Application Monitor Application (APPMON)

version 1.0

Magnus Fröberg

1997-05-02

Typeset in LaTeX from SGML source using the DOCBUILDER 3.0 Document System.

Contents

APPMON Reference Manual

Short Summaries

- Erlang Module **appmon** [page 2] – The Graphical Node and Application Process Tree Viewer.

appmon

The following functions are exported:

- `start() -> {ok, Pid}`
  [page 2] Starts the node window

appmon (Module)

The application monitor `appmon` is a graphical node and application viewer. The tool shows an overview of all applications on all nodes, as well as the process tree of any application on any node.

Exports

```
start() -> {ok, Pid}
```

It starts the node window and will dynamically monitor all known Erlang nodes.

The Node Window

The node window monitors nodes. The window is named *appmon*. There is a menu-bar at the top, and below that each known node is shown in its own area, separated by a thin line. A node is represented by its name as the root of its application tree. The node name represents the application controller on this node. The name itself is a menu, where operations on the node can be performed (rebooting it for instance), by clicking. To the far left is a load meter for each node. Load can be measured as processor time/elapsed time, or as the length of the runtime queue, by selecting the appropriate option from the Options menu. Below the node name, the application controller, is a tree of applications. A clicking on one of these applications starts the application monitor window.

The node window has a menu-bar with the following options:

File:Quit Stops the monitor.

Options:Listbox Opens the Listbox Window.

Options:Load: time Load is calculated as processor time.

Options:Load: queue Load is calculated as the length of the ready queue, which is the number of processes that are ready to execute.

Options:Load: progressive A progressive scale is used for load values. Even a small machine load will show up on the meter, and if the machine load doubles the meter will show less than double the load. This is a nice feature for demos. Default is load mode.

Options:Load: linear A linear scale is used for load values. When the machine load doubles, the meters shows double the load.

Each application controller visible in the node window can be selected by mouse click and produces a menu with the following options:

Reboot Reboots the node

Restart Restarts the node

Stop Stops the node

Ping Pings the node. This is useful for re-establishing contact

The Listbox Window

The listbox window lists the nodes and applications that are shown in the appmon window. This window can be used when there are so many applications in the appmon window that it will be hard to read their names.

The List Box window is opened by a single click on *Options: List Box*. A clicking on *Close* closes the window.

The Application Window

The application window monitors an application. The name of the window is the application name and the node name. The window shows a menu-bar, a toolbar and the application tree.

The Applicaton window can be opened from the appmon window by a single click on the application, or from the listbox window by dubbel clicking on the application name or by marking the application and click on the *Load* button.

Application trees come in two varieties:

- a supervision view where a strict supervision tree is shown
- a process view where all linked processes in the application are shown.

In supervision mode, the tree-gathering and algorithm building assume conformance to supervision implementation. Otherwise it works for standard Erlang.

The application window menu-bar contains the following options:

File:Quit Quits the application window.

Options:Refresh Updates process information and refreshes the screen.

Options:Sup. view Shows strict supervision tree only.

Options:Proc. view Shows all linked processes.

The application toolbar controls the actions of the mouse pointer. Clicking on a process in the application window have different meaning depending on which tool is selected. The procedure is to first press a button - T for trace, for example - and then click on the process that should be traced.

The application window toolbar contains the following options:

Info Process information is printed in the shell. This is the default mode and this mode is automatically resumed after a mouse click.

Send Sends a message to the selected process. A window pops up where the message can be written (and the destination Pid can also be changed). Double-clicking on this button displays a pop-up window, where the Pid and the message can be entered.

Trace Toggles the tracing of a process On/Off. This is implemented with the `sys:trace` call and not all processes respond to this.

Kill Send an exit signal, which cannot be trapped to the process.

Files

Appmon consists of the following modules:

`appmon` The node window. It is required on the server node only.

`appmon_lb` The listbox window. It is required on the server node only.

`appmon_a` The application window. It is required on the server node only.

`appmon_dg` Internal file containing data structure handling. It is required on the server node only.

`appmon_place` Internal file containing the process placement algorithm. It is required on the server node only.

`appmon_txt` The help text viewer.

`appmon_info` The node information centre. The file is required on all nodes to be monitored, including the server node.

Files and Distribution

Appmon is a distributed program, but all graphics are handled on a central node and `GS` need only be loaded on this single node. All real monitoring is done in a single module (`appmon_info`), which runs on each node acting as an information centre. The various windows, the node window, the listbox window and the application window, then subscribe to information.

In summary then, only the `appmon_info` module must be accessible by the clients, the rest of the appmon modules are only needed at the server.

Troubleshooting

If the node window does not show all the nodes expected, then check the following:

1. Are the nodes visible with the `nodes()` command in the shell? If not, do `net_adm:ping(NodeName)` to add nodes to the Erlang distribution.

2. Can the appmon_info module be loaded at the missing node or nodes?

If the application window does not show the supervision tree, then check that the application have a real application master. It is absolutely necessary that the top process behaves as a supervisor when the supervision view is used.

Chapter 2

APPMON Release Notes

The Application Monitor Application (*APPMON*) is a graphical utility used to supervise Applications executing on several Erlang nodes. The process-tree of an Application can furthermore be monitored.

2.1 APPMON Release Notes

This document describes the changes made to the appmon application.

APPMON 1.0.4

Fixed Bugs and malfunctions

- The application window seems to die in rare cases when new processes are found.
  Own Id: OTP-1970

APPMON 1.0.3

Fixed Bugs and malfunctions

- Text fits into the load meter frame.
  Own Id: OTP-1358

Known bugs and problems

- Links to the same remote process doesn't look right.
  Own Id: OTP-1202
- The application window seems to die in rare cases when new processes are found.
  Own Id: OTP-1970

APPMON 1.0.2

Fixed Bugs and malfunctions

- Appmon now conforms to standard window protocol concerning deletion and stopping of application and node windows.
  Own Id: OTP-1179
- Scrollbars now automaticly appears when window becomes too large.
  Own Id: OTP-1163

APPMON 1.0.1

Improvements and new feature

- Appmon now displays process info in a window instead of a shell.
  Own Id: OTP-1137
- Appmon now uses proc_lib ancestor information if present, otherwise it makes an educated guess on which links are primary and which are not.
  Own Id: OTP-1143

Fixed Bugs and malfunctions

- Named process pids are now updated in application window.
  Own Id: OTP-1044
- The appmon toolbar now uses spelled out names instead of one letter abbreviations.
  Own Id: OTP-1091
- All colour names converted to RGB tuples.
  Own Id: OTP-1136
- Links to processes on remote nodes are shown as blue lines in the application window.
  Own Id: OTP-1159
- Application window stays up when application dies and reconnects when application comes up again.
  Own Id: OTP-1161
- Appmon .app file updated to include necessary file.
  Own Id: OTP-1205
- Appmon is more stable and forgiving when Erlang nodes are stopped and started repeatedly.
  Own Id: OTP-1210

Incompatibilities with APPMON 1.0

-

Known bugs and problems

Scrollbars doesn't behave quite as they should.

Own Id: OTP-1163

Index

Modules are typed in *this way*.
Functions are typed in `this way`.