

# Debugging YaST Scripts

Tips for debugging YaST modules

**Ladislav Slezák**

Ing.

**Novell.**<sup>®</sup>

# Debugger in YaST

There is no debugger in YaST

– There was a simple debugger in the past...

So how to debug a YaST script?

# Debugging YaST

- Use `y2log` (`/var/log/YaST2/y2log`)
  - > Put `y2milestone()` wherever it may be useful
  - > Use `y2debug` – not logged by default
  - > Use `y2internal` for testing, remove or change it to `y2debug` later
  
- Use 'showy2log' colorizer
  - > Command: `'showy2log -v -- tail -f'`
  - > Displays `y2log` in colors depending on the log level (errors in red...)
  - > Formats YCP structures (lists, maps) – more readable output
  - > Open a separate terminal window with it

# More verbose logging

## Enabling debug messages

- Set variable Y2DEBUG=1 (ZYPP\_FULLLOG=1)
- Press Shift+F7 in the Qt UI (dynamic switching on/off decreases useless output)

## Another shortcut

- Shift+Ctrl+Alt+X in Qt UI opens an xterm window with shell

# Debugging in YCP

## Backtrace in YCP code

- Use -1 as the first parameter in `y2<loglevel>()` call

```
y2internal(-1, Backtrace: );
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] sound/read_routines.ycp:51 Backtrace:
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] sound/read_routines.ycp:157 extractUniqueKey (comment)
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] sound/read_routines.ycp:224 read_modprobe (.modprobe_sound)
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] Sound.ycp:516 read_save_info ()
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] Sound.ycp:710 Sound::ReadModulesConf ()
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] sound/wizards.ycp:262 Sound::Read (true)
```

```
2007-10-10 15:21:54 <5> cyclops(14553) [YCP] clients/sound.ycp:588 CommandLine::Run (cmdline)
```

## Breakpoints

- Use e.g. `Popup::Message()` calls
- Do not forget to remove them! (`svn diff`)

# Modifying Installation System

- Use /y2update for overriding the current files
  - YCP interpreter looks the files up in /y2update, \$Y2DIR, \$HOME/.yast2 and in \$PREFIX (/usr)
  - Create e.g. /y2update/clients, copy the file from /usr there
  - Don't forget to compile YCP modules (`yccpc -c`)
  - Works also for non YCP parts (agents, package-bindings...)
  - Very convenient for debugging YCP clients (clients are reloaded when executed)
  - Use `start_shell=1` boot option to modify the system before start
- Modify the installation system (inst-sys)
- Create an addon-product
  - Similar to y2update
  - Also for adding packages to the inst-sys

# Debugging YaST Agents

- For interactive debugging run  
`/usr/lib/YaST2/bin/y2base studio scr`
- Use ``Read(.path)`, ``Write(.path,...)`,  
``Execute(.path,...)`, ``Dir(.path)` commands  
instead of e.g. `SCR::Read()` in YCP script

**Note:** `Rlwrap` (Readline wrapper) provides easy editing with history for any command line tool

# Debugging a crash

- Crash – usually SIGSEGV (11), SIGABORT (6), see y2log
- Backtrace – probably the most valuable information, tells the exact location of the crash and program stack
- \*-debuginfo packages are required for full debug output
  
- How to get a backtrace?
  - > Start YaST in gdb debugger

```
gdb /usr/lib/YaST2/bin/y2base
run <module> qt
```
  - > Attach gdb to running YaST

```
gdb /usr/lib/YaST2/bin/y2base <PID>
```
  - > Reproduce the crash
  - > Use 'bt' command in gdb
  
- See <http://en.opensuse.org/Bugs/YaST> for more details

# UI Debugging

- Debugging functions
  - > UI::DumpWidgetTree() - dumps UI tree to y2log
  - > UI::FakeUserInput() - simulates user input

# Questions?

- Do you have another tip for debugging?

**Novell.**<sup>®</sup>