Operating Systems Lab
Dept of CSE, LNMIIT, Jaipur
27 Jan 2021
Lab Session 02

Note:

1. You have already setup a VirtualBox and tested an Ubuntu system. The instructors have created a platform for many of your lab lessons using an educational OS called PintOS.
2. Please download the VirtualBox plugging computer from the google classroom for this subject. Import it into your VirtualBox.
3. A very detailed and useful description of PintOS html is available here: https://web.stanford.edu/class/cs140/projects/pintos/pintos.html and as pdf in our classroom site.
4. You can find many more sources and copies of the documents on the Internet, for example: https://en.wikipedia.org/wiki/Pinto or https://cs162.org/resources/ or http://www.imperial.ac.uk/computing/current-students/course-admin/noticeboards/second-year/labs/os-lab-project/ Suffice to say that many good universities use PintOS to teach operating systems.
5. Last but not the least. There is this warning, PintOS is small but challenging. And you need to learn it by discussions among your groups and actually trying things. Instructors will try to help but the real learning will require working on the programs and support software tools.

Goals for this Lab Session

1. Get PintOS running on students' computers
2. This lab is primarily to familiarise students with the PintOS software layout and test regime.
3. Make students aware of the make files and let them learn about its purpose as self-help study. Some not too difficult tutorials are available at: https://makefiletutorial.com/ and https://www.tutorialspoint.com/makefile/index.htm
4. Write C programs using pthreads library https://computing.llnl.gov/tutorials/pthreads/ A tutorial link to a CMU site is https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html

Installing and Testing PintOS Platform

1. Download and unzip from google classroom: PintoOS TestBed Virtual Computer on your host (physical) computer. Import it into Oracle VirtualBox to get an Ubuntu 16 OS running.
2. This computer has PintOS source code, Bochs and Qemu already setup. PintOS is located under folder os: /home/pintos/os.
3. Adjust your virtual screen size to meet your computer specifications (right click and then Change Desktop Background – under tab all settings >> screen display).

4. Right click and bring up a terminal by selecting Open Terminal. Run command: cd /home/pintos/os/PintOS/src/threads
5. Run commands (one by one):

   ```
   make clean

   make

   cd build

   make check
   ```

The final few lines that a successful installation of the platform will return are (after some me) will include:

```
20 of 27 tests failed.
```

There are two lines after this in the output.

Threads, Userprog, …

PintOS code available from Stanford repository (and on your downloaded virtual computer) is a partially implemented operating system – a large C + assembly language program with a main() function. The program can be boot-loaded to an Intel X86 computer.

This is exactly what you did under section Installing and Testing PintOS Platform. The computer receiving the OS was a simulated computer!

If the loaded OS (PintOS) were a full function OS, we could have run "Hello world" program on it. Hello world program is a user program with its own function main(). Each user program runs as a process. The program running as a process requests and receives services from the operating system through system calls and is scheduled by the OS scheduler.

Since PintOS does not have some facilities yet, PintOS has used a few tricks to test this and note the failures.

To get a different (later) perspective we may attempt to jump to use program test stage. To do so, do the following changes to the support scripts:

1. Edit line 259 in Perl script /home/pintos/os/bin/pintos: Replace name "threads" in path to "userprog". This is where the compiler will place the loadable kernel.
2. Edit line 362 in Perl script /home/pintos/os/bin/Pintos.pm: Replace name "threads" in path to "userprog".
3. In a terminal window, first cd to threads directory and run command: make clean. (Run command: cd /home/pintos/os/PintOS/src/threads)
4. Next, run commands:

   ```
   cd ../userprog

   run command: make

   cd build

   make check
   ```

And now wait for a long time to let various test programs run and fail!

UNDO THE TWO EDITS. We are not planning to work on USERPROG chapter of PintOS this year.

THREAD TESTS

Let us get a little more familiar with the threads section. Feel free to work in small groups if your learning style requires that arrangement.

When PintOS is loaded, the boot-loader hands the control to the main function (in the C program that implements the OS). PintOS document explains use of tools cscope and ctags to guide you to locate function main(). Try tracing PintOS code to figure out how the execution progresses.

Since given PintOS is not yet a functioning OS, the test cases that we could run through command "make check" where organised by C function in file /home/pintos/os/PintOS/src/tests/threads/tests.c The function calls various other test functions which are in `.c` files in the same folder as tests.c.

The function in file tests.c is linked into the operating system code together with the functions called from this function. A real OS, however, does not link user programs into the OS code – user programs run at an arm's length to the operating systems as independent processes. Since these test functions are linked into the OS at this stage, they have access to all internal code of the OS. User programs are not so fortunate.

The output of various tests is readable (after the test run) in `.output` files created in folder /home/pintos/os/PintOS/src/threads/build/tests/threads/ There are .ck files that contain the expected outputs. Expected output of a test is compared against the actual output for the tests to *pass* or *fail* the test. So, you can easily find the expected output for many tests.

Ignore all tests tagged mlfbq – Multi-Level Feed-Back Queue. We will not work on these tests this year. You will, however, be taught this topic in the class!

PintOS implementation aims to provide many of the functions that you have discussed in the class (pthreads library). Take the codes for some of the test functions included in PintOS tests/threads directory and construct programs using pthreads library to produce the correct outputs as described in the test-checking .ck files.

For better understanding choose a test case that failed during PintOS make check run. You may find some of the function very intriguing or difficult. Just choose some that are not too confusing to you.

It must be obvious to you that you will be executing your solution program as an Ubuntu program. It would require you to define a function main() in your solution code. To avoid corrupting the PintOS tests, you must create a separate copy of the tests folder away from os folder to work on your solution.

Your group may also test your skills with make file for this exercise. Write each function in a file of its own. With only a small number of files in your solution, writing a make file will not be a difficult task.