plugin schedulers interface.

  This is the main part of the project which will make use of internal events by now.

April 27th, 2012

- Test plugin.

  As much useful as time we have left to implement it.

- Current state storing.

  Define syntax.

  Parser function.

  Saver function.

  Define files location by user.

  Add the option to the 'transition tabs' syntax.

  Add the option to the control program.

- Action finalizing internal events.

  Finalize, error and success.

- Trigger currently valid events.

As we can see this set of tasks is a lot more dense than the previous. That's because we are taking in account that maybe we can't develop all the features because a lack of time, but again, maybe we don't have that lack of time and the project finishes with all those features in it. That's why the tasks are ordered by priority.
The minimum desired last task is the test plugin.