Eina Debug & Clouseau

EFL Dev Day 2017 @Malta
• Introduction
• How it works?
• Rules of the game
• Eina Debug internals
• Clouseau internals
• Demo
• Resources
• Eina Debug:
  • Communication channel between an EFL debugger and an EFL application
  • Internal features: EvLog, Backtrace, CPU…
  • Supports dynamic opcodes registrations

• Clouseau: EFL debugger
How it works?

- Green link: connection to the daemon as a normal application
- Red link: connection to the daemon as a debugger

```
ssh ip -L7777:localhost:6666
```
Connection establishment

- Launch efl_debugd (EFL daemon)
- Launch the application to debug
- In case Clouseau is not launched in the target, create a connection to the target with port forwarding, e.g:
  
  ```
  ssh 10.x.x.x -L7777:127.0.0.1:6666
  
  sdb forward tcp:7777 tcp:6666
  ```

- Launch Clouseau:
  
  With -l option to connect to the local daemon
  
  With -r port to connect remotely (7777 in our previous example)
• The daemon must run before the app is launched
• If the connection drops, the app must be restarted to establish a new connection with the daemon
• The apps and the daemon communicate via a UNIX socket
• The debuggers connect via a TCP port to the daemon. For a remote connection, an external tunneling (ssh, sdb…) is needed.
• eW era elttil naidne!
• The client sends automatically to the daemon:
  protocol version / pid / application name

• Debuggers can register to get update on addition/deletion of applications

• When a new application says hello, the daemon updates the debuggers:
  client id / pid / application name

• When a new debugger connects, the daemon sends all the applications information
Opcodes registration

```c
#define CPU_FREQ_ON 3
#define CPU_FREQ_OFF 4
#define EVLOG_GET 5

void cpu_on_op()
{
    // CPU_ON_OP
}

void cpu_off_op()
{
    // CPU_OFF_OP
}

void evlog_get_op(void (*cb)())
{
    // EVLOG_GET
}
```

```
void _cpufreq_on()
{
    // CPU_FREQ_ON
}

void _cpufreq_off()
{
    // CPU_FREQ_OFF
}

void _evlog_get(void (*cb)())
{
    // EVLOG_GET
}
```
# Packet header and dispatching

<table>
<thead>
<tr>
<th>Total size</th>
<th>Client id source or destination</th>
<th>Opcode</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Op string</th>
<th>Callback</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CPU/Freq/on</td>
<td>_cpufreq_on</td>
</tr>
<tr>
<td>4</td>
<td>CPU/Freq/off</td>
<td>_cpufreq_off</td>
</tr>
<tr>
<td>5</td>
<td>EvLog/get</td>
<td>_evlog_get</td>
</tr>
</tbody>
</table>
Add new operations

- At anytime, opcodes can be registered via the function `eina_debug_opcodes_register`:
  - Session: if NULL, will use the default connection to the daemon
  - The opcodes, in this format:

```c
EINA_DEBUG_OPCODES_ARRAY_DEFINE(_OPS,
    {"Domain/Feature/action", &opcode, &_op_cb},
    {NULL, NULL, NULL}
);
```
- A function to call when the opcodes have been received by the daemon
Requests handling

- Callback prototype:

  ```c
  Eina_Bool (*)(Eina_Debug_Session *session, int srcid, void *payload, int payload_size);
  ```

- It is the responsibility of the developer to extract the payload correctly (including endianness)

- If a response is required, the function `eina_debug_session_send` can be used to send a response to the requester
• Full rewriting of the application
• Includes the main features of the first Clouseau
• Extensions
• Snapshot new mechanism
Clouseau extensions

- WTF is that??!?!?!?

- Two extensions are sitting by default inside Clouseau repository:
  - Objects introspection:
    - Displays a tree of the objects of the application
    - Screenshot can be taken and shown easily
    - Highlighting on focus feature is available
  - EvLog:
    - Allows you to instrument events and timelines
    - Tool initially written by Carsten
Clouseau: create an extension

• Need to compile as a library

• Mandatory functions:
  • extension_name_get: get the name of the extension
  • extension_start: request to start the extension
  • extension_stop: request to stop the extension
• Information needed by Clouseau:
  • session_changed: invoked when the session changes
  • app_changed: invoked when the app changes
  • import_data: invoked when the user opened a snapshot file
  • export_data: invoked when the user wants to save a snapshot
  • ui_object: the main object to display on the screen
• Information needed by the extension:
  • session: the current session
  • app_id: the current application id
  • path_to_config: the path to the extensions configuration
Snapshots

- Supports snapshot of many extensions in one EET file
- Each extension serializes the information into a buffer

<table>
<thead>
<tr>
<th>App name / pid</th>
<th>Extension 1</th>
<th>...</th>
<th>Extension n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Data + length</td>
<td></td>
<td>Data + length</td>
</tr>
<tr>
<td></td>
<td>Version</td>
<td></td>
<td>Version</td>
</tr>
</tbody>
</table>
• Repositories:
  • Eina Debug: core/efl
  • Clouseau: tools/clouseau
• Wiki: https://phab.enlightenment.org/w/eina_debug/
Q & A

ZENDO