

# Project 0: Getting Real

## Preliminaries

Fill in your name and email address.

孙绍聪 2000012977@stu.pku.edu.cn

If you have any preliminary comments on your submission, notes for the TAs, please give them here.

Please cite any offline or online sources you consulted while preparing your submission, other than the Pintos documentation, course text, lecture notes, and course staff.

[IntrList]. R. Brown, Ralf Brown's Interrupt List, 2000.

## Booting Pintos

A1: Put the screenshot of Pintos running example here.

```

C:\WINDOWS\system32\cmd.exe - docker run -it --rm --name pintos --mount type=bind,source=D:\OperatingSystem\pintos,target=/ho...
root@974d29489e34:~/pintos/src/threads/build# pintos --
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/3kGi8hvdVV.dsk -m 4 -net none -no
graphic -monitor null
Pintos 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... 104,755,200 loops/s.
Boot complete.

C:\WINDOWS\system32\cmd.exe - docker run -it --rm --name pintos --mount type=bind,source=D:\OperatingSystem\pintos,target=/ho...
Boot complete.
qemu-system-i386: terminating on signal 2

root@7f059a9c6dbf:~/pintos/src/threads/build# exit
exit
C:\Users\Mr.Sun>docker run -it --rm --name pintos --mount type=bind,source=D:\OperatingSystem\pintos,target=/home/PKUOS/
pintos pkuflyingpig/pintos bash
root@974d29489e34:~# cd pintos/src/threads
root@974d29489e34:~/pintos/src/threads# cd build
root@974d29489e34:~/pintos/src/threads/build# pintos --bochs --
squish-pty bochs -q

=====
Bochs x86 Emulator 2.6.2
Built from SVN snapshot on May 26, 2013
Compiled on Nov 18 2021 at 12:44:44
=====
00000000000i[ ] reading configuration from bochsrc.txt
00000000000e[ ] bochsrc.txt:8: 'user_shortcut' will be replaced by new 'keyboard' option.
00000000000i[ ] installing nogui module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
Pintos hda1
Loading.....
Kernel command line:
Pintos booting with 4,096 kB RAM..
383 pages available in kernel pool.
383 pages available in user pool.
Calibrating timer... 102,400 loops/s.
Boot complete.

```

## Debugging

### QUESTIONS: BIOS

B1: What is the first instruction that gets executed?

ljmp \$0xf000,\$0xe05b

B2: At which physical address is this instruction located?

0xffff0

## QUESTIONS: BOOTLOADER

B3: How does the bootloader read disk sectors? In particular, what BIOS interrupt is used?

By using BIOS interrupt, serial(0x14).

B4: How does the bootloader decide whether it successfully finds the Pintos kernel?

It reads the partition table on each system hard disk and scan for a partition of type 0x20. This type means the Pintos kernel has been found.

B5: What happens when the bootloader could not find the Pintos kernel?

It will print the string "\rNot found\r" and notify BIOS that boot failed(0x18).

B6: At what point and how exactly does the bootloader transfer control to the Pintos kernel?

At line 168 in file loader.S, the bootloader transfers control to the Pintos kernel by an indirectly jump instruction "ljmp \*start", after the kernel being loaded. The loader has to store the 32-bit address and then jump indirectly through that location since the 80x86 doesn't have an instruction to jump to an absolute segment:offset kept in registers.

```
mov $0x2000, %ax
mov %ax, %es
mov %es:0x18, %dx
mov %dx, start
movw $0x2000, start + 2
ljmp *start <--here
```

## QUESTIONS: KERNEL

B7: At the entry of `pintos_init()`, what is the value of expression `init_page_dir[pd_no(ptov(0))]` in hexadecimal format?

0x00

B8: When `pallocc_get_page()` is called for the first time,

B8.1 what does the call stack look like?

```
#0 pallocc_get_page (flags=(PAL_ASSERT | PAL_ZERO)) at ../../threads/pallocc.c:113
#1 0xc00204a8 in paging_init () at ../../threads/init.c:218
#2 0xc0020412 in pintos_init () at ../../threads/init.c:146
#3 0xc002013d in start () at ../../threads/start.S:180
```

B8.2 what is the return value in hexadecimal format?

0xc0101000

B8.3 what is the value of expression `init_page_dir[pd_no(ptov(0))]` in hexadecimal format?

0x00

B9: When `palloc_get_page()` is called for the third time,

B9.1 what does the call stack look like?

```
#0 palloc_get_page (flags=PAL_ZERO) at ../../threads/palloc.c:112
#1 0xc0020b7f in thread_create (name=0xc002e9c5 "idle", priority=0,
function=0xc0020fae , aux=0xc000efbc) at ../../threads/thread.c:178
#2 0xc0020a74 in thread_start () at ../../threads/thread.c:111
#3 0xc002042b in pintos_init () at ../../threads/init.c:165
#4 0xc002013d in start () at ../../threads/start.S:180
```

B9.2 what is the return value in hexadecimal format?

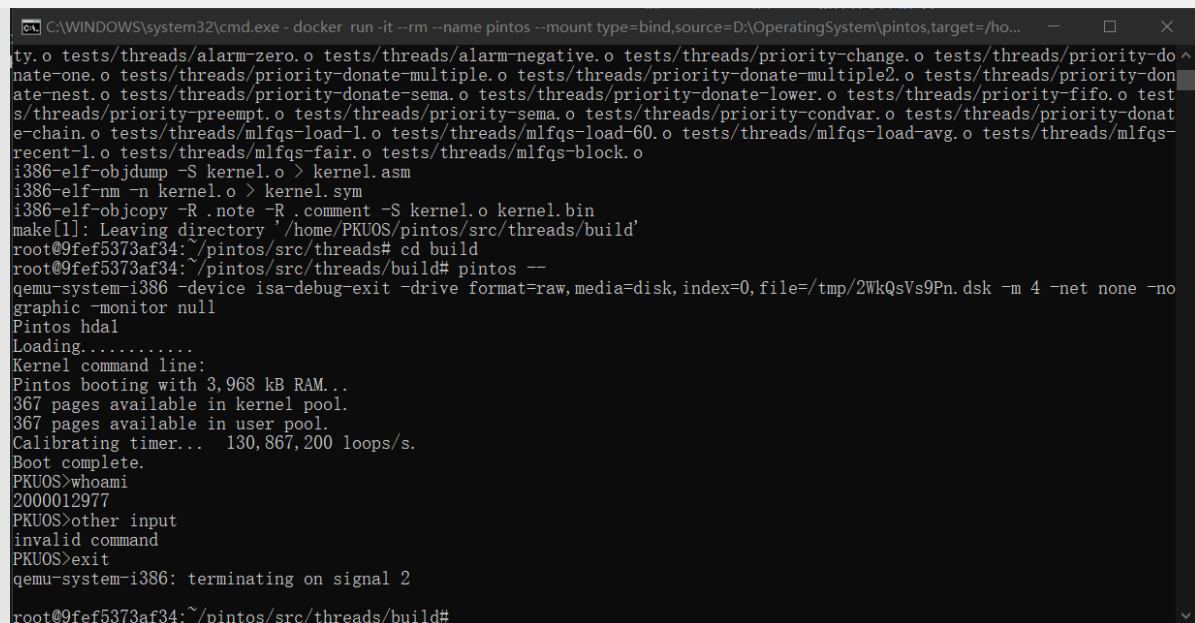
0xc0103000

B9.3 what is the value of expression `init_page_dir[pd_no(ptov(0))]` in hexadecimal format?

0x102027

## Kernel Monitor

C1: Put the screenshot of your kernel monitor running example here. (It should show how your kernel shell respond to `whoami`, `exit`, and `other input`.)



```
C:\WINDOWS\system32\cmd.exe - docker run -it --rm --name pintos --mount type=bind,source=D:\OperatingSystem\pintos,target=/ho...
ty.o tests/threads/alarm-zero.o tests/threads/alarm-negative.o tests/threads/priority-change.o tests/threads/priority-do
nate-one.o tests/threads/priority-donate-multiple.o tests/threads/priority-donate-multiple2.o tests/threads/priority-don
ate-nest.o tests/threads/priority-donate-sema.o tests/threads/priority-donate-lower.o tests/threads/priority-fifo.o test
s/threads/priority-preempt.o tests/threads/priority-sema.o tests/threads/priority-condvar.o tests/threads/priority-donat
e-chain.o tests/threads/mlfqs-load-l.o tests/threads/mlfqs-load-60.o tests/threads/mlfqs-load-avg.o tests/threads/mlfqs-
recent-l.o tests/threads/mlfqs-fair.o tests/threads/mlfqs-block.o
i386-elf-objdump -S kernel.o > kernel.asm
i386-elf-nm -n kernel.o > kernel.sym
i386-elf-objcopy -R .note -R .comment -S kernel.o kernel.bin
make[1]: Leaving directory '/home/PKUOS/pintos/src/threads/build'
root@9fef5373af34:~/pintos/src/threads# cd build
root@9fef5373af34:~/pintos/src/threads/build# pintos --
qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/2WkQsVs9Pn.dsk -m 4 -net none -no
graphic -monitor null
Pintos 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... 130,867,200 loops/s.
Boot complete.
PKUOS>whoami
2000012977
PKUOS>other input
invalid command
PKUOS>exit
qemu-system-i386: terminating on signal 2
root@9fef5373af34:~/pintos/src/threads/build#
```

C2: Explain how you read and write to the console for the kernel monitor.

The kernel monitor reads and displays the command by the function `getline()` declared in `threads/init.c`. It reads the user's input from the keyboard buffer (through `input_getc()`) and displays the characters on the screen (through `printf()`) one by one. This `getline` process will terminate either the "enter" key (`\r`) is detected or user's input reaches the length limitation.