Project 0: Getting Real

Preliminaries

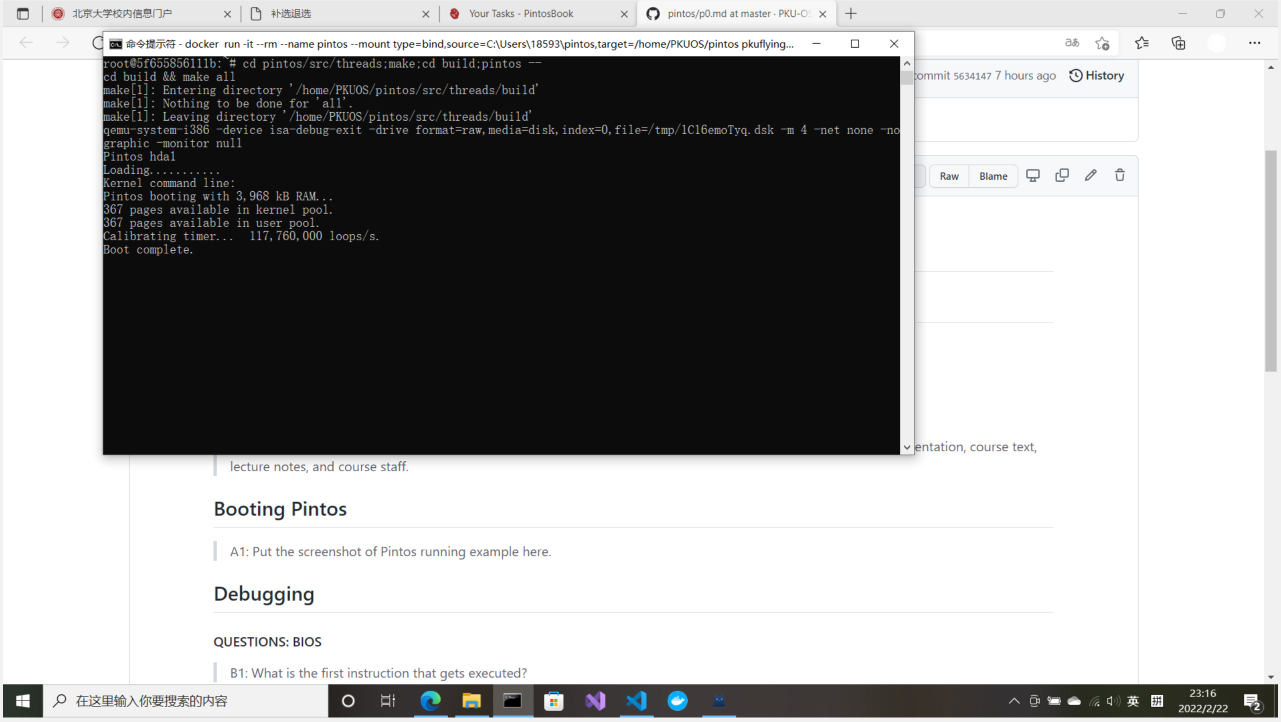
> Fill in your name and email address.

Jin Yang 2000012935@stu.pku.edu.cn

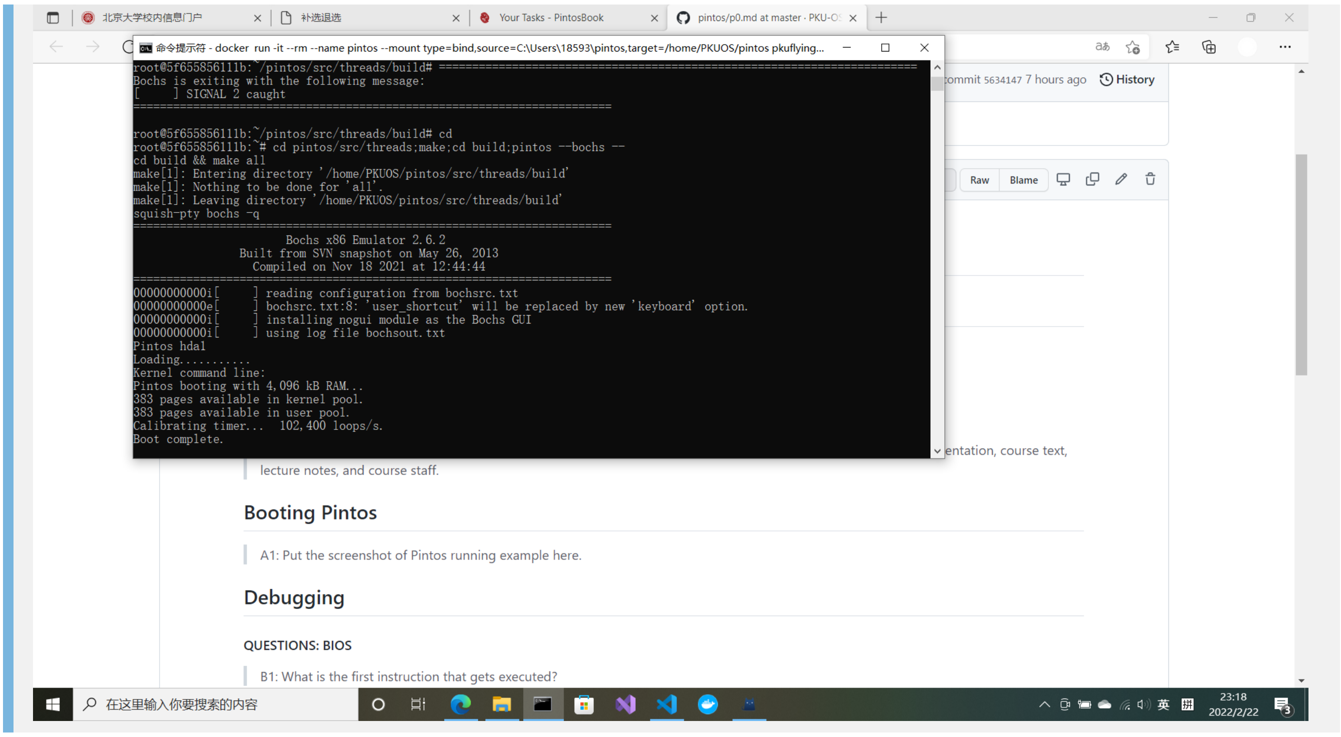
> If you have any preliminary comments on your submission, notes for the TAs, please give them here.

> 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.

Booting Pintos

> A1: Put the screenshot of Pintos running example here.
>
> 
>

北京大学校内信息门户
> 补选退选
> Your Tasks - PintosBook
> pintos/p0.md at master · PKU-O
> commit 5634147 7 hours ago
> History
> Raw
> Blame
> 在这里输入你要搜索的内容
> 23:16
> 2022/2/22
>
> 命令提示符 - docker run -it --rm --name pintos --mount type=bind,source=C:\Users\18593\pintos,target=/home/PKUOS/pintos pkuflying...
>
> ```
> root@5f655856111b:~# cd pintos/src/threads;make;cd build;pintos --
> cd build && make all
> make[1]: Entering directory '/home/PKUOS/pintos/src/threads/build'
> make[1]: Nothing to be done for 'all'.
> make[1]: Leaving directory '/home/PKUOS/pintos/src/threads/build'
> qemu-system-i386 -device isa-debug-exit -drive format=raw,media=disk,index=0,file=/tmp/1C16emoTyq.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...  117,760,000 loops/s.
> Boot complete.
> ```
>
> ...entation, course text, lecture notes, and course staff.
>
> Booting Pintos
>
> A1: Put the screenshot of Pintos running example here.
>
> Debugging
>
> QUESTIONS: BIOS
>
> B1: What is the first instruction that gets executed?


北京大学校内信息门户
补选退选
Your Tasks - PintosBook
pintos/p0.md at master · PKU-OS
命令提示符 - docker run -it --rm --name pintos --mount type=bind,source=C:\Users\18593\pintos,target=/home/PKUOS/pintos pkuflying...
commit 5634147 7 hours ago
History
Raw
Blame
在这里输入你要搜索的内容
23:18
2022/2/22

```
root@5f655856111b:~/pintos/src/threads/build# ========================================================================
Bochs is exiting with the following message:
[      ] SIGNAL 2 caught
========================================================================

root@5f655856111b:~/pintos/src/threads/build# cd
root@5f655856111b:~# cd pintos/src/threads;make;cd build;pintos --bochs --
cd build && make all
make[1]: Entering directory '/home/PKUOS/pintos/src/threads/build'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/PKUOS/pintos/src/threads/build'
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.
```

…entation, course text, lecture notes, and course staff.

Booting Pintos

A1: Put the screenshot of Pintos running example here.

Debugging

QUESTIONS: BIOS

B1: What is the first instruction that gets executed?

Debugging

QUESTIONS: BIOS

> B1: What is the first instruction that gets executed?

```
ljmp   $0xf000,$0xe05b
```

> B2: At which physical address is this instruction located?

```
0xfffffff0
```

QUESTIONS: BOOTLOADER

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

Firstly,call **read_mbr** to find the partitioned hard disk, in the disk,use **check_partition** to find the matched partition, in the partition, call **read_sector** to read the sector;***int $0x13*** is used when reading sector.

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

Check the address **%es:(%si)** and **%es:4(%si)**, if **%es:4(%si)** is **0x20**, the kernel is found and if **%es:(%si)** is **0x80**, it is bootable.

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

It prints "*Not found*" and calls an interrupt ***int $0x18***

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

After reading all the sectors of the kernel, it sets the start address(0x20000) and uses a **ljmp** instruction to transfer control to the kernel

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?

```
0x0
```

> 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 0xc00203aa in paging_init () at ../../threads/init.c:168 #2 0xc002031b in pintos_init () at ../../threads/init.c:100 #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?

```
0x0
```

> 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:113 #1 0xc0020a81 in thread_create (name=0xc002e895 "idle", priority=0, function=0xc0020eb0 , aux=0xc000efbc) at ../../threads/thread.c:178 #2 0xc0020976 in thread_start () at ../../threads/thread.c:111 #3 0xc0020334 in pintos_init () at ../../threads/init.c:119 #4 0xc002013d in start () at ../../threads/start.S:180

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

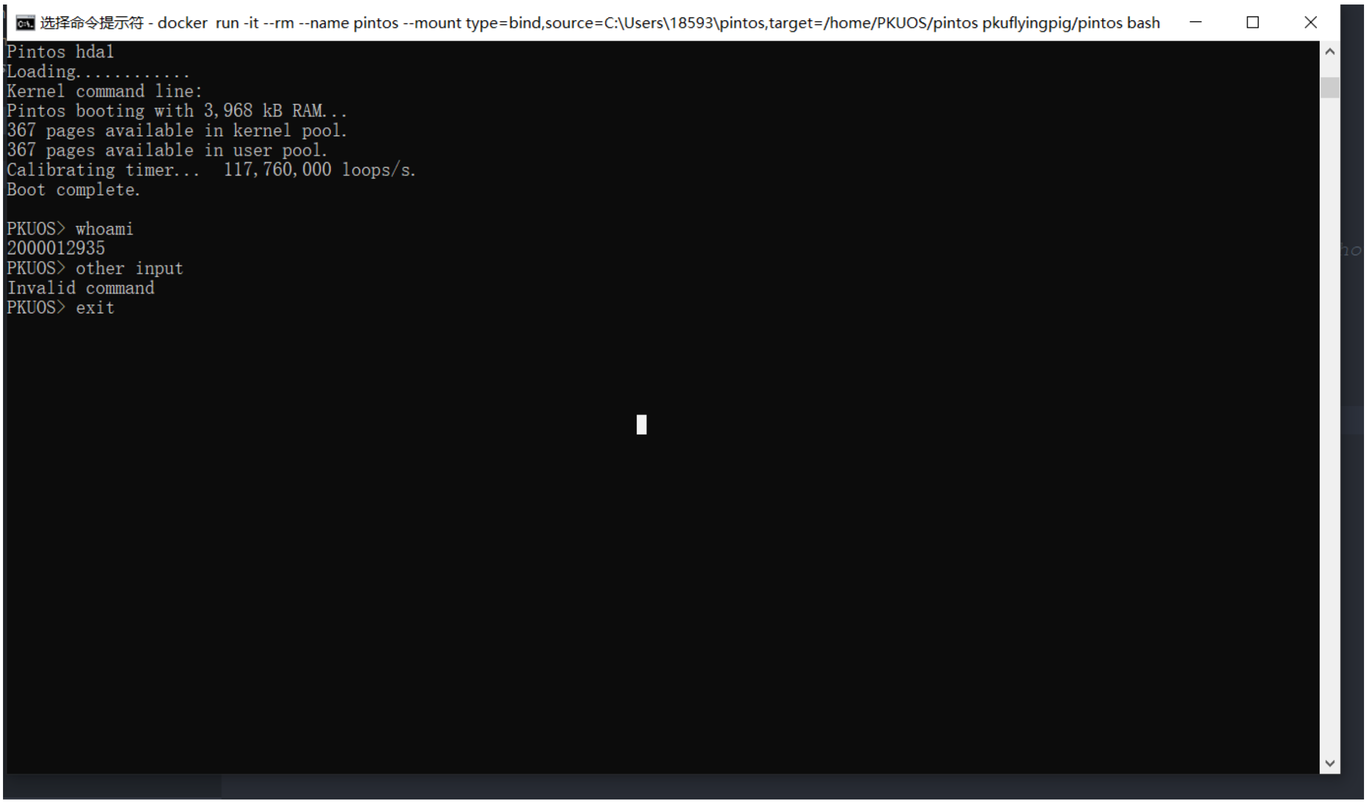
```
0xc0103000
```

> > 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`.)



选择命令提示符 - docker run -it --rm --name pintos --mount type=bind,source=C:\Users\18593\pintos,target=/home/PKUOS/pintos pkuflyingpig/pintos bash

```
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...  117,760,000 loops/s.
Boot complete.

PKUOS> whoami
2000012935
PKUOS> other input
Invalid command
PKUOS> exit
```

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

Use **input_getc()** to read the command, and store the command in a string ***cmd*** at the same time, print the command with **putchar()**.For the valid command, use **printf()** to print strings.