Installation Guide of PintOS

Prof: Jorge Gonzalez
jgonzalez@utec.edu.pe
TA: Martin Carrasco
martin.carrasco@utec.edu.pe

April 1, 2020

1 Making a docker container for PintOS

1. Create the Dockerfile *(Use the template Dockerfile included to guide you).*
   - Pull the docker image from Docker Hub (e.g.: notice that in our template we pull the image of Ubuntu 14:04).
   - Install dependencies (e.g.: qemu, build-essentials, linux-headers-generic).
   - Set enviromental variables (e.g: `export $PINTOS_HOME=pint-os`).
2. Create a volume to have data be persistant e.g. *sudo docker volume create my_volume PATH* (see additional info in `item 5`).
3. Build the container e.g. `sudo docker build -t pintos .` ((dot) is part of the command).
4. Run the container e.g. `sudo docker run -it --volume my_volume:pint-os --name pint-sim pintos`. For more details about using volumes, please refer to Docker Documentation.

2 Seting up and compiling PintOS

1. Make sure all the dependencies were installed correctly
2. Compile the following submodules
   - `userprog`
   - `vm`
   - `filesys`
3. Compile **/pint-os/utils** (**NOTE:** compile by using `make` on the folder).
4. Only if there is an error related to `floor()` funcion, edit **/pint-os/utils/Makefile** to replace `CFLAGS= -lm` to `LDLIBS = -lm` then compile.
5. Edit **/pint-os/thread/Make.vars** and change `SIMULATOR=` to `SIMULATOR= --qemu` then compile in **/pint-os/threads**.
6. Change **/pint-os/utils/pintos** `$sim=bochos` to `$sim=qemu` in `line 103`.
7. Change **/pint-os/utils/pintos**, check `line 257`, `$name = find_file('kernel.bin')` to point to **/pint-os/threads/build/kernel.bin**
8. Edit `line 362` of **/pint-os/utils/Pintos.pm**, `$name = find_file('loader.bin')` and point it to **/pint-os/threads/build/loader.bin**
9. Go to folder **/pint-os/utils/** and test your program by executing: `./pintos run alarm-multiple`. The expected output is:

```
root@a9ecd025bd0e:/pint-os# pintos run alarm-multiple
qemu-system-i386 -device isa-debug-exit -hda /tmp/D86qNeoZCC.dsk -m 4 -n
PiLo hda1
Loading..........
Kernel command line: run alarm-multiple
Pintos booting with 4,088 kB RAM...
382 pages available in kernel pool.
382 pages available in user pool.
Calibrating timer...  258,457,600 loops/s.
Boot complete.
Executing 'alarm-multiple':
```

```
(alarm-multiple) begin
(alarm-multiple) Creating 5 threads to sleep 7 times each.
(alarm-multiple) Thread 0 sleeps 10 ticks each time,
(alarm-multiple) thread 1 sleeps 20 ticks each time, and so on.
(alarm-multiple) If successful, product of iteration count and
(alarm-multiple) sleep duration will appear in nondescending order.
(alarm-multiple) thread 0: duration=10, iteration=1, product=10
(alarm-multiple) thread 0: duration=10, iteration=2, product=20
(alarm-multiple) thread 1: duration=20, iteration=1, product=20
(alarm-multiple) thread 0: duration=10, iteration=3, product=30
(alarm-multiple) thread 2: duration=30, iteration=1, product=30
(alarm-multiple) thread 3: duration=40, iteration=1, product=40
(alarm-multiple) thread 0: duration=10, iteration=4, product=40
(alarm-multiple) thread 1: duration=20, iteration=2, product=40
```

10. Optional: Add to the PATH the pintos binary folder using `export PATH=/pint-os/utils:$PATH`.

11. See PintOS Documentation for more information. Notice that we are using Qemu.