Theme-D User Guide

Tommi Höynälänmaa

May 28, 2018

Contents

1 Installation

1.1 Debian-based systems

These instructions apply to Debian-based Linux distributions such as Debian and Ubuntu. The default directory configuration of Theme-D is stored in file `/etc/theme-d-config.scm`. You may override this by defining environment variable `THEME_D_CONFIG_FILE` to be the path of your own configuration file. The root directory of the Theme-D installation shall be called *theme-d-root-dir*. By default this is `/usr/share/theme-d` in Debian-based installations and `/usr/local/share/theme-d` in other installations.

Install first one of the packages `guile-2.0-dev` or `guile-2.2-dev`. Note that these packages can't be installed simultaneously. Use command

```
sudo apt-get install guile-2.0-dev
```

or

```
sudo apt-get install guile-2.2-dev
```

You can check if these packages have already been installed with commands

```
dpkg -s guile-2.0-dev
dpkg -s guile-2.2-dev
```

1.1.1 Package `guile-2.0-dev` and Amd64 / Intel 64-bit x86 Processor Architecture

1. Install TH Scheme Utilities version 1.4.1 in case you do not have it already. Versions 1.3 and 1.4 work, too. See http://www.iki.fi/tohoyn/theme-d/. Installation is done with command

   ```
   sudo dpkg -i th-scheme-utilities_1.4.1_all.deb
   ```

   in the directory where you have the Debian file.

2. Install `libthemedsupport` version 1.1 with command

   ```
   sudo dpkg -i libthemedsupport_1.1_amd64.deb
   ```

   in the directory where you have the Debian file.

3. Install Theme-D with command

   ```
   sudo dpkg -i theme-d_1.1.0_amd64.deb
   ```

   in the directory where you have the Debian file.

1.1.2 Other Configurations

1. Install TH Scheme Utilities version 1.4.1 in case you do not have it already. Versions 1.3 and 1.4 work, too. See http://www.iki.fi/tohoyn/theme-d/. Installation is done with command

   ```
   sudo dpkg -i th-scheme-utilities_1.4.1_all.deb
   ```

   in the directory where you have the Debian file.

2. Rebuild the `libthemedsupport` version 1.1 package for your architecture. See instructions in the `libthemedsupport` documentation. Give the command

   ```
   sudo dpkg -i libthemedsupport_1.1_arch.deb
   ```

in the directory where you have built the Debian package file. Here *arch* is the name of your processor architecture.

3. Change to the directory where you want to unpack the Theme-D source code

4. Unpack Theme-D source code with command

```
tar xzvf mypath/theme-d_1.1.0.tar.gz
```

   where *mypath* is the path of the Theme-D package file.

5. Change to the subdirectory `theme-d-1.1.0`.

6. Change the architecture entry (10th line) in file `debian/control` from `amd64` to your processor architecture if necessary.

7. If you use Guile 2.2 change the value of the variable `GUILE_VERSION` to 2.2 in file `debian/rules` (6th line).

8. Give command

```
dpkg-buildpackage -uc -us
```

9. Give commands

```
cd ..
sudo dpkg -i theme-d_1.1.0_arch.deb
```

   where *arch* is the name of your processor architecture.

1.2 Other Systems

1. Install Guile 2.0 or 2.2 if you don't have it already. Check the version of the Guile development environment with commands

```
pkg-config --modversion guile-2.0
pkg-config --modversion guile-2.2
```

   See http://www.gnu.org/software/guile/.

2. Install TH Scheme Utilities version 1.4.1 in case you do not have it already. See the instructions in the TH Scheme Utilities documentation. Versions 1.3 and 1.4 work, too.

3. Install the `libthemedsupport` library version 1.1. See instructions in the `libthemedsupport` documentation.

4. Create some directory and unpack Theme-D package there with command

   `tar xzvf` *theme-package-path*`/theme-d_1.1.0.tar.gz`

   The subdirectory `theme-d-1.1.0` of the directory where you unpacked Theme-D shall be called *theme-d-source-dir*.

5. Change to the the subdirectory *theme-d-source-dir*.

6. Give command

   `./configure`

   You may give the following options to command `./configure`:

   - `--with-guile=`*version* : Specify the Guile version explicitly. The version has to be either 2.0 or 2.2.
   - `--without-support-library` : Don't use the `libthemedsupport` library.
   - `--disable-extra-math` : Don't include the `(standard-library extra-math)` module in your installation.
   - `--disable-posix-math` : Don't include the `(standard-library posix-math)` module in your installation.

   If you use option `--without-support-library` option you also have to use options `--disable-extra-math` and `--disable-posix-math`.

7. Change to the the subdirectory *theme-d-source-dir* and give command

   `make`

   in order to prepare the code for installation. Install Theme-D with command

   `sudo make install-complete`

   If you have logged in as the root user you may use command

   `make install-complete`

   If you do not have `sudo` you may try command

   `su root make install-complete`

   Finally, give command

```
make compile-scheme-code
```

1.3 Local mode

Using Theme-D in the source code tree without installing it is called *local mode*. This is useful if you develop Theme-D itself. It is recommended that you should not use Theme-D simultaneously with Debian-based installation and in local mode.

1. Install guile 2.0 in case you do not have it already. See http://www.gnu.org/software/guile/.
2. Install TH Scheme Utilities version 1.4.1 in case you do not have it already. See the previous sections.
3. Install the `libthemedsupport` library version 1.1. See instructions in the `libthemedsupport` documentation.
4. Create some directory and unpack Theme-D package there with command

   `tar xzvf` *theme-package-path*`/theme-d_1.1.0.tar.gz`

5. Go into the the subdirectory `theme-d-1.1.0` of the directory created in the previous step. Give commands

   ```
   ./configure
   make
   make setup-local-config
   ```

   See section 1.2 for the configure options.

2 Removing the software

2.1 Debian-based systems

Give command

```
sudo dpkg --purge theme-d
```

If you want to remove TH Scheme Utilities, too, give command

```
sudo dpkg --purge th-scheme-utilities
```

If you want to remove the `libthemedsupport` library give command

```
sudo dpkg --purge libthemedsupport
```

2.2 Other systems

Give command

```
sudo make uninstall-complete
```

in directory *theme-d-source-dir*. See documentation in TH Scheme Utilities source package for instructions to remove the library. See documentation of `libthemedsupport` for instructions how to remove the library.

3 Theme-D environment

4 File Extensions

Theme-D source files have the following extensions:

- `.thp` for proper programs
- `.ths` for scripts
- `.thi` for interfaces
- `.thb` for bodies

Theme-D compiled pseudocode files have the following extensions:

- `.tcp` for proper programs
- `.tcs` for scripts
- `.tci` for interfaces
- `.tcb` for bodies

5 Unit Root Directories

When you define a unit with full name

(*dir-1* `...` *dir-n* *unit-name*)

the module must have filename *unit-name* with proper extension (see the previous section) and it must be located in subdirectory

dir-1`/.../`*dir-n*`/`

of some directory *unit-root-dir*. The directory *unit-root-dir* is called a *unit root directory*. If a unit name has only one component you may omit the parentheses from the unit name. When you compile of link a Theme-D unit you must specify one or more unit root directories where the imported modules are searched. These are called the *module search directories*. You should always have directory *theme-d-root-dir*`/theme-d-code` among the module search directories so that the standard libraries are found by the compiler and by the linker.

6 Compiling a Theme-D Unit

Give command

`theme-d-compile` *options unit-name*

where *unit-name* is the file name of the Theme-D unit. Options are

- `--module-path=` *paths* or `-m` *paths* : Module search paths separated with :'s
- `--output=` *output-filename* or `-o` *output-filename* : The output filename
- `--unit-type=` *unit-type* or `-u` *unit-type* : The unit type (`program`, `interface`, or `body`)
- `--message-level=` *message-level* or `-l` *message-level* : Compiler message level, integer number from 0 to 3.
- `--expand-only` : Do only macro expansion on the source.
- `--no-expansion` : Compile the source without macro expansion.
- `--backtrace` : Print backtrace on compilation error.
- `--pretty-print` : Pretty print the pseudocode output.
- `--no-verbose-errors` : Less information in the error messages.
- `--show-modules` : Show information about loading modules.

By default the unit type is computed from the source file extension. The default module search path is *theme-d-root-dir*`:..` If you use option `-m` you may include the Theme-D default module search path in your custom path by adding an extra ":" in the beginning of the new path, e.g. `:my-path1:my-path2`. The default target file path is obtained by removing the path and the extension from the source filename and appending the appropriate extension to the result. The default message level is 1. Message level 0 means no output at all except in case of error. Message level 1 displays also message on successful compilation or linking. Message level 2 displays some debug information and level 3 a lot of debug information. When `--expand-only` is set the default target filename is `myunit.expanded.thx` for source file `myunit.thx`.

Suppose that you have your own Theme-D code at directory *my-theme-d-dir* and you have a program called `(mod-1 ... mod-n)` at location

```
mod-1/.../mod-n.thp
```

In order to compile the program give commands

```
cd my-theme-d-dir
theme-d-compile mod-1/.../mod-n.thp
```

Suppose that you have a module (an interface and a body) with name `(mod-1 ... mod-n)` in files `mod-1/.../mod-n.thi` and `mod-1/.../mod-n.thb`. In order to compile the module give commands

```
cd my-theme-d-dir
theme-d-compile mod-1/.../mod-n.thi
theme-d-compile mod-1/.../mod-n.thb
```

If you want to have the compiled files in the same subdirectory where the source files are, which is usually the case, give commands

```
cd my-theme-d-dir
theme-d-compile -o mod-1/.../mod-n.tci \
  mod-1/.../mod-n.thi
theme-d-compile -o mod-1/.../mod-n.tcb \
  mod-1/.../mod-n.thb
```

If you use Theme-D without installing it you have to use command

```
MYPATH/theme-d-VERSION/translator/theme-d-compile.scm
```

instead of `theme-d-compile`. Here `MYPATH` is the path where you have unpacked Theme-D.

7 Linking a Theme-D Program

Give command

`theme-d-link` *options program-name*

where *program-name* is the file name of the Theme-D program. Options are

- `--module-path=` *paths* or `-m` *paths* : Module search paths separated with :'s
- `--output=` *output-filename* or `-o` *output-filename* : The output filename
- `--intermediate-file=` *filename* : The output filename
- `--intermediate-language=` *language* : The language used for the intermediate file, either `tree-il` or `scheme`.
- `--message-level=` *message-level* or `-l` *message-level* : Linker message level, integer number from 0 to 3.
- `--no-final-compilation` : Do not compile the linker result file with `guild compile`.
- `--no-strip` : Do not strip away unused code.

- `--no-factorization` : Do not factorize the type expressions out of procedure implementations.
- `--no-weak-assertions` : Do not check ordinary assertions. Strong assertions are always checked.
- `--backtrace` : Print backtrace on linking error.
- `--pretty-print` : Pretty print the linker output.
- `--no-verbose-errors` : Less information in the error messages.
- `--keep-intermediate` : Keep the intermediate Tree-IL or Scheme file
- `--link-to-cache` : Link the target file into the guile cache.
- `--runtime-pretty-backtrace` : Generate the code to support runtime pretty printed backtraces.
- `--no-unlinked-procedure-names` : Do not generate code for reporting unlinked procedure names.

By default Theme-D linker produces a guile objcode file. Actually, Theme-D makes a guile Tree-IL or Scheme file and uses guile to make an objcode file from that. The default intermediate language is Tree-IL. Note that many optimizations are performed only with Tree-IL. Runtime pretty backtraces are supported only for Tree-IL. If you want to optimize your code for speed you should link your program without pretty backtraces when you no longer need them for debugging.

If you use option `--module-path` or `-m` you may include the Theme-D default module search path in your custom path by an extra ":" in the path as in compilation. Suppose that you have your own Theme-D code at directory *my-theme-d-dir* and you have a program called `(mod-1 ... mod-n)` at location `mod-1/.../mod-n.thp`. In order to link the program give commands

```
cd my-theme-d-dir
theme-d-link mod-1/.../mod-n.thp
```

The previous commands place the linked file into the root of subdirectory *my-theme-d-dir*. If you want to place the linked file in the same directory where the source files are use the following commands:

```
cd my-theme-d-dir
theme-d-link -o mod-1/.../mod-n.go \
  mod-1/.../mod-n.thp
```

If you use Theme-D without installing it you have to use command

```
MYPATH/theme-d-VERSION/translator/theme-d-link.scm
```

instead of `theme-d-link`. Here `MYPATH` is the path where you have unpacked Theme-D.

8 Running a Theme-D program

Suppose you have your linked Theme-D program in file `my-prog.go`. You can run this program with command

```
run-theme-d-program my-prog.go
```

If you need to import your own Scheme files into the Theme-D runtime environment (because of the foreign function interface) you can do this by defining the environment variable `THEME_D_CUSTOM_CODE`. Separate the file names with :'s. The program `run-theme-d-program` accepts the following arguments:

- `--no-verbose-errors` : No verbose information about errors (exceptions).
- `--pretty-backtrace` : Display pretty printed backtrace on error.

Note that the `--pretty-backtrace` option works only if you have linked your Theme-D program with option `--runtime-pretty-backtrace`.

If you use Theme-D without installing it you have to use command

```
MYPATH/theme-d-VERSION/runtime/run-theme-d-program.scm
```

instead of `run-theme-d-program`. Here `MYPATH` is the path where you have unpacked Theme-D.

The pretty printed runtime backtrace has the following format:

number kind name module

⋮

where *kind* is the kind of the called procedure, *name* is the name of the procedure and *module* is the module where the procedure has been defined. The *kind* may take the following values:

- `toplevel`: A toplevel procedure
- `local`: A local procedure
- `instance`: An instance of a parametrized procedure
- `zero`: A procedure used to generate the zero value of a class

If you have linked a program to Scheme code (with options `--intermediate-language=scheme` and `--keep-intermediate`) you can run the generated Scheme file with command

```
run-theme-d-program-scheme my-prog.scm
```

Note that the error option and the backtrace option are not supported with `run-theme-d-program-scheme`.

9 Distributing linked Theme-D programs

If your target environment has Theme-D installed it is sufficient to distribute only the linked `.go` file.

If you don't want to assume this you need to distribute the following files:

- `theme-d-VERSION/runtime/run-theme-d-program.scm`
- `theme-d-VERSION/runtime/runtime-theme-d-environment.go`
- `theme-d-VERSION/runtime/os-main.go`
- `theme-d-VERSION/runtime/theme-d-stdlib-support.go`
- `theme-d-VERSION/runtime/prt.go`
- `/etc/theme-d-config.scm` or file `.theme-d-config.scm` in your home directory.

If your target system doesn't use the optimization of the mathematical procedures you also need to distribute file `theme-d-VERSION/runtime/theme-d-alt-support.go`. The optimization is on by default. These files are licensed under GNU Lesser General Public License.

You have to ensure that the variable `gl-theme-d-runtime-dir` in the configuration file points to the directory where you install the runtime environment. Variable `gl-use-support-lib?` specifies whether you use the `libthemedsupport` library or not. Other configuration variables are not used by the runtime environment. If you use the support library the library `libthemedsupport` has to be installed in the target system. The use of the support library is recommended.

10 Compiling, Linking, and Running Test and Examples Programs

In order to install the Theme-D testing environment change to the directory where you want the environment to be installed and give command

```
setup-theme-d-test-env.sh
```

This directory shall be called *theme-d-test-dir* in the sequel. The test programs are located in subdirectory `test-env/theme-d-code/tests` and the example programs in `test-env/theme-d-code/examples`. Subdirectory `tools` contains scripts to run tests.

The example programs are built by giving command `make -f user.mk` in subdirectory `test-env/theme-d-code/examples`. The example programs are run with command `run-theme-d-program` *program*`.go`.

If `testX` is a program compile it with command

```
theme-d-compile -m :..  testX.thp
```

and link with command

```
theme-d-link -m :..  testX.tcp
```

in directory *theme-d-test-dir*`/test-env/theme-d-code/tests`.

If `testX` is a module compile it with commands

```
theme-d-compile -m :..  testX.thi
theme-d-compile -m :..  testX.thb
```

in directory *theme-d-test-dir*`/test-env/theme-d-code/tests`.

Note that some test programs import test modules in which case you must compile the modules before the program that uses them. When a test program imports several test modules compile first all the interfaces of the imported modules and then all the bodies of the imported modules. Compile the interfaces in the order they are numbered. Note also that some test programs require the examples to be built.

In order to run a test `testX` give commands

```
run-theme-d-program testX.go
```

in directory *theme-d-test-dir*`/test-env/theme-d-code/tests`.

If you want to build all the tests at once build the examples first. Then change to the directory *theme-d-test-dir*`/test-env/tools`. Compile the tests with command `./compile-tests.scm` and link them with command `./link-test-programs.scm`. Then run the linked programs with command `./run-test-programs.scm`. The test results can be checked with commands `check-test-compilation.scm`, `check-test-program-linking.scm`, and `check-test-runs.scm`, for compilation, linking and running, respectively. All these scripts are located in directory *theme-d-test-dir*`/tools`.

11 Other things

An Emacs mode for Theme-D can be found at `tools/theme-d.el`. There are some example programs in subdirectory `theme-d-code/examples` in the Theme-D source package. You can compile, link, and run them following the instructions given in sections 6, 7, and 8. If you install the Theme-D Debian package twice the configuration file `theme-d-config.scm` may not be installed. This problem is solved by uninstalling Theme-D and installing it again.

Theme-D translator uses the following notation for printing pair and tuple types: `(:pair` r s`)` is printed as { r . s } and `(:tuple` t_1 `...` t_n`)` is printed as {t_1 `...` t_n}. Note that this notation is not accepted in Theme-D code.

12 Comments

The linker requires that the compiled modules are placed in a proper subdirectory hierarchy under some directory among the module search directories. This condition is fulfilled if you define the module search directories to include all

the unit root directories used by your source files and put the compiled files into same directories with the source files.