uberSpark Documentation

Release Version: 5.0; Release Series: Chase

https://uberspark.org

Aug 30, 2019

CONTENTS:

Described below are details on the software requirements and dependencies, build, verification and intallation of the uberSpark core libraries and hardware model

CHAPTER

ONE

SOFTWARE REQUIREMENTS AND DEPENDENCIES

We assume your are working in: `/home/<home-dir>/<work-dir>`

Replace `<home-dir>` with your home-directory name and `<work-dir>` with any working directory of your choice.

1.1 Development OS and Base Packages

You will need a working Ubuntu 16.04.x LTS 64-bit environment for development and verification. This can either be a Virtual Machine (VM) (e.g., VirtualBox) or a container (e.g., Windows WSL). As of this writing, the Ubuntu 16.04.x LTS VM ISO image is available at:

```
http://releases.ubuntu.com/16.04/ubuntu-16.04.6-desktop-amd64.iso
```

You will need to first perform an `update` to download the latest package lists from the repositories as shown below:

```
sudo apt-get update
```

After the update completes, you will need to install the following base packages required for development as shown below:

```
sudo apt-get install git gcc binutils autoconf
sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 gcc-multilib
sudo apt-get install ocaml ocaml-findlib ocaml-native-compilers
sudo apt-get install graphviz libzarith-ocaml-dev libfindlib-ocaml-dev
sudo apt-get install make unzip
```

1.2 OCaml Compiler and Base Packages

You will then need to install the OCaml Package manager as shown below:

```
wget https://raw.github.com/ocaml/opam/master/shell/opam\_installer.sh -O - \| sh -s /
↪usr/local/bin
```

After the OCaml Package Manager installs successfully, configure the opam environment and switch to the appropriate OCaml compiler version as shown below:

```
eval ``opam config env``
opam switch 4.02.3
```

After the opam environment switch, install the following opam packages in order:

```
opam install menhir.20170712
opam install ocamlgraph.1.8.7
opam install ocamlfind.1.7.3
opam install zarith
opam install yojson
```

1.3 Coq Proof Assistant

The Coq Proof Assistant is a required package for both CompCert as well as Frama-C. You need to install the Coq Proof Assistant via opam as shown below:

```
opam install coq.8.6.1
```

1.4 CompCert Certified Compiler

The CompCert compiler is used to compile the C code for verified uberobjects within uberSpark. The Compcert version currently supported is v3.0.1 and can be installed as shown below:

```
wget http://compcert.inria.fr/release/compcert-3.1.tgz
tar -xvzf compcert-3.1.tgz
cd CompCert-3.1
./configure x86_32-linux
make all
sudo make install
cd ..
```

1.5 Frama-C Verification Framework

The Frama-C verification framework is used to discharge uberobject invariants and properties within uberSpark. The Frama-C version currently supported is `Phosphorus-20170501` and can be installed as shown below:

```
wget http://frama-c.com/download/frama-c-Phosphorus-20170501.tar.gz
tar -xvzf frama-c-Phosphorus-20170501.tar.gz
cd frama-c-Phosphorus-20170501
./configure
make
sudo make install
cd ..
```

You will also need to install Frama-C backend theorem provers such as CVC3, Alt-Ergo and Z3. The WP Frama-C plugin manual referenced below contains a chapter on installing the theorem provers:

```
http://frama-c.com/download/wp-manual-Phosphorus-20170501.pdf
```

Note that you will need to install the correct versions of Why3 and the provers as described in the aforementioned Frama-C WP plugin manual. For example, Why3 version 0.87.3 and Alt-ergo version 1.30. This can be done via opam as shown below in the context of Why3:

```
opam install why3.0.87.3
```

CHAPTER TWO

BUILDING AND INSTALLING UBERSPARK

2.1 Building uberSpark Tools

You will need to build the uberSpark toolchain before any other tasks. For this purpose, While in the top-level directory of the uberSpark repository, switch directory to uberSpark sources:

```
cd src
```

Then prepare for the build as below:

```
./bsconfigure.sh
./configure
```

And finally, build the toolchain:

```
make
```

2.2 Installing uberSpark

Upon a successful build, you will need to install the uberSpark toolchain, system headers and hardware-model related files. You can do this using the following command (while in the same directory of uberSpark sources `src/`):

```
sudo make install
```

CHAPTER

THREE

VERIFYING, BUILDING AND INSTALLING UBERSPARK LIBRARIES

The uberSpark core libraries provide commonly used runtime functionality for a uberobject. The core libraries current comprise a tiny C runtime library, a fledging crypto library (currently SHA-1 supported) and a library for platform hardware access.

This section of the documentation will describe how you can verify, build and install the aforementioned libraries. For the subsections that follow, you need to be in the top-level of the uberSpark source repository.

3.1 Verify uberSpark Libraries

To verify the uberSpark libraries first switch to the uberSpark library sources:

```
cd src/libs
```

Then prepare for verification and subsequent build:

```
./bsconfigure.sh
./configure
```

And, finally verify the libraries using:

```
make verify-ubersparklibs
```

Note: The verification typically takes a few minutes to complete and should finally terminate with a success message.

3.2 Build uberSpark Libraries

uberSpark uobject runtime libraries are built using the certified CompCert compiler in combination with uberSpark tools (to handle Assembly as CASM). To build the uberSpark libraries you need to use the following command:

```
make build-ubersparklibs
```

3.3 Installing uberSpark Libraries

uberSpark libraries are installed to the development system using the following command:

```
sudo make install
```

Note: The libraries are installed to the default location `/usr/local/uberspark` but can be over-ridden by the `--prefix` option during configuration using the `configure` utility.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search