# Pintos Installation Guide

This document is based on the information provided in the following links:

- Pintos introduction
- Installing Pintos
- Install Pintos with Qemu
- You can download pintos from this link.

## Introduction

Pintos is a simple operating system framework for the 80x86 architecture. It supports kernel threads, loading and running user programs, and a file system, but it implements all of these in a very simple way. In the Pintos projects, you and your project team will strengthen its support in all three of these areas. You will also add a virtual memory implementation.

Pintos could, theoretically, run on a regular IBM-compatible PC. However, this is not a realistic option. Therefore, we will run Pintos projects in a system simulator, that is, a program that simulates an 80x86 CPU and its peripheral devices accurately enough that unmodified operating systems and software can run under it. In class we will use the Bochs and QEMU simulators. Pintos has also been tested with VMware Player.

This document explains how to get started working with Pintos. You should read the entire document and perform the required setup before you start working on any phase of the project.

## Downloading Pintos

Download Pintos and unzip it in your home directory. You better review the different directories in the pintos directory in the Source Tree Overview.

In the rest of this document, we use the following two variables:

- $HOME: refers to your user home, for example, /home/yourname.
- $PINTOSHOME: refers to the path at which you unizp the pintos project, for example, /home/yourname/osF14/pintos.

Notes:

- You better use the absolute paths in all the pintos files that we need to change.
- An example download command:
  $ wget http://web.stanford.edu/class/cs140/projects/pintos/pintos.tar.gz

## Installation Prerequisite

**The Pintos development environment is targeted at Unix-like systems. It has been most extensively tested on GNU/Linux, in particular the Debian and Ubuntu distributions, and Solaris. It is not designed to install under any form of Windows.**

Prerequisites for installing a Pintos development environment include the following, on top of standard Unix utilities:

- Check whether your machine is 32-bit or 64 bit. You can use the command: `$ uname -m`. If the output is `i686` or `i386`, then the machine is 32-bit. If it is `x86_64`, then it is 64-bit.
- You first need to update the utility that you are using for download. For example, you will use apt-get on ubuntu, and therefore, the command you should use is: `$sudo apt-get update`
- Required: GCC. Version 4.0 or later is preferred. Version 3.3 or later should work. If the host machine has an 80x86 processor, then GCC should be available as gcc; otherwise, an 80x86 cross-compiler should be available as i386-elf-gcc.
  You will need to use the following command (ubuntu users): `$sudo apt-get upgrade gcc`.
- Required: GNU binutils. Pintos uses addr2line, ar, ld, objcopy, and ranlib. If the host machine is not an 80x86, versions targeting 80x86 should be available with an i386-elf- prefix.
  You will need to use the following command (ubuntu users): `$sudo apt-get install binutils`.
- You will also need to install the following packages:
  - `$ sudo apt-get install pkg-config`
  - `$ sudo apt-get install zlib1g-dev`
  - `$ sudo apt-get install libglib2.0-dev`
  - `$ sudo apt-get install gcc libc6-dev`
  - `$ sudo apt-get install autoconf`
  - `$ sudo apt-get install libtool`
  - `$ sudo apt-get install libsdl1.2-dev`
  - `$ sudo apt-get install g++`
  - `$ sudo apt-get install libx11-dev`
  - `$ sudo apt-get install libxrandr-dev`
  - `$ sudo apt-get install libxi-dev`

- Required: Perl. Version 5.8.0 or later is preferred. Version 5.6.1 or later should work.
  You will need to use the following command (ubuntu users): sudo apt-get upgrade perl.
- Recommended: GDB is helpful in debugging (see section E.5 GDB). If the host machine is not an 80x86, a version of GDB targeting 80x86 should be available as i386-elf-gdb. Command: `$sudo apt-get install libc6-dbg gdb`.
- Required: GNU make, version 3.80 or later.
  You will need to use the following command (ubuntu users): sudo apt-get install make.
- Recommended: QEMU, version 0.11.0 or later. If QEMU is not available, Bochs can be used.
- Recommended: X. Being able to use an X server makes the virtual machine feel more like a physical machine, but it is not strictly necessary.
- Optional: Texinfo, version 4.5 or later. Texinfo is required to build the PDF version of the documentation.
- Optional: TeX. Also required to build the PDF version of the documentation.
  You will need to use the following command (ubuntu users): sudo apt-get install texlive-full.
- Optional: VMware Player. This is a third platform that can also be used to test Pintos.

## Instructions for Installing Qemu

You can do the following to install Qemu on your machine:

- `$ sudo apt-get install git`
- `$ mkdir $HOME/apps`
- `$ cd $HOME/apps`
- `$ git clone git://git.qemu-project.org/qemu.git`
- `$ cd $HOME/apps/qemu`
- `$ ./configure --target-list=x86_64-softmmu --disable-werror --enable-debug` (for 64-bit machines)
  or
  `$ ./configure --target-list=i386-softmmu --disable-werror --enable-debug` (for 32-bit machines).

- If you get this error "pixman >= 0.21.8 not present.", execute the following command and repeat the configuration step. `$ git submodule update --init pixman`
- if you get this error "DTC (libfdt) version >= 1.4.0 not present.", execute the following command and repeat the configuration step. `$ git submodule update --init dtc`
- `$ make`
- `$ sudo make install`
- For 64-bit machines: `$ ln -s $HOME/apps/qemu/x86_64-softmmu/qemu-system-x86_64 /bin/qemu`
- For 32-bit machines: `$ ln -s $HOME/apps/qemu/i386-softmmu/qemu-system-i386 /bin/qemu`

## Instructions for Installing Bochs

You can do the following to install Bochs on your machine:

- `$ cd $HOME/apps`
- Get the bochs latest release:
  `$ wget http://sourceforge.net/projects/bochs/files/bochs/2.6.8/bochs-2.6.8.tar.gz`
- `$ tar -xzvf bochs-2.6.8.tar.gz`
- `$ cd $HOME/apps/bochs-2.6.8`
- copy "bochs-fix.patch" to "bochs" directory
- `$ cat ../bochs-fix.patch | patch -p1`
- Edit the file
  `.conf.linux` to comment `which_config=plugins` and uncomment `which_config=normal`.
- `$ ./.conf.linux`
- `$ make`
- `$ sudo make install`

The above will guarantee that you run a GUI interface for bochs. However, you will need to change it as follows to get the pintos `make check` run properly on your machine.

- Edit the file `.conf.linux` to add –with-nogui to the ./configure command.

- `$ ./.conf.linux`
- `$ make`
- `$ sudo make install`

## Instructions for Installing Pintos

1. **Updating the PATH Environment Variable:**
   To be able to use the pintos utils "backtrace", "pintos", "pintos-gdb", "pintos-mkdisk", "pintos-set-cmdline", and "Pintos.pm", you need to add their pathes to PATH. You need to add the following line in the $HOME/.bashrc file and restart the terminal:
   PATH=$PINTOSHOME/src/utils:$PATH (Do not forget to replace $PINTOSHOME with the actual path to your pintos directory).

   Note that you will need to restart your machine or run this command `source /.bashrc`.

2. **GDBMACROS**
   Edit $PINTOSHOME/src/utils/pintos-gdb to change the definition of GDBMACROS to point to where you installed gdb-macros. Test the installation by running pintos-gdb without any arguments. If it does not complain about missing gdb-macros, it is installed correctly.
   Change this line GDBMACROS=/usr/class/cs140/pintos/pintos/src/misc/gdb-macros to GDBMACROS=$PINTOSHOME/src/misc/gdb-macros.

   Example: `GDBMACROS=/home/ubuntu/pintos/src/misc/gdb-macros`

   Note: do not forget to use the absolute path in place of $PINTOSHOME.

3. Replace $PINTOSHOME/src/devices/shutdown.c with the provided file. (Thanks to Moustafa Abdelaziz for pointing this out. Reference: `https://github.com/jinmel/pintos_mac/blob/master/devices/shutdown.c`.

4. **Compile the Utilities**
   Compile the remaining Pintos utilities by typing make in src/utils. Steps are as follows:

   - $ cd $PINTOSHOME/src/utils
   - $ make

   Note: If you get an error saying "Undefined reference to 'floor'", do the following changes: Edit "Makefile" in the current directory and replace "LDFLAGS = -lm" with "LDLIBS = -lm" and redo the previous step (issue make command in the utils directory).

5. **Set Qemu as Your Simulator**
   If you would like to use the qemu, then edit $PINTOSHOME/src/threads/Make.vars and change the line SIMULATOR = –bochs to SIMULATOR = –qemu.

6. **Compile Pintos Kernel**
   Steps are as follows:

   - $ cd $PINTOSHOME/src/threads/
   - $ make

   A new "build" directory will be created as a sub-directory of: $PINTOSHOME/src/threads/.

7. **Edit util Files**

   - If you would like to make the qemu as your default simulator, then edit $PINTOSHOME/src/utils/pintos by replacing $sim = "bochs" if !defined $sim; in line number 103 with $sim = "qemu" if !defined $sim;.
   - Edit $PINTOSHOME/src/utils/pintos by replacing kernel.bin in line number 257 with the absolute path pointing to it: $PINTOSHOME/src/threads/build/kernel.bin.
   - Edit $PINTOSHOME/src/utils/Pintos.pm by replacing loader.bin in line number 362 with the absolute path pointing to it: $PINTOSHOME/src/threads/build/loader.bin.

## Running Pintos

Pintos should now be ready for use. You can test running pintos as follows:
> pintos run alarm-multiple

## Useful Links

Debugging versus Testing
Grading
Acknowledgements
Trivia

## Notes for Mac Users

- You better use the modified pintos version available at this link
- For installing qemu, you can install brew and use it as follows: `brew install qemu`.