Project 0: Getting Real

Preliminaries

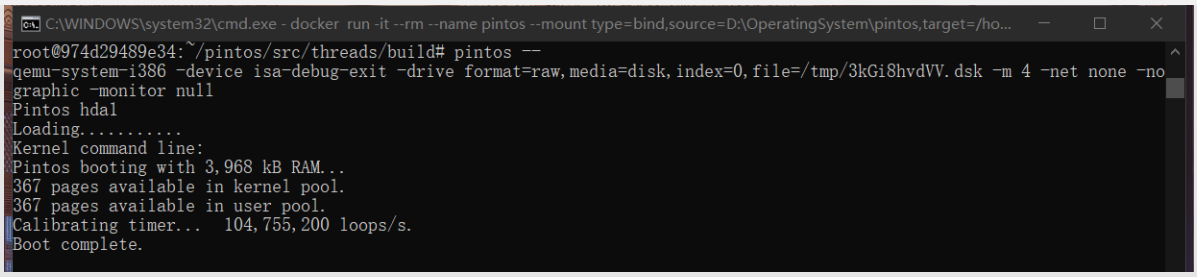
> Fill in your name and email address.
>
> > 孙绍聪 2000012977@stu.pku.edu.cn

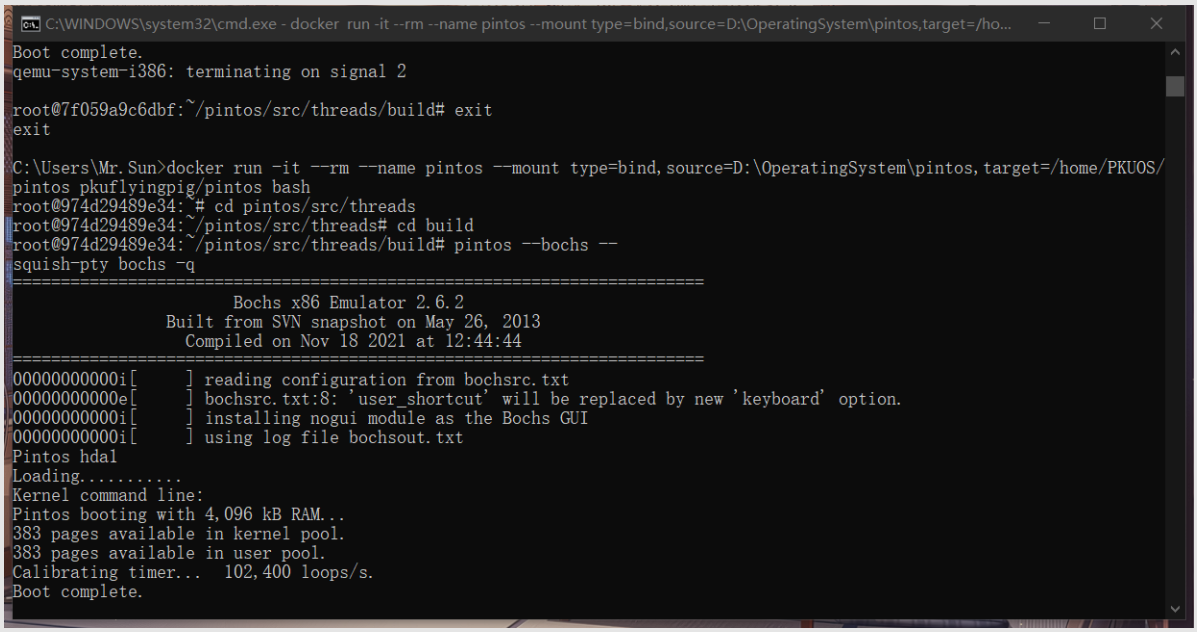
> If you have any preliminary comments on your submission, notes for the TAs, please give them here.

> 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.





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 decides 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 a indirectly jump instruction "ljmp *start", after the kennel 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 `palloc_get_page()` is called for the first time,
>
> > B8.1 what does the call stack look like?
> >
> > > #0 palloc_get_page (flags=(PAL_ASSERT | PAL_ZERO)) at ../../threads/palloc.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`.)
>
> 
>

> 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 diaplays the characters on the screen(through printf()) one by one. This getline process will terminate either the "enter" key('\r') is decated or user's input reaches the length limitation.