Tools Configuration

Mozart - Using the NTNU Software Farm

The **Mozart** Programming System is an implementation of the **Oz** language. Most of the course is based on Oz. Mozart will be used for a large part of the exercises and assignments, and it is important that you get acquainted with it as soon as possible.

Mozart can be run through the NTNU "Software Farm", which gives students and employees access to many different software programs, without having to install them.

The Software Farm can be accessed using the Remote Desktop Protocol (RDP), and it is possible to access is on either Windows, Linux, or Mac systems, and even on Android. Instructions to access the software farm can be found here. The NTNU Software Farm has several sections. Mozart can be found in the "Developer" section (**devfarm.ntnu.no**).

It is also possible to access the Software Farm through a web browser, provided that it supports HTML5. The drawback of this option is that it is more difficult to work with multiple files and to save/read files. However, note that when using the Software Farm you have access to your home directory at NTNU, and you can find it under "This PC" and then "Hjemmeområdet".

Note that the Software Farm can be accessed only from within the NTNU network, or through the VPN.

Mozart - Local installation

In case you want to install Mozart locally, here is some useful information.

Windows

A Windows installer is provided as artifact in the GitHub repository of Mozart2. Just download it and run the installer.

However, to work properly, the Mozart environment requires Emacs to be installed as well. Links to a windows installer for Emacs can be found here. After installing Emacs, you need to set the OZEMACS environment variable to the executable from your Emacs installation.

To edit system variables in Windows 10, first search for "Edit the system environment variables" and then select "Environment variables...". There, add a new variable (either for your user or a global one). The variable should be named "OZEMACS" and it should have the path of the Emacs executable as its value, for example "C\Program Files\Emacs\emacs-28.1\bin\runemacs.exe"

Linux

Debian/Ubuntu (.deb) and Red Hat (.rpm) packages are provided as artifacts in the GitHub repository of Mozart2. Just download the one appropriate for your distribution, and install it. I have tested Mozart2 with Debian 11, and it works properly.

Unfortunately the dependency to Emacs is not set in the package, and so it has to be installed manually. Emacs is typically available in any linux package manager (e.g., apt-get install emacs).

MacOS

On MacOS is where people are experiencing most problems. These are some options:

- Use the "Dorothy" docker image kindly provided by one of your colleagues from previous year (see post below).
- Use the .dmg package provided as artifact in the GitHub repository of Mozart2. You might have to modify security policies on the operating system for this to work.
- Use the Homebrew package manager, and follow the instructions below.

Installing Mozart2 locally on macOS with Homebrew:

1) Make sure you have installed Homebrew (https://brew.sh/), a really nice package manager.
2) Run the command `brew tap dskecse/tap` in the terminal.
3) Run the command `brew install --cask mozart2` in the terminal.
4) Open up .zshrc (should be located in your home path, ~)
5) Add the following lines:

```
alias ozc='/Applications/Mozart2.app/Contents/Resources/bin/ozc'
alias ozemulator='/Applications/Mozart2.app/Contents/Resources/bin/ozemulator'
alias ozem='/Applications/Mozart2.app/Contents/Resources/bin/ozemulator'
alias ozengine='/Applications/Mozart2.app/Contents/Resources/bin/ozengine'
alias ozwish='/Applications/Mozart2.app/Contents/Resources/bin/ozwish'
```

6) Now you should be able to compile code with `ozc -c file.oz` and run them from the terminal with `ozem file.ozf`. Where .ozf is the compiled .oz file.

Mozart - Through a Docker Container

Dorothy (named after the protagonist of Frank Baum's Oz novels) is a Docker image for running Mozart-Oz and the OPI from within a container, either standalone or as part of a CDE (Containerised Development Environment) - e.g. using VSCode's Remote - Containers extension.

Mozart has been known to suffer serious compatibility flaws on certain systems - notably code that compiles and runs flawlessly on other machines failing to compile on those running non-Debianesque Linux distros or macOS.

This appears to be the most reliable version to run Oz/Mozart on macOS machines.

Kindly provided by Richard.

Using VSCode with Oz

There is also an Oz extension for VS Code, which works quite well and has some code completion features.

However, **you still need to have the Mozart platform installed** on your machine, because execution is still through the Oz emulator provided with it. We recommend that you familiarize yourself with the standard Oz/Mozart platform before starting using the VSCode plugin. Otherwise you will not be able to understand error messages and other problems that will happen.

Note that execution through VSCode is not performed as you might be used to with other programming languages, but instead you must right-click on the source file and select "Feed File". You still have the two buffers "Oz Compiler" and "Oz Emulator", which show the output of the compiler and of the runtime environment, respectively.