SWISH Installer Documentation

Table of Contents

1 About

1.1 About

The purpose of this repository is to handle the installation and creation of serveral packages and helpers in order to install Cplint[1] on SWISH[2] with an Rserve[3] environment as well as the "vanilla" version of SWISH.

SWISH relies on SWI Prolog[4], a prolog interpteter.

1.1.1 Terminology

In order to understand this documentation correctly some terminology used here must be explained.

`#` means that the command must be executed by `root`.

`$` means that the command must be executed by the current user.

1.1.2 Directory listings

When you see something like the following, it represents a directory listing of the first named (head) directory[5]:

```
common/
|-- rserve-sandbox
|-- swish
‘-- swish-cplint
```

[1] See item [Cplint] in Chapter 7 [References], page 15.

[2] See item [SWISH] in Chapter 7 [References], page 15.

[3] See item [Rserve] in Chapter 7 [References], page 15.

[4] See item [Swipl] in Chapter 7 [References], page 15.

[5] These are generated from `$ tree --charset=ascii -d <dirname>`

2 Installation

2.1 Arch Linux

In order to have a fully functional installation of SWISH or Cplint on SWISH you either have to install the packages `swish` or `swish-cplint`.

There at least two possibilities to install both packages.

2.1.1 Using an AUR helper

Yaourt[1] is among the most popular AUR[2] helpers available. The following commands[3] will install all the dependencies automatically.

```
$ yaourt -Sa swish-cplint --noconfirm
```

or

```
$ yaourt -Sa swish --noconfirm
```

> **Note:** When you install Yaourt on Parabola GNU/Linux-libre[4] you will be asked to remove `your-freedom`.

2.1.2 Without using an AUR helper

2.1.2.1 Cplint on SWISH

Execute the following commands sequentially.

- Install `swi-prolog-devel`:

  ```
  $ wget "https://aur.archlinux.org/cgit/\
  aur.git/snapshot/swi-prolog-devel.tar.gz"
  $ tar -zxvf swi-prolog-devel.tar.gz
  $ cd swi-prolog-devel
  $ makepkg -sri --noconfirm
  ```

- Install `rserve-sandbox-docker`:

  ```
  $ wget "https://aur.archlinux.org/cgit/\
  aur.git/snapshot/rserve-sandbox-docker.tar.gz"
  $ tar -zxvf rserve-sandbox-docker.tar.gz
  $ cd rserve-sandbox-docker
  $ makepkg -sri --noconfirm
  ```

- Install `swish-cplint`:

  ```
  $ wget "https://aur.archlinux.org/cgit/\
  aur.git/snapshot/swish-cplint.tar.gz"
  $ tar -zxvf swish-cplint.tar.gz
  $ cd swish-cplint
  $ makepkg -sri --noconfirm
  ```

---

[1] See item [Yaourt] in Chapter 7 [References], page 15.

[2] See item [AUR] in Chapter 7 [References], page 15.

[3] See item [Packages on the AUR] in Chapter 7 [References], page 15.

[4] See item [Parabola] in Chapter 7 [References], page 15.

2.1.2.2 SWISH

- Install `swi-prolog-devel`:

```
$ wget "https://aur.archlinux.org/cgit/\
aur.git/snapshot/swi-prolog-devel.tar.gz"
$ tar -zxvf swi-prolog-devel.tar.gz
$ cd swi-prolog-devel
$ makepkg -sri --noconfirm
```

- Install `swish`:

```
$ wget "https://aur.archlinux.org/cgit/\
aur.git/snapshot/swish.tar.gz"
$ tar -zxvf swish.tar.gz
$ cd swish
$ makepkg -sri --noconfirm
```

2.1.3 Daemons Management

Arch Linux and derivative distros use Systemd[5] as the init system. First see Chapter 3 [Pre-run], page 4, and then see Chapter 4 [Running], page 5.

2.2 Debian

TODO

[5] See item [systemd] in Chapter 7 [References], page 15.

3 Pre-run

3.1 Cplint on SWISH

3.1.1 Docker

Before doing anything else you must start Docker[1] manually. See Chapter 4 [Running], page 5.

3.1.2 Rserve sandbox

To download the Docker image file as well as all dependencies for R, run the following command:

```
$ sudo -u rsd rserve-sandbox-docker -i
```

3.1.3 Cplint on SWISH

To download all the prolog packages necessary in order to run Cplint on SWISH correcly:

```
$ sudo -u swish swish-cplint -i
```

3.2 SWISH

The vanilla version of swish doesn't need a pre-run.

[1] See item [Docker] in Chapter 7 [References], page 15.

4 Running

4.1 Systemd

4.1.1 Cplint on SWISH

4.1.1.1 Start Docker

Docker is a dependency which must be started manually.

```
# systemctl start docker
```

4.1.1.2 Start

The following command will run `swish-cplint` as well as `rserve-sandbox-docker` as its dependency:

```
# systemctl start swish-cplint
```

4.1.1.3 Enable

To start `swish-cplint` and `rserve-sandbox-docker` at boot:

```
# systemctl enable swish-cplint
```

4.1.1.4 Stop

You can stop both services[1] with:

```
# systemctl stop swish-cplint
# systemctl stop rserve-sandbox-docker
```

4.1.1.5 Status

To check the status:

```
# systemctl status swish-cplint
# systemctl status rserve-sandbox-docker
```

4.1.2 SWISH

Running and managing SWISH alone is simpler.

4.1.2.1 Start Docker

```
# systemctl start docker
```

4.1.2.2 Start

```
# systemctl start swish
```

4.1.2.3 Enable

```
# systemctl enable swish
```

[1] stopping `swish-cplint` does not imply that `rserve-sandbox-docker` will be stopped

4.1.2.4 Stop

```
# systemctl stop swish
```

4.1.2.5 Status

```
# systemctl status swish
```

5 Development

5.1 Repository structure

```
.
|-- common
|   |-- rserve-sandbox
|   |-- swish
|   `-- swish-cplint
`-- distributions
    |-- archLinux-based
    |   |-- rserve-sandbox
    |   |-- swish
    |   `-- swish-cplint
    `-- debian-based
```

The `distributions` directory all the files useful to build packages for a specific distribution. Makefiles are used to achieve this.

The `common` directory contains all the files which are not distribution specific.

5.1.1 Distribution-specific files

See Chapter 5 [Building the packages], page 7,

5.1.2 Common files

```
common/
|-- rserve-sandbox
|   `-- run.sh
|-- shared_functions.sh
|-- swish
|   |-- run.pl
|   `-- run.sh
`-- swish-cplint
    |-- install_web_iface_deps.pl
    |-- run.pl
    `-- run.sh
```

5.1.2.1 run.sh

The `run.sh` files are helpers to start and stop the daemons. These helpers should work on any distribution. You can edit the variables at the top of each file accordingly.

5.1.2.2 run.pl

The `run.pl` files are a modified version of the original files with the same name. These have been created in order to lauch SWISH as a background program. Without the changes contained in those files, you couldn't run SWISH in the background.

5.1.2.3 shared_functions.sh

Another important file is `shared_functions.sh` which contains all common functions for the `run.sh` files. It must be appended to every `run.sh`, within a Makefile for example with the following:

```
$ cat shared_functions.sh >> {rserve-sandbox,swish,swish-cplint}/run.sh
```

5.1.2.4 install_web_iface_deps.pl

Cplint on SWISH contains a post installation script called `install_web_iface_deps.pl`. This script, called by `run.sh`, simply installs all the necessary Prolog dependencies automatically.

5.2 Guidelines to create packages

Makefiles are used to build the package in a new directory by copying all necessary files (also from the `common` directory). Each Makefile is distribution-based specific.

You will read general information about the packages, all their install and remove actions and how to build your version of the modified packages.

The install and remove actions have to be done sequentially.

5.2.1 Rserve sandbox

5.2.1.1 General information

- Name

  `rserve-sandbox-docker`
- Data directory

  `/usr/share/rserve-sandbox-docker`
- Dependencies
  - R
  - Docker

5.2.1.2 Install actions

- Pre
  - None
- During
  - Make a symbolic link to be able to call `rserve-sandbox-docker` from `/usr/bin`

    ```
    $ ln -s /usr/share/rserve-sandbox-docker/run.sh \
    /usr/bin/rserve-sandbox-docker
    ```
- Post
  - Add `rserve` user and group.

    ```
    # getent group rserve &>/dev/null || groupadd -r rserve >/dev/null
    # getent passwd rserve &>/dev/null || useradd -m -d /home/rserve \
    -s /bin/false -r -g rserve rserve >/dev/null
    # chmod 750 /home/rserve
    ```

  - Add `rsd` user and group.

    ```
    # getent group rsd &>/dev/null || groupadd -r rsd >/dev/null
    # getent passwd rsd &>/dev/null || useradd \
    -s /bin/false -r -g rsd rsd >/dev/null
    ```

  - Add the new user to the ‘docker‘ group

    ```
    # gpasswd -a rsd docker >/dev/null
    ```

  - Change ownership of the package data directory

    ```
    # chown -R rsd:rsd /usr/share/rserve-sandbox-docker
    ```

5.2.1.3 Remove actions

- Pre
  - None
- During
  - None
- Post
  - Tell the user that `/home/rserve`, `rsd` user and group, `rserve` user and group and all the Docker files can be removed (this depends on the package remove policies of the chosen distro).

5.2.2 Cplint on SWISH

5.2.2.1 General information

- Name

  `swish-cplint`
- Data directory

  `/usr/share/swish-cplint`
- Dependencies
  - SWI Prolog (developement version)
  - Git
  - libXinerama
  - libXpm
  - Rserve sandbox
  - Bower (make dependency)

5.2.2.2 Install actions

- Pre
  - Compile the server

    ```
    $ bower --allow-root install
    $ make src
    ```

  - Copy `run.pl`, `run.sh` and `install_web_iface_deps.pl` in SWISH's root directory.

- During
  - Make a symlink to be able to call `swish-cplint` from `/usr/bin`

    ```
    $ ln -s /usr/share/swish-cplint/run.sh /usr/bin/swish-cplint
    ```
- Post
  - Add `swish` user and group

    ```
    # getent group swish &>/dev/null || groupadd -r swish >/dev/null
    # getent passwd swish &>/dev/null || useradd -m -d /home/swish \
    -r -g swish swish >/dev/null
    ```
  - Add `swish` user to the previously created `rserve` group.

    ```
    # gpasswd -a swish rserve >/dev/null
    ```
  - Change ownership of the package data directory

    ```
    # chown -R swish:swish /usr/share/swish-cplint
    ```

5.2.2.3 Remove actions

- Pre
  - None
- During
  - None
- Post
  - Tell the user that `/home/swish` and `swish` user and group can be removed (this depends on the package remove policies of the chosen distro).

5.2.3 SWISH

5.2.3.1 General information

- Name

  `swish`
- Data directory

  `/usr/share/swish`
- Dependencies
  - SWI Prolog (developement version)
  - libXinerama
  - libXpm
  - Bower (make dependency)

5.2.3.2 Install actions

- Pre
  - Compile the server

    ```
    $ bower --allow-root install
    $ make src
    ```
  - Copy `run.pl` and `run.sh` and in SWISH's root directory.

- During
  - Make a symlink to be able to call `swish` from `/usr/bin`

    ```
    $ ln -s /usr/share/swish/run.sh /usr/bin/swish
    ```
- Post
  - Add `swish` user and group

    ```
    # getent group swish &>/dev/null || groupadd -r swish >/dev/null
    # getent passwd swish &>/dev/null || useradd -m -d /home/swish \
    -r -g swish swish >/dev/null
    ```
  - Change ownership of the package data directory

    ```
    # chown -R swish:swish /usr/share/swish
    ```

5.2.3.3 Remove actions

- Pre
  - None
- During
  - None
- Post
  - Tell the user that `/home/swish` and `swish` user and group can be removed (this depends on the package remove policies of the chosen distro).

5.3 Building the packages

5.3.1 Arch Linux

```
archLinux-based/
|-- Makefile
|-- rserve-sandbox
|   |-- .install
|   |-- PKGBUILD
|   `-- rserve-sandbox-docker.service
|-- swish
|   |-- .install
|   |-- PKGBUILD
|   `-- swish.service
`-- swish-cplint
    |-- .install
    |-- PKGBUILD
    `-- swish-cplint.service
```

Once you've made changes you can run `$ make` then change directory into one of the new `.aur` generated directories and finally run `$ makepkg -sri` to install the package.

5.4 Help pages

5.4.1 Rserve sandbox

```
rsd [OPTION]
Docker spec for running Rserve in a sandbox

Only a single option is permitted.
    -h      print this help
    -i      install dependencies
    -k      kill rserve-sandbox-docker
    -s      start rserve-sandbox-docker

Exit status:
 0  if OK,
 1  some error occurred.

Full documentation at: <https://github.com/frnmst/rserve-sandbox>
```

5.4.2 Cplint on SWISH

```
swish-cplint [OPTION]
SWI-Prolog for SHaring: a SWI-Prolog web IDE integrated with the cplint suite

Only a single option is permitted.
    -h      print this help
    -i      install dependencies
    -k      kill swish-cplint
    -s      start swish-cplint

Exit status:
 0  if OK,
 1  some error occurred.

Full documentation at: <https://github.com/friguzzi/swish>
and at: <https://github.com/friguzzi/cplint>
```

5.4.3 SWISH

```
swish [OPTION]
SWI-Prolog for SHaring: a SWI-Prolog web IDE

Only a single option is permitted.
    -h      print this help
    -k      kill swish
    -s      start swish

Exit status:
 0  if OK,
 1  some error occurred.
```

Full documentation at: <https://https://github.com/SWI-Prolog/swish>

6 Thanks

I want to thank the SWI Prolog, Arch Linux and Systemd communities as well the authors of the free software used here, which made the creation of these packages possible.

I also want to thank Fabrizio Riguzzi which tested the packages and gave me advices about them.

7 References

Some quotations reported here are taken directly from the respective web sites.

[Cplint] A suite of programs for reasoning with probabilistic logic programs. See `https://github.com/friguzzi/cplint`

[SWISH] A web browser interface for SWI Prolog to share code. See `https://github.com/SWI-Prolog/swish` for the original version and `https://github.com/friguzzi/swish` for the version made by Fabrizio Riguzzi that uses Cplint.

[Rserve] A docker image that enables to use the R and Rserve environment in a secure way. See `https://github.com/JanWielemaker/rserve-sandbox` for the original version made by Jan Wielemaker and `https://github.com/frnmst/rserve-sandbox/tree/distro-package` version by Franco Masotti which is used here. A client to access Rserve from Prolog is also necessary. See `https://github.com/JanWielemaker/rserve_client`

[Swipl] SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications. See `http://www.swi-prolog.org/` and `https://github.com/SWI-Prolog/swipl-devel` which is the developement version used here.

[Yaourt] A pacman wrapper with extended features and AUR support. To install Yaourt follow the instructions reported on `https://archlinux.fr/yaourt-en`.

[AUR] The Arch User Repository (AUR) is a community-driven repository for Arch users. It contains package descriptions (PKGBUILDs) that allow you to compile a package from source with makepkg and then install it via pacman. The AUR was created to organize and share new packages from the community and to help expedite popular packages' inclusion into the community repository. See `https://aur.archlinux.org/` for the AUR homepage and `https://wiki.archlinux.org/index.php/Arch_User_Repository` for a complete explanation.

[Parabola] A fully free, simple, and lightweight operating system. See `https://parabola.nu`.

[systemd] systemd is a suite of basic building blocks for a Linux system. It provides a system and service manager that runs as PID 1 and starts the rest of the system. See `https://www.freedesktop.org/wiki/Software/systemd/`.

[Docker] Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries anything that can be installed on a server. This guarantees that the software will always run

the same, regardless of its environment. See https://www.docker.com/.

[Packages on the AUR] Here follows a list to the AUR packages:

- SWI Prolog (developement version) https://aur.archlinux.org/packages/swi-prolog-devel
- Rserve sandbox https://aur.archlinux.org/packages/rserve-sandbox-docker/
- Cplint on SWISH https://aur.archlinux.org/packages/swish-cplint/
- SWISH https://aur.archlinux.org/packages/swish/