Installation guide for mk modules

Modules are available both in source and binary packages. This guide cover the installation of binary distribution.

Installing VirtualBox

Installation of VirtualBox depends on your operating system. A binary package for the most common platform is available and it can be download from VirtualBox downloads page.
Even if you already installed VirtualBox I suggest to check if it is updated to the latest version.

Installing Vagrant

Also Vagrant is distributed in binary packages and they are available for almost all operating systems. It is enough to get the appropriate installer from its download page.

Configuring Vagrant

Unfortunately, the Windows platform lacks of some tools, like an ssh client. I suggest to install CMDER to continue, CMDER is a console emulator with POSIX-like built-in functionalities.

Once CMDER is launched, the command `vagrant` is available as `vagrant.exe`.

The first step is the virtual machine setup. Vagrant gives you an easy way to access to a lot of virtual machines already configured.

At first you have to create a directory where to store the new virtual machine, now, from the terminal emulator, move inside the just created directory (with the command `cd`).
In order to keep the virtual machine up to date with the host use the Vagrant plugin vbguest, it can be installed using the command:

```
vagrant plugin install vagrant-vbguest
```

The command `init` is used to create a new virtual machine, obviously all the modules are developed on Linux and I suggest to use Debian as distribution for your setup, then run the commands:

```
vagrant init -m debian/jessie64
```

Debian jessie is actually an old distribution but very well tested with this system, you may want to use a newer distribution instead, *e.g.*:

```
vagrant init -m debian/stretch64
```

The virtual machine is configured to work inside the directory where it is located. During the initialization a file called `Vagrantfile` is created. It stores the configuration of the virtual machine and it is a simple text file (it is a Ruby script) that can be edited with any type of editor.

By default Vagrant creates a new virtual machine with 1GB of RAM and synchronises the current directory with the directory `/vagrant` in the virtual environment. The synchronisation is done by means of `rsync`, it is not available on Windows and it is not in realtime on the other platforms.

For this reason I suggest to use a different configuration and you can replace the contents of `Vagrantfile` by:

```
Vagrant.configure("2") do |config|
  config.vm.box = "debian/jessie64"
  config.vm.synced_folder ".", "/vagrant", type: "virtualbox"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
  end
end
```

The number `2048` is the amout of RAM for the virtual machine in MB. The minimum value that I suggest is `1024`, but if you need to compile a large C++ project it is a good idea to increase to `4096` and configure the compiler to avoid pipes (`-pipe` flag).
If you like to use a different distribution, you can change `debian/jessie64` with the name of another distribution. A searchable list of all the available virtual machine for Vagrant can be inspected on Atlas.

Remember, a virtual machine is like a real computer, to turn on the virtual machine use the command:

```
vagrant up
```

You can stop it with `vagrant halt` or you can suspend it with `vagrant suspend`. The command `vagrant status` can be used to check if the virtual machine is already running or not.
Once the virtual machine is running you can login into with:

```
vagrant ssh
```

Installing mk

Finally the actual installation of the the environmental module can be done from within the vagrant virtual environment using the following procedure:

First download the whole modules distribution from this link

save the downloaded file in the Vagrant shared directory, then start Vagrant and type

```
sudo useradd -u 1001 -U -M swadmin
cd /vagrant
sudo tar xvjf modules_apsc.tar.bz2 -C /
```

Activating mk

To check whether the installation worked correctly, try activating mk by means of the following commands:

```
. /u/sw/etc/bash.bashrc
module avail
```

you should see the following if everything worked correctly:

```
------------------------- /u/sw/modules/toolchains -------------------------
   gcc-glibc/7
Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching
any of the "keys".
```