Example for module files

The module system makes setting up different versions of software much easier to maintain. Module files can be very simple, or very complicated. All they have to do though is to set the appropriate paths and variables for the execution. I have used modified versions from CLAIX18 (the RWTH Aachen cluster), which I have cleaned for this repository.

Root module

The root module is the one doing all the work, defining the module and setting the environment.

The variable `PROJECTHOME` needs to be adjusted to the home of where the software is. If it is your own home directory, you can also use `$::env(HOME)` as an argument.

The installation variable `XTBHOME` also has to be adjusted accordingly. It currently assumes that the software is stored in a directory `${PROJECTHOME}/local/$modulename/$modulename-$version`, which could resolve to `/home/polyluxus/local/xtb/xtb-6.2.3` for the most recent version of xtb. The variable `version` will be set from the version module, see below.

```
#%Module1.0###-*-tcl-*-##############################################################
##
## xtb modulefile
## https://www.chemie.uni-bonn.de/pctc/mulliken-center/software/xtb
###

# Naming the module
set modulename "xtb"

# Home directory of the installation
set PROJECTHOME "/home/polyluxus"

# Define local variable with path to installation software
# version will be set by referring module file
set XTBHOME "${PROJECTHOME}/local/$modulename/$modulename-$version"

# This is the help output, printed by "module help"
proc ModulesHelp { } {
  # define the global variables version and modulename in this function
  global version
  global modulename
  puts stderr "*** This module initialises the $modulename $version environment ***"
  puts stderr "    An extended tight-binding semi-empirical program package        "
  puts stderr "    Please contact xtb@thch.uni-bonn.de for details.                "
  puts stderr "    See also: https://github.com/grimme-lab/xtb                     "
}

# Short description (preferably 1 line) what the loaded software does,
# or what the module is good for.
# Printed by "module whatis":
module-whatis "$modulename - extended tight-binding semi-empirical program package"

# If module should be loaded, check for conflicts and print info
switch [module-info mode] {
  load {
    # Is this module already loaded?
    set conflict "$modulename/$version"
    if { [is-loaded $conflict]} {
      puts stderr "$conflict already loaded, doing nothing."
      return
    }
```

```
    # Is a different version already loaded? (This would check xtb6)
    set conflict $modulename
    if { [is-loaded $conflict]} {
      puts stderr "$conflict already loaded; conflicts with $modulename/$version."
      puts stderr "Try unloading $conflict first."
      exit
    }
    # Check if the software is really installed, if not error and abort.
    if { ![file isdirectory $XTBHOME] } {
      puts stderr "$modulename is not installed on this machine."
      exit
    }
    # Nothing failed, print a success message:
    puts stderr "Loading $modulename $version"
  }
  unload {
    puts stderr "Unloading $modulename $version"
  }
}

# XTBHOME is the root directory of the loaded xtb version
setenv        XTBHOME  $XTBHOME

# Recommendations for loading:
prepend-path  XTBPATH  $::env(HOME)
prepend-path  XTBPATH  $XTBHOME
prepend-path  XTBPATH  ${XTBHOME}/share/xtb

# The following paths need to be set/adjusted for the 6.0 distributions
prepend-path  PATH     $XTBHOME/bin
# Old MANPATH (~ 6.0)
if { [file isdirectory $XTBHOME/man] } {
  prepend-path  MANPATH  $XTBHOME/man
}
# New MANPATH (~ 6.2.x)
if { [file isdirectory $XTBHOME/share/man] } {
  prepend-path  MANPATH  $XTBHOME/share/man
}
# Include libraries
if { [file isdirectory $XTBHOME/lib] } {
  prepend-path  LD_LIBRARY_PATH     $XTBHOME/lib
}
# Include the scripts directory, if it exists (very old, or very custom)
if { [file isdirectory $XTBHOME/scripts] } {
  prepend-path  PATH     $XTBHOME/scripts
}
# Include the python directory, if it exists
if { [file isdirectory $XTBHOME/python] } {
  prepend-path  PATH        $XTBHOME/python
  prepend-path  PYTHONPATH  $XTBHOME/python
}
```

Version module

The version module is only necessary to set the `version` variable, so that the root module will find the correct software path.

The variable `module_base_path` needs to be adjusted to point to the directory of the root module.

```
#%Module1.0###-*-tcl-*-############################################################
##
## XTB modulefile
##

# For the local install
set module_base_path "/home/rwth0425/modules/source"

# Needs to be adjusted for what is to be loaded
set MAJORVERSION "6"
set MINORVERSION "2"
set REVISION     "3"

# Will be read by the source module file
set version "$MAJORVERSION.$MINORVERSION.$REVISION"

# Load the source module file
source "$module_base_path/xtb6/xtb6"
```

If no version control is wanted, one module with hard-coded paths should also do the trick.

Disclaimer: I currently have no system to check whether these files work, I tried to incorporate some of the changes in the setup given in the xtb manual. You'll get the gist, use them as templates and review every line before use.