Installing Pintos OS

CS2042 - Operating Systems

August 2021

1 Getting Started

This is a prerequisite for the OS module where you will be introduced to a teaching operating system framework for the 80x86 architecture. It supports basic functionalities such as virtual memory, kernel threads, user programs, and file systems, which you will learn about throughout the semester. The implementation of these functions in Pintos is simple compared to a real Operating System like Linux, yet adequate for you to understand the concepts behind them. The final system you are going to build can be run on a regular IBM-compatible PC (well theoretically), but since you don't have access to such a device we will be using an x86 emulator known as QEMU to host the OS.

1.1 System Requirements

To get started with the project, you will need to have access to a machine that can build Pintos with the appropriate environment setup. Since you will be using your own PC/Laptop to work on this we suggest that you install an UNIX/LINUX based operating system such as Ubuntu if you do not have access to one. We have used 64-bit Ubuntu 18.04, so we advice you to use the same distribution. Although, Ubuntu 16.04 should also work without an issue.

You can also use a windows machine to work on this project, but you will need to have access to a virtualization tool such as VMware or Virtual Box. Although, since you will be spending a considerable amount of time examining source code and editing them, unless you have a sufficiently powerful PC the learning experience might be strenuous due to the performance of the VM. Hence, we highly recommend using a UNIX/LINUX based OS.

2 Installation Guide

2.1 Obtaining Pintos

Download the source code of the Pintos distribution using the following command :

```
$ git clone https://github.com/jhu-cs318/pintos.git
```

This will create a pintos directory where you ran the command and download all the necessary files into it. You can then switch inside the newly created directory and examine the file structure. If you are interested you can open some files and check out the source code too. Don't panic if everything looks daunting at the beginning, with time and patience you will be able to understand them.

2.2 Compiler Toolchain

Before building Pintos you need to have two sets of tools installed in your machine.

- 80x86 cross-compiler toolchain for 32-bit architecture including a C compiler, assembler, linker and a debugger
- x86 emulator such as QEMU

The compiler toolchain is a collection of tools that turns source code into binary executables for a target architecture. Pintos is written in C and x86 assembly, and runs on 32-bit x86 machines. So we will need the appropriate C compiler (`gcc`), assembler (`as`), linker (`ld`) and debugger (`gdb`).

If you are using a Linux machine, it is likely equipped with the compiler toolchain already, but it should support 32-bit x86 architecture. By running `objdump -i | grep elf32-i386` in a terminal you can verify this. If the command gives an output, it means the default toolchain of your system supports our requirements. If so you can jump to section 3, otherwise you will need to build the toolchain from source.

2.3 Building Toolchain from Source

When you are building the toolchain from source, make sure to add `i386-elf-` prefix to the build target in order to distinguish the new toolchain from your system's default toolchain.
e.g. `i386-elf-gcc` , `i386-elf-as`

2.3.1 Prerequisites

- Standard build tools including `make` , `gcc` , etc
- For building `GBD` , you will need ncurses and textinfo libraries.
- You can install the above libraries for Ubuntu with,

```
$ sudo apt-get install build-essential automake git
$ sudo apt-get install libncurses5-dev texinfo
```

2.3.2 Building the Toolchain

A script is provided to you in `pintos/src/misc/toolchain-build.sh` that automates the building instructions for the relevant toolchain. You only have to run the script and modify your PATH settings once the script finishes running. The script is tested to run on recent releases of Ubuntu.

```
$ SWD=/path/to/setup
$ mkdir -p $SWD
$ cd /path/to/pintos/src
$ misc/toolchain-build.sh $SWD
```

Replace `$SWD` with a real path e.g. `/home/cs2042/Desktop/toolchain` to store the toolchain source and build. After successfully running the above command, add the toolchain path to your environment variable settings by editing the `.bashrc` file in your home directory. Save the file and restart the terminal for the PATH variable change to take place.

```
$ export PATH=$SWD/x86_64/bin:$PATH
```

2.3.3 Verify Installation

After following the above steps you can check whether the installation was successful and you have the correct version 6.2.0 by running the following commands.

```
$ which i386-elf-gcc
$ i386-elf-gcc --version
```

If you get a command not found error, check if you have added export PATH=$SWD/x86_64/bin:$ PATH to your .bashrc and restart the terminal.

3 Emulator

In addition to the toolchain, you will also need an x86 emulator to run Pintos. We will be using the modern QEMU emulator for this purpose. Bochs is another emulator that you can use which has a higher accuracy with full emulation compared to QEMU, but it is slower. Hence we will stick to QEMU, but feel free to figure out how to run Pintos on Bochs. To install QEMU,

- Ubuntu ealier than 18.10: sudo apt-get install qemu libvirt-bin
- Ubuntu 18.10 or after: sudo apt-get install qemu libvirt-daemon-system libvirt-clients

4 Pintos Utility Tools

The Pintos source distribution comes with a few handy scripts that you will be using frequently. They are located within src/utils/ . The most important one is the pintos Perl script, which you will be using to start and run tests in pintos. Make sure to add it to the PATH environment variable! The src/misc/gdb-macros provides a number of GDB macros that you will find useful for debugging Pintos. The pintos-gdb is a wrapper around the i386-elf-gdb that reads this macro file at start. It assumes the macro file resides in ../misc .

The example commands to do the above setup for the Pintos utilities are:
(replace /path/to/swd/x86_64 with the actual directory path)

```
$ cd pintos/src/utils && make
$ cp backtrace pintos Pintos.pm pintos-gdb pintos-set-cmdline \
  pintos-mkdisk setitimer-helper squish-pty squish-unix \
  /path/to/swd/x86_64/bin
$ mkdir /path/to/swd/x86_64/misc
$ cp ../misc/gdb-macros /path/to/swd/x86_64/misc
```

5 Other Requirements

- Required: Perl Version 5.8.0 or later
- Recommended:
  - cgdb (**strongly recommended**)

    * Once you installed cgdb, the `pintos-gdb` script would use cgdb to invoke GDB.
  - ctags
  - cscope
  - NERDTree
  - YouCompleteMe
- Optional:
  - GUI editors (e.g. VS Code or CLion)

6 Running Pintos OS

After you have built the toolchain and installed the emulator, you can run the following commands to test Pintos.

```
$ cd pintos/src/threads
$ make
$ cd build
$ pintos --
```

If the installation is successful, a QEMU window will pop up and the following message will be displayed signaling a successful booting of Pintos. If you see `Boot complete` at the end, you are good to go.

```
PiLo hda1
Loading...........
Kernel command line:
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  838,041,600 loops/s.
Boot complete.
```