PINTOS- PROJECT 1

STEP BY STEP INSTALLATION GUIDE

COMPILED BY

S.DHANYA ABHIRAMI

TERMINOLOGY

- QEMU (short for **Quick Emulator**) is a free and open-source hosted hypervisor that performs hardware virtualization
- A **hypervisor** or virtual machine monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines. A computer on which a **hypervisor** runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine.
- The **GNU Debugger** (**GDB**) is a portable debugger that runs on many Unix-like systems and works for many programming languages, including C, C++ and Java

PRE-REQUISITES

- Linux OS (Here Ubuntu 16.4 is used)
- qemu (Emulator)
- GNU make
- GNU binutils
- perl
- gdb

Code

sudo apt-get install qemu make binutils perl gdb

Obtaining the Source Code

- Download the tar.gz file containing the PintOS source
- http://www.stanford.edu/class/cs140/projects/pintos/pintos.tar.gz(mirror)
- Extract the contents to the $HOME directory.

SETTING THE PATH VARIABLES

To add /src/utils to the $PATH

- Open terminal
- Add the line
  PATH=”$PATH:$HOME/pintos/src/utils” to
  .profile or .bashrc
  $ echo PATH="$PATH:$HOME/pintos/src/utils"
  >> ~/.bashrc
- Reload your shell
  $ source ~/.bashrc
- To verify that the environment has been set
  correctly, run the following command
  $ pintos

```
skanda@skanda-Lenovo-G50-80:~$ echo PATH="$PATH:$HOME/pintos/src/utils" >>~/.profile
skanda@skanda-Lenovo-G50-80:~$ source ~/.profile
skanda@skanda-Lenovo-G50-80:~$ pintos
pintos, a utility for running Pintos in a simulator
Usage: pintos [OPTION...] -- [ARGUMENT...]
where each OPTION is one of the following options
  and each ARGUMENT is passed to Pintos kernel verbatim.
Simulator selection:
  --bochs                (default) Use Bochs as simulator
  --qemu                 Use QEMU as simulator
  --player               Use VMware Player as simulator
Debugger selection:
  --no-debug             (default) No debugger
  --monitor              Debug with simulator's monitor
  --gdb                  Debug with gdb
Display options: (default is both VGA and serial)
  -v, --no-vga           No VGA display or keyboard
  -s, --no-serial        No serial input or output
  -t, --terminal         Display VGA in terminal (Bochs only)
Timing options: (Bochs only)
  -j SEED                Randomize timer interrupts
  -r, --realtime         Use realistic, not reproducible, timings
Testing options:
  -T, --timeout=N        Kill Pintos after N seconds CPU time or N*load_avg
                         seconds wall-clock time (whichever comes first)
  -k, --kill-on-failure  Kill Pintos a few seconds after a kernel or user
                         panic, test failure, or triple fault
Configuration options:
  -m, --mem=N            Give Pintos N MB physical RAM (default: 4)
File system commands:
  -p, --put-file=HOSTFN  Copy HOSTFN into VM, by default under same name
  -g, --get-file=GUESTFN Copy GUESTFN out of VM, by default under same name
  -a, --as=FILENAME      Specifies guest (for -p) or host (for -g) file name
Partition options: (where PARTITION is one of: kernel filesys scratch swap)
  --PARTITION=FILE       Use a copy of FILE for the given PARTITION
  --PARTITION-size=SIZE  Create an empty PARTITION of the given SIZE in MB
  --PARTITION-from=DISK  Use of a copy of the given PARTITION in DISK
  (There is no --kernel-size, --scratch, or --scratch-from option.)
Disk configuration options:
  --make-disk=DISK       Name the new DISK and don't delete it after the run
  --disk=DISK            Also use existing DISK (may be used multiple times)
Advanced disk configuration options:
  --loader=FILE          Use FILE as bootstrap loader (default: loader.bin)
```

COMPILING THE SOURCE

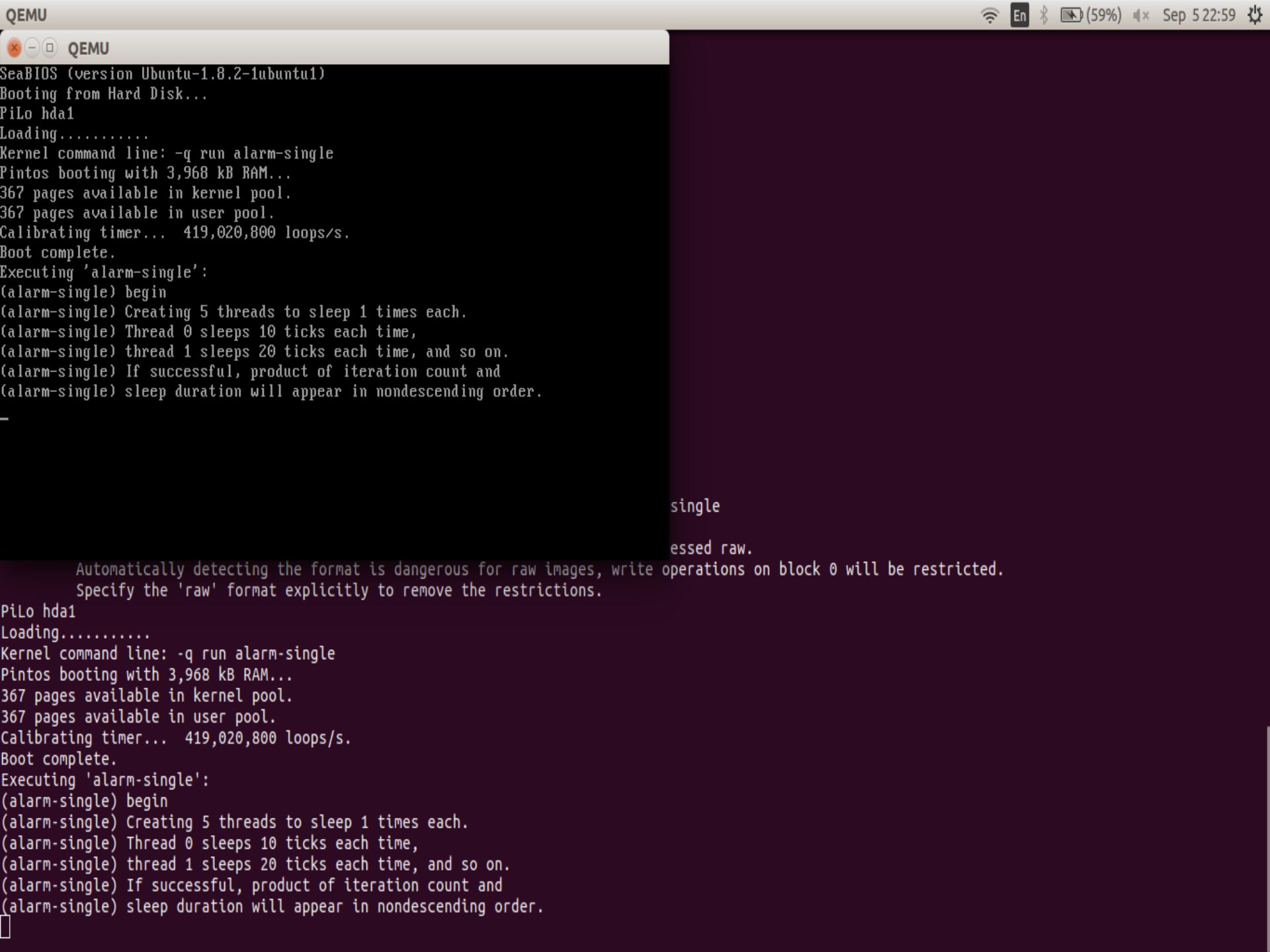
- Open the pintos perl script in src/utils/ and replace line 623 (my (@cmd) = ('qemu');) with my (@cmd) = ('qemu-system-i386');
- open the file src/devices/shutdown.c
  Now, Insert the line outw(0x604, 0x0 | 0x2000); after printf ("Powering off...\n"); serial_flush ();
- Head over to src/threads directory
  $ cd pintos/src/threads
- Set the simulator to qemu in Make.vars file
  SIMULATOR = --qemu
- Compile the source with the make command
  $ make

```
File system commands:
  -p, --put-file=HOSTFN    Copy HOSTFN into VM, by default under same name
  -g, --get-file=GUESTFN   Copy GUESTFN out of VM, by default under same name
  -a, --as=FILENAME        Specifies guest (for -p) or host (for -g) file name
Partition options: (where PARTITION is one of: kernel filesys scratch swap)
  --PARTITION=FILE         Use a copy of FILE for the given PARTITION
  --PARTITION-size=SIZE    Create an empty PARTITION of the given SIZE in MB
  --PARTITION-from=DISK    Use of a copy of the given PARTITION in DISK
  (There is no --kernel-size, --scratch, or --scratch-from option.)
Disk configuration options:
  --make-disk=DISK         Name the new DISK and don't delete it after the run
  --disk=DISK              Also use existing DISK (may be used multiple times)
Advanced disk configuration options:
  --loader=FILE            Use FILE as bootstrap loader (default: loader.bin)
  --geometry=H,S           Use H head, S sector geometry (default: 16,63)
  --geometry=zip           Use 64 head, 32 sector geometry for USB-ZIP boot
                           (see http://syslinux.zytor.com/usbkey.php)
  --align=bochs            Pad out disk to cylinder to support Bochs (default)
  --align=full             Align partition boundaries to cylinder boundary to
                           let fdisk guess correct geometry and quiet warnings
  --align=none             Don't align partitions at all, to save space
Other options:
  -h, --help               Display this help message.
skanda@skanda-Lenovo-G50-80:~$ cd pintos/src/threads
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$ SIMULATOR=--qemu
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$ make
cd build && make all
make[1]: Entering directory '/home/skanda/pintos/src/threads/build'
gcc -m32 -c ../../devices/shutdown.c -o devices/shutdown.o -g -msoft-float -O -fno-stack-protector -nostdinc -I../.. -I../../lib -I../../lib/kernel -W
all -W -Wstrict-prototypes -Wmissing-prototypes -Wsystem-headers  -MMD -MF devices/shutdown.d
ld -melf_i386 -T threads/kernel.lds.s -o kernel.o threads/start.o threads/init.o threads/thread.o threads/switch.o threads/interrupt.o threads/intr-st
ubs.o threads/synch.o threads/palloc.o threads/malloc.o devices/pit.o devices/timer.o devices/kbd.o devices/vga.o devices/serial.o devices/block.o dev
ices/partition.o devices/ide.o devices/input.o devices/intq.o devices/rtc.o devices/shutdown.o devices/speaker.o lib/debug.o lib/random.o lib/stdio.o
lib/stdlib.o lib/string.o lib/arithmetic.o lib/ustar.o lib/kernel/debug.o lib/kernel/list.o lib/kernel/bitmap.o lib/kernel/hash.o lib/kernel/console.o
 tests/threads/tests.o tests/threads/alarm-wait.o tests/threads/alarm-simultaneous.o tests/threads/alarm-priority.o tests/threads/alarm-zero.o tests/t
hreads/alarm-negative.o tests/threads/priority-change.o tests/threads/priority-donate-one.o tests/threads/priority-donate-multiple.o tests/threads/pri
ority-donate-multiple2.o tests/threads/priority-donate-nest.o tests/threads/priority-donate-sema.o tests/threads/priority-donate-lower.o tests/threads
/priority-fifo.o tests/threads/priority-preempt.o tests/threads/priority-sema.o tests/threads/priority-condvar.o tests/threads/priority-donate-chain.o
 tests/threads/mlfqs-load-1.o tests/threads/mlfqs-load-60.o tests/threads/mlfqs-load-avg.o tests/threads/mlfqs-recent-1.o tests/threads/mlfqs-fair.o t
ests/threads/mlfqs-block.o
objcopy -R .note -R .comment -S kernel.o kernel.bin
make[1]: Leaving directory '/home/skanda/pintos/src/threads/build'
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$
```

TEST PROGRAM 1

- Run a program called ‘alarm-single’ in PintOS with QEMU
  $ pintos --qemu -- -q run alarm-single

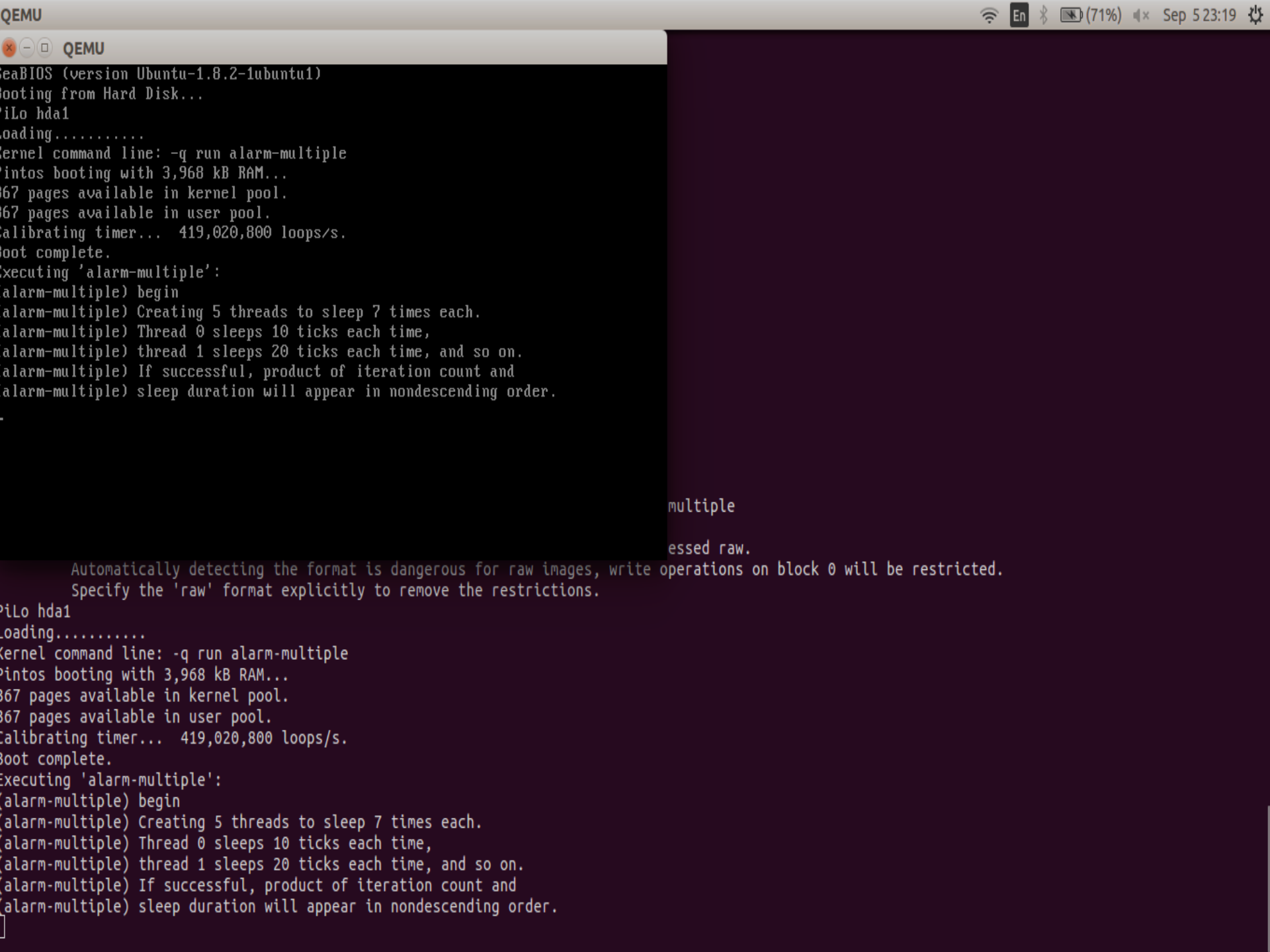
```
ices/partition.o devices/ide.o devices/input.o devices/intq.o devices/rtc.o devices/shutdown.o devices/speaker.o lib/debug.o lib/random.o lib/stdio.o
lib/stdlib.o lib/string.o lib/arithmetic.o lib/ustar.o lib/kernel/debug.o lib/kernel/list.o lib/kernel/bitmap.o lib/kernel/hash.o lib/kernel/console.o
 tests/threads/tests.o tests/threads/alarm-wait.o tests/threads/alarm-simultaneous.o tests/threads/alarm-priority.o tests/threads/alarm-zero.o tests/t
hreads/alarm-negative.o tests/threads/priority-change.o tests/threads/priority-donate-one.o tests/threads/priority-donate-multiple.o tests/threads/pri
ority-donate-multiple2.o tests/threads/priority-donate-nest.o tests/threads/priority-donate-sema.o tests/threads/priority-donate-lower.o tests/threads
/priority-fifo.o tests/threads/priority-preempt.o tests/threads/priority-sema.o tests/threads/priority-condvar.o tests/threads/priority-donate-chain.o
 tests/threads/mlfqs-load-1.o tests/threads/mlfqs-load-60.o tests/threads/mlfqs-load-avg.o tests/threads/mlfqs-recent-1.o tests/threads/mlfqs-fair.o t
ests/threads/mlfqs-block.o
objcopy -R .note -R .comment -S kernel.o kernel.bin
make[1]: Leaving directory '/home/skanda/pintos/src/threads/build'
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$ pintos --qemu -- -q run alarm-single
qemu-system-i386 -hda /tmp/OlQImywy1Q.dsk -m 4 -net none -serial stdio
WARNING: Image format was not specified for '/tmp/OlQImywy1Q.dsk' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading..........
Kernel command line: -q run alarm-single
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  104,755,200 loops/s.
Boot complete.
Executing 'alarm-single':
(alarm-single) begin
(alarm-single) Creating 5 threads to sleep 1 times each.
(alarm-single) Thread 0 sleeps 10 ticks each time,
(alarm-single) thread 1 sleeps 20 ticks each time, and so on.
(alarm-single) If successful, product of iteration count and
(alarm-single) sleep duration will appear in nondescending order.
(alarm-single) thread 0: duration=10, iteration=1, product=10
(alarm-single) thread 1: duration=20, iteration=1, product=20
(alarm-single) thread 2: duration=30, iteration=1, product=30
(alarm-single) thread 3: duration=40, iteration=1, product=40
(alarm-single) thread 4: duration=50, iteration=1, product=50
(alarm-single) end
Execution of 'alarm-single' complete.
Timer: 280 ticks
Thread: 0 idle ticks, 280 kernel ticks, 0 user ticks
Console: 986 characters output
Keyboard: 0 keys pressed
Powering off...
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$
```


QEMU
En
(59%)
Sep 5 22:59
QEMU
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)
Booting from Hard Disk...
PiLo hda1
Loading.....
Kernel command line: -q run alarm-single
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 419,020,800 loops/s.
Boot complete.
Executing 'alarm-single':
(alarm-single) begin
(alarm-single) Creating 5 threads to sleep 1 times each.
(alarm-single) Thread 0 sleeps 10 ticks each time,
(alarm-single) thread 1 sleeps 20 ticks each time, and so on.
(alarm-single) If successful, product of iteration count and
(alarm-single) sleep duration will appear in nondescending order.
single
essed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.....
Kernel command line: -q run alarm-single
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 419,020,800 loops/s.
Boot complete.
Executing 'alarm-single':
(alarm-single) begin
(alarm-single) Creating 5 threads to sleep 1 times each.
(alarm-single) Thread 0 sleeps 10 ticks each time,
(alarm-single) thread 1 sleeps 20 ticks each time, and so on.
(alarm-single) If successful, product of iteration count and
(alarm-single) sleep duration will appear in nondescending order.

TEST PROGRAM 2

- Run a program called ‘alarm-single’ in PintOS with QEMU
  $ pintos --qemu -- -q run alarm-multiple

```
skanda@skanda-Lenovo-G50-80:~/pintos/src/threads$ pintos --qemu -- -q run alarm-multiple
qemu-system-i386 -hda /tmp/YU3XM3G_0l.dsk -m 4 -net none -serial stdio
WARNING: Image format was not specified for '/tmp/YU3XM3G_0l.dsk' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading...........
Kernel command line: -q run alarm-multiple
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  209,510,400 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 1: duration=20, iteration=1, product=20
(alarm-multiple) thread 0: duration=10, iteration=2, 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 1: duration=20, iteration=2, product=40
(alarm-multiple) thread 3: duration=40, iteration=1, product=40
(alarm-multiple) thread 0: duration=10, iteration=4, product=40
(alarm-multiple) thread 4: duration=50, iteration=1, product=50
(alarm-multiple) thread 0: duration=10, iteration=5, product=50
(alarm-multiple) thread 1: duration=20, iteration=3, product=60
(alarm-multiple) thread 2: duration=30, iteration=2, product=60
(alarm-multiple) thread 0: duration=10, iteration=6, product=60
(alarm-multiple) thread 0: duration=10, iteration=7, product=70
(alarm-multiple) thread 1: duration=20, iteration=4, product=80
(alarm-multiple) thread 3: duration=40, iteration=2, product=80
(alarm-multiple) thread 2: duration=30, iteration=3, product=90
(alarm-multiple) thread 4: duration=50, iteration=2, product=100
(alarm-multiple) thread 1: duration=20, iteration=5, product=100
(alarm-multiple) thread 1: duration=20, iteration=6, product=120
(alarm-multiple) thread 2: duration=30, iteration=4, product=120
(alarm-multiple) thread 3: duration=40, iteration=3, product=120
(alarm-multiple) thread 1: duration=20, iteration=7, product=140
```


QEMU
En
(71%)
Sep 5 23:19
QEMU
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)
Booting from Hard Disk...
PiLo hda1
Loading.....
Kernel command line: -q run alarm-multiple
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 419,020,800 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.
multiple
essed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.....
Kernel command line: -q run alarm-multiple
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 419,020,800 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.

Conclusion

- Pintos successfully installed!

REFERENCS

- Ben Pfaff, Anthony Romano, Godmar Back. The Pintos Instructional Operating System Kernel. SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.
- https://web.stanford.edu/class/cs140/projects/pintos/pintos_1.html
- https://web.stanford.edu/class/cs140/projects/pintos/pintos.pdf