Università della Svizzera italiana

Faculty of Informatics

Operating Systems
Spring 2025
PintOS Installation Instructions

Prof. Fernando Pedone
Assisted by:
Eliã Batista, Lorenzo Martignetti

Summary

The objective of this document is to help you setup an environment in which you can edit, compile and test the Pintos operating system, which will be used throughout the course. Follow the instructions presented here and, if you have problems, contact the TAs via e-mail or through the forums on iCorsi.

Rationale

To make it easier for you and so that everyone has the same environment, we decided on providing a virtual machine. This machine is already configured with what you need to compile, run and debug Pintos. You will still be able to edit the source code from outside the VM, using whatever text editor you prefer.

Required software

Virtual Box

You will need to download and install an up-to-date version of VirtualBox.

Download it from `https://www.virtualbox.org/wiki/Downloads`.

On a Mac, you can run:

```
$ brew cask install virtualbox
```

if you have homebrew[1] installed.

Vagrant

We will not use Virtual Box directly. To create and start a virtual machine we will use a tool that automates that for us, called Vagrant.

Download it from `https://www.vagrantup.com/downloads.html`.

Again, if you have homebrew, you can simply run:

```
$ brew cask install vagrant
```

To check it was installed correctly, in a terminal, run:

[1] `https://brew.sh`

```
$ vagrant version
```

It should tell you its version and if it is up-to-date.

1 Starting the Virtual Machine

The first thing we have to do is download the project archive from iCorsi. Done that, extract the archive somewhere:

```
$ tar xzvf pintos0.tar.gz
```

After extracting, you will have a folder called `pintos-env`. This folder contains all the pintos source code (inside `pintos-env/pintos`). The `pintos-env` folder will also be available from inside the virtual machine.

To start the virtual machine up, do:

```
$ cd pintos0     # go into folder extracted from the .tar.gz archive
$ vagrant up     # start the VM. this will take a while the first time...
```

The first time you execute `vagrant up`, Vagrant will download a base Ubuntu image. It will then start Virtual Box and do all the setup required. When the command finishes, we will be able to `ssh` into the virtual machine using:

```
$ vagrant ssh    # ssh into the running virtual machine
```

To stop the virtual machine, you have two options:

- `vagrant suspend` - save the machine state to disk and suspend (uses more disk space)
- `vagrant halt` - shutdown the machine (takes more time to boot up again)

2 Working with the Virtual Machine

The virtual machine is a basic linux distribution (Ubuntu) and already has everything required for the assignments. When you start it up using Vagrant, the `pintos-env` folder that you extracted will be **automatically synchronized** between your (real) machine and the running virtual machine. You can edit the pintos source code using your favourite editor, outside the VM, and the changes will be available inside the VM. When you ssh into the virtual machine, the pintos code will be at `/pintos-env` (notice the / before the name).

2.1 Compiling Pintos

Compiling and running pintos is done from inside the virtual machine. To check that everything is working as intended, `vagrant ssh` into the VM and run:

```
$ cd /pintos-env/pintos/threads    # go into the threads project folder
$ make clean                       # clear old binaries if any
$ make                             # compile pintos
$ make check                       # run the unit tests
```

After running `make check` you should see that **20 of 27 tests failed**. Throughout the course we will implement new functionalities and some of these tests will start passing.

2.2 Running tests

There are many tests. See `pintos/tests/threads`. To run a single test, do:

```
$ cd /pintos-env/pintos/threads
$ pintos -v -- run alarm-single   # run the alarm-single test
```

To run all tests in one of the project folders (`threads`, `userprog`, `vm` or `filesys`):

```
$ cd /pintos-env/pintos/threads   # go into one of the projects
$ make
$ make check
```

2.3 Debugging

Debugging will also be done from inside the virtual machine. To debug, we are going to use two terminals, both connected to the VM. In one of the terminals, we will start a test in debug mode:

```
$ cd /pintos-env/pintos/threads
$ pintos --gdb -- run alarm-single     # this will start in debug mode and wait for a connection
```

From the other terminal, we will start GDB and connect to the running pintos

```
$ cd /pintos-env/pintos/threads
$ pintos-gdb build/kernel.o      # start gdb
gdb> debugpintos                 # connect to the running pintos
```

The `debugpintos` command is just a macro to save some typing - it is equivalent to doing:

```
gdb> target remote localhost:1234    # connect to the running pintos
```

Appendix E on the pintos documentation (available in the iCorsi page) explains how to use GDB and other debugging techniques.