JOS Introduction

Ziqi Zhang（张子祺） Hanwen Lei（雷瀚文）

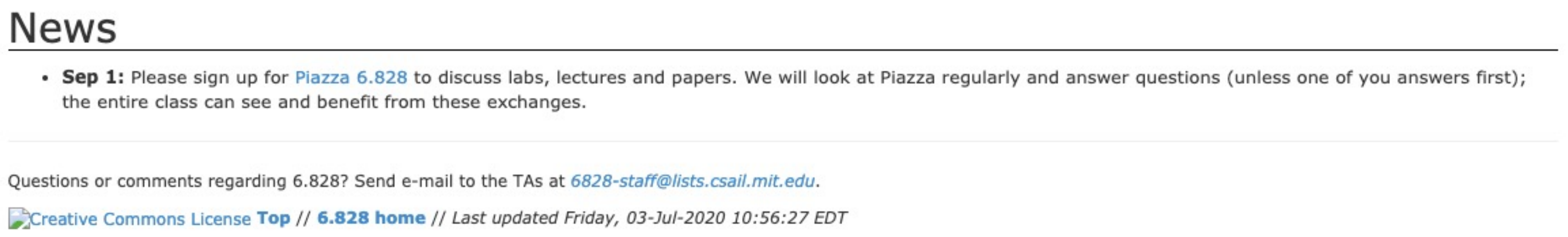
2021/09/15

Outline

- **JOS Overview**
- Course schedule & grading
- Some tips & tools
- Hands-on Lab 1: Bootloader

6.828: Operating System Engineering

JOS Overview


News

- **Sep 1:** Please sign up for Piazza 6.828 to discuss labs, lectures and papers. We will look at Piazza regularly and answer questions (unless one of you answers first); the entire class can see and benefit from these exchanges.

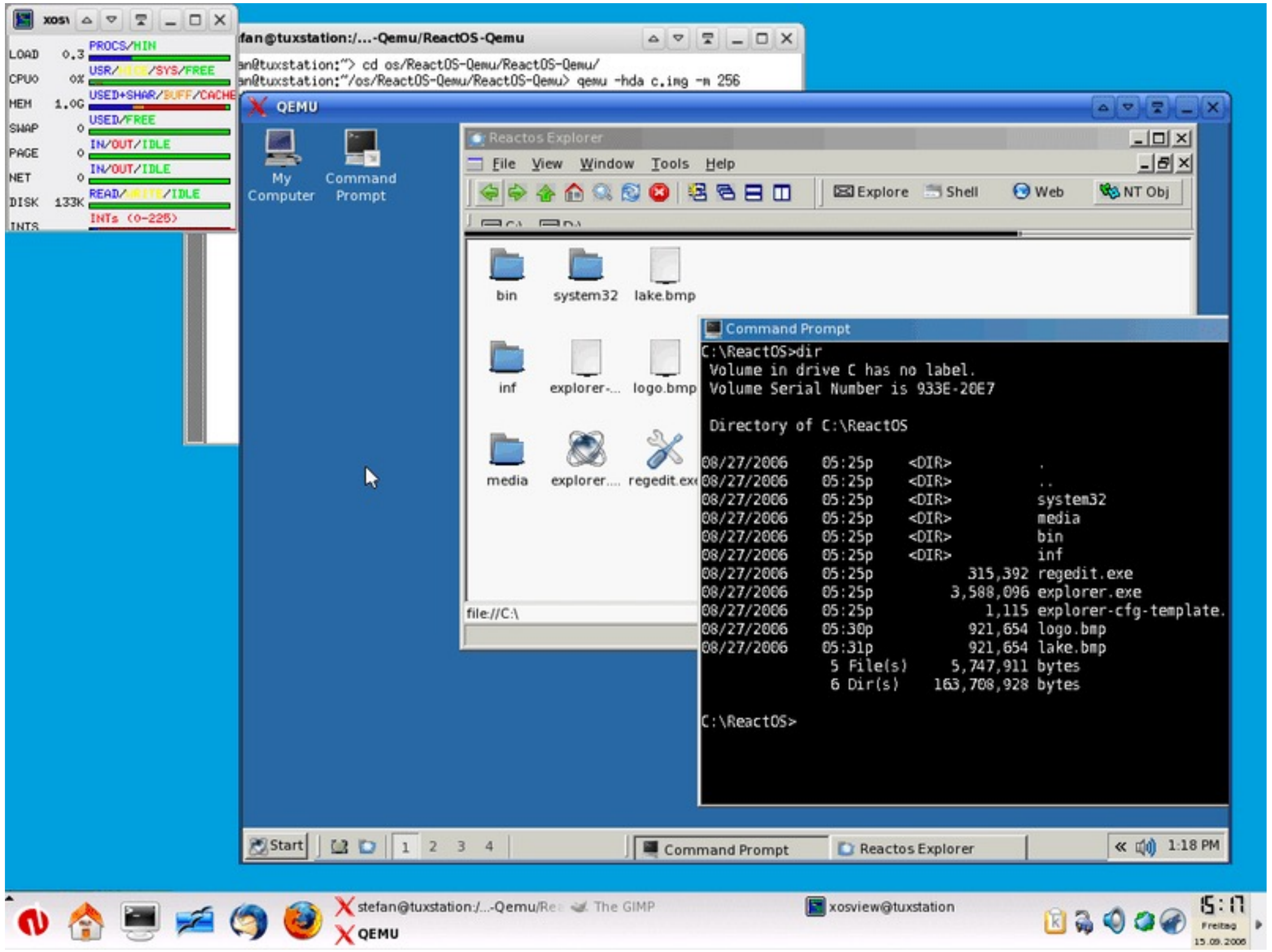
Questions or comments regarding 6.828? Send e-mail to the TAs at *6828-staff@lists.csail.mit.edu*.

Creative Commons License **Top** // **6.828 home** // *Last updated Friday, 03-Jul-2020 10:56:27 EDT*

- JOS is an x86-based OS designed by MIT for teaching
  - See website: https://pdos.csail.mit.edu/6.828/2018/
  - Note that we use course materials of **2018,** not 2019 or 2020
- To finish JOS, several labs are provided, in which we'll implement important OS components from scratch
  - Lab 1: bootloader
  - Lab 2: virtual memory management
  - Lab 3: user mode and system call
  - Lab 4: multitasking
  - Lab 5: file system
- We'll understand OS concepts better if we've actually implemented them

Lab environment

- JOS is a real OS that can run on real x86 hardware
- JOS will take over all hardware during runtime
- We cannot debug JOS as a common application
- Instead, we use **QEMU**, an emulator provides hardware emulation



```
an@tuxstation:~> cd os/ReactOS-Qemu/ReactOS-Qemu/
an@tuxstation:~/os/ReactOS-Qemu/ReactOS-Qemu> qemu -hda c.img -m 256
```

```
C:\ReactOS>dir
 Volume in drive C has no label.
 Volume Serial Number is 933E-20E7

 Directory of C:\ReactOS

08/27/2006    05:25p    <DIR>          .
08/27/2006    05:25p    <DIR>          ..
08/27/2006    05:25p    <DIR>          system32
08/27/2006    05:25p    <DIR>          media
08/27/2006    05:25p    <DIR>          bin
08/27/2006    05:25p    <DIR>          inf
08/27/2006    05:25p           315,392 regedit.exe
08/27/2006    05:25p         3,588,096 explorer.exe
08/27/2006    05:25p             1,115 explorer-cfg-template.
08/27/2006    05:30p           921,654 logo.bmp
08/27/2006    05:31p           921,654 lake.bmp
               5 File(s)      5,747,911 bytes
               6 Dir(s)     163,708,928 bytes

C:\ReactOS>
```

- See https://pdos.csail.mit.edu/6.828/2018/tools.html and https://pdos.csail.mit.edu/6.828/2018/labguide.html for more details

Outline

- JOS Overview
- **Course schedule & grading**
- Some tips & tools
- Hands-on Lab 1: Bootloader

Lab Schedule

Lab	Start time	End time	Weeks	Report Submission
Lab1	2021.9.16	2021.9.22	1 week	2021.9.30
Lab2	2021.9.23	2021.10.20	3 weeks	2021.10.24
Lab3	2021.10.21	2021.11.10	3 weeks	2021.11.14
Lab4	2021.11.11	2021.12.1	3 weeks	2021.12.5
Lab5	2021.12.2	2021.12.16	2 weeks	2021.12.19

Quizzes

- Objective: check the completion and the understanding of labs
- Schedule
  - There are two quizzes in this semester
  - Quizzes will be scheduled after lab3 and lab5

OS Lab: Grading (35%)

- Complete the lab requirements: 60%
  - Complete the coding tasks
  - Submit the report
  - **Plagiarize is strictly prohibited!!**
  - Due date is hard deadline!
    - Deduct 10% for each late day (3 days maximum for each lab)
- Quiz : 40%
  - Two quizzes: 20% each
- Bonuses: 10%
  - One or more challenge for each lab
  - Choose from a given list of challenges

Submission

- Archive both source code and report into one zip file
  - ID_name_#lab.zip, for example 1801111369_张子祺_lab1.zip
- Report requirement
  - Illustrate the detail implementation of each task, add code snippets if necessary
  - State which challenge do you choose at the beginning of the report
  - The submission format is **PDF.** You can write with latex, word or markdown but remember to transfer the final format into PDF.

Outline

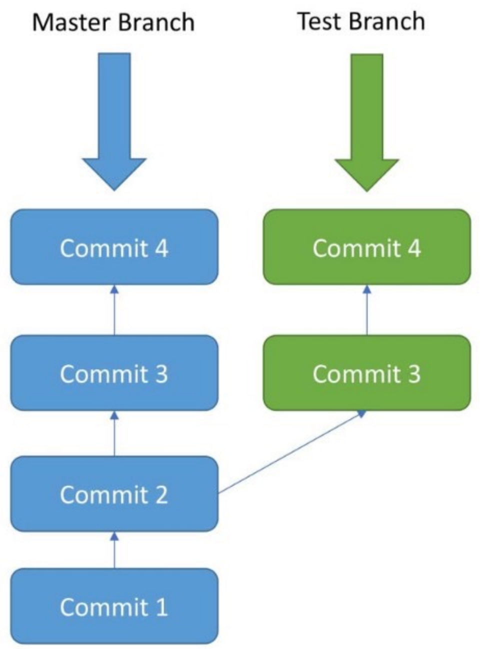
- JOS Overview
- Course schedule & grading
- **Some tips & tools**
- Hands-on Lab 1: Bootloader

Some tips

- Start early!
  - Debugging JOS can be time consuming
- **Read documents and understand existing code carefully before writing your own code**
  - JOS official lab guides
  - Intel Software Developer's Manual: https://software.intel.com/en-us/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4
  - OSDev Wiki: https://wiki.osdev.org/
- Maybe look for bugs in previous labs will help if you encounter strange bugs in current lab
  - JOS has an automated grader, but it only ensures basic correctness

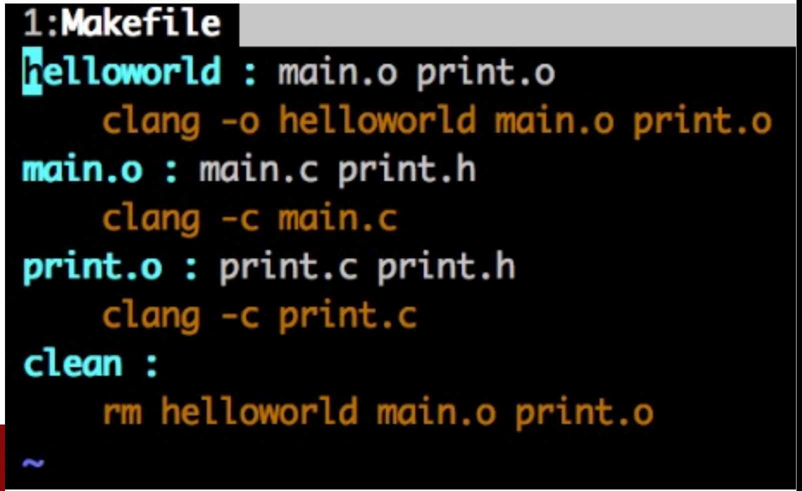
Tools: git

- JOS is a large project, you need to manage your complex code
  - You may forget a previous small change
  - Backup with remote repository
- Frequently used commands
  - Git clone
  - Git add, git commit –m
  - Git status, git log
  - Git reset
  - Git branch / git checkout
  - Git push

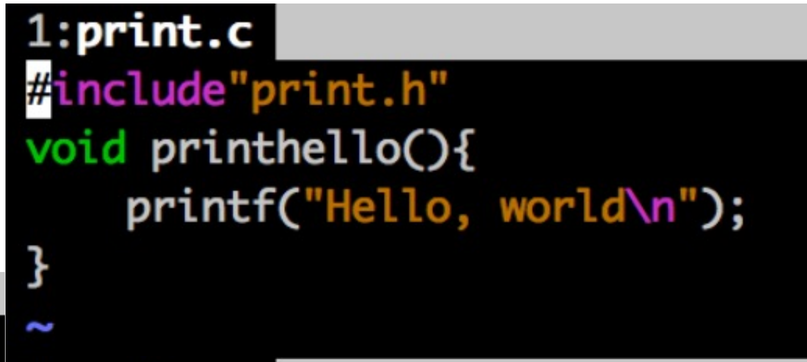


Tools

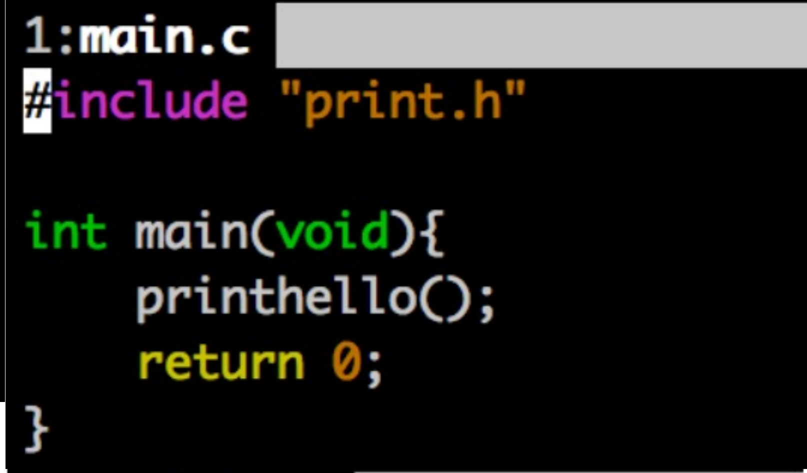
- Makefile
  - Tells ``make’’ how to compile and link a program
  - Make qemu
  - Make qemu-nox
  - Make qemu-gdb
  - Make qemu-nox-gdb
- GDB



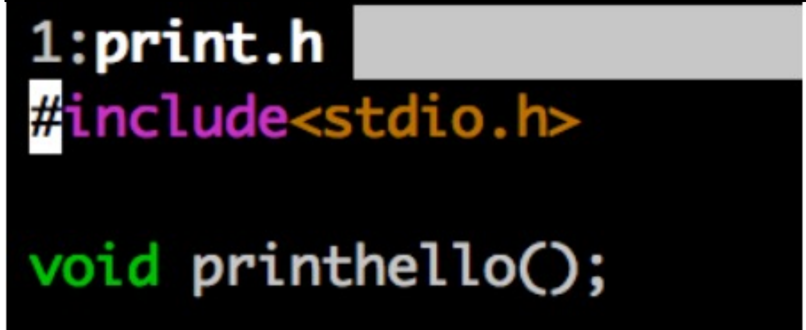
```
1:Makefile
helloworld : main.o print.o
	clang -o helloworld main.o print.o
main.o : main.c print.h
	clang -c main.c
print.o : print.c print.h
	clang -c print.c
clean :
	rm helloworld main.o print.o
~
```



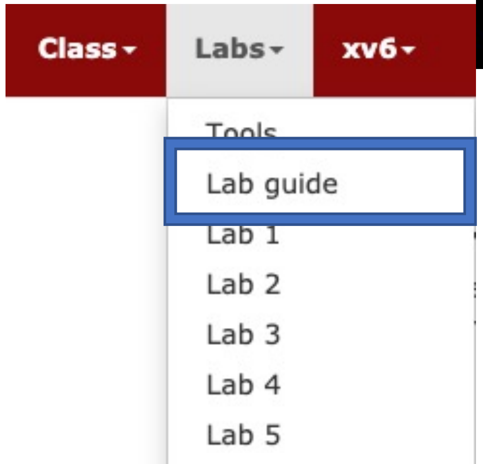
```
1:print.c
#include"print.h"
void printhello(){
    printf("Hello, world\n");
}
~
```



```
1:main.c
#include "print.h"

int main(void){
    printhello();
    return 0;
}
```



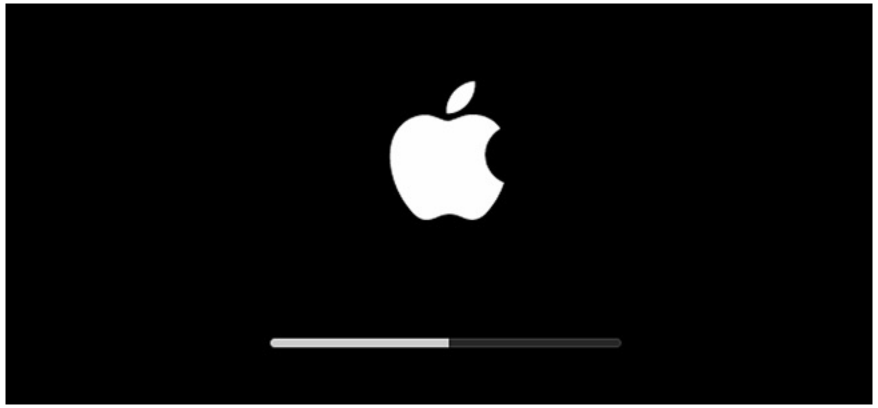
```
1:print.h
#include<stdio.h>

void printhello();
```

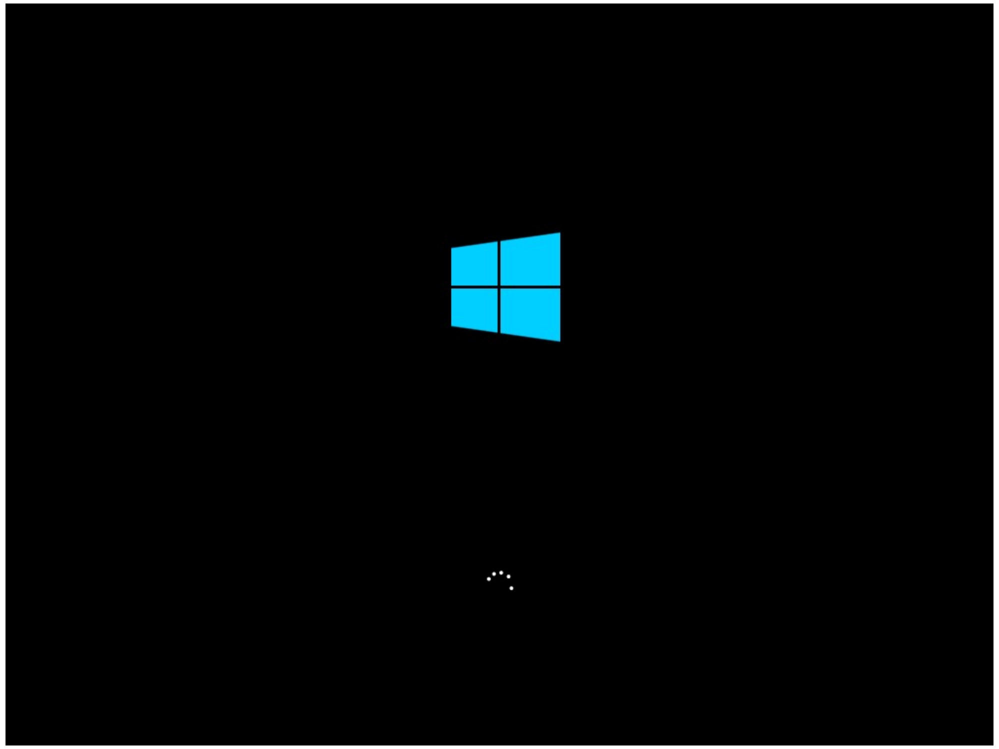


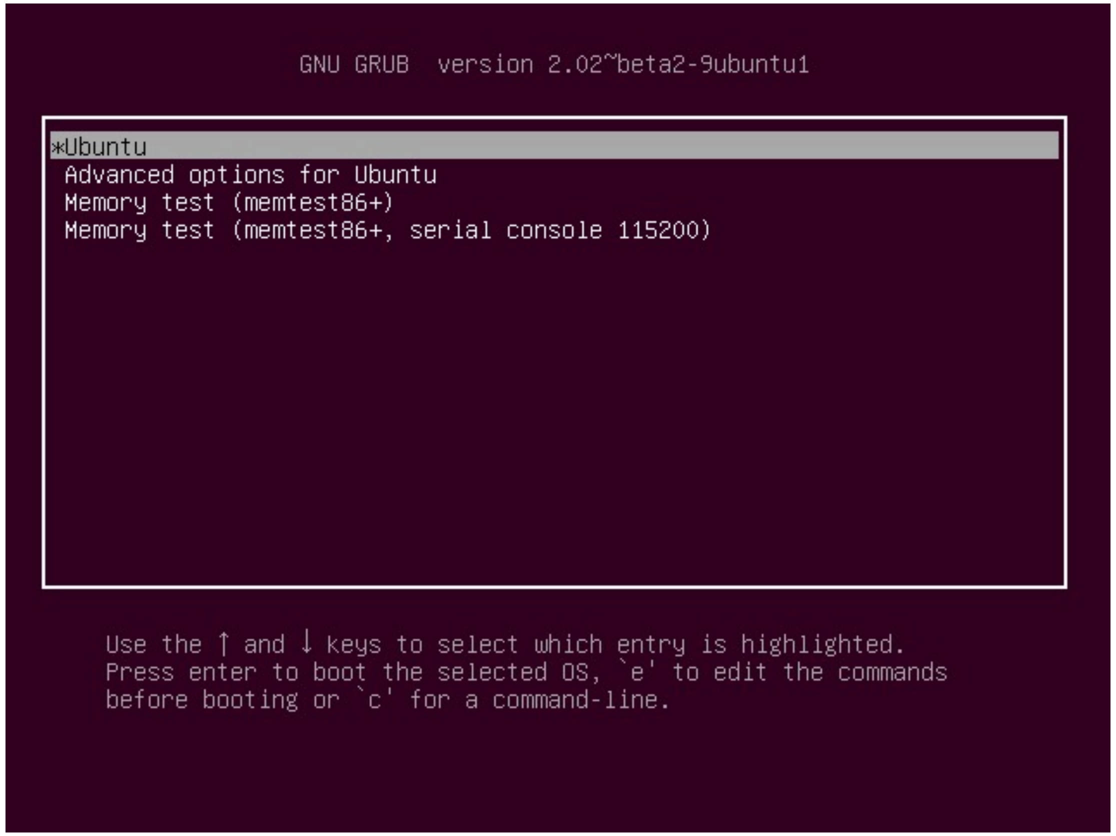
Outline

- JOS Overview
- Course schedule & grading
- Some tips
- **Hands-on Lab 1: Bootloader**

How does a PC boot?





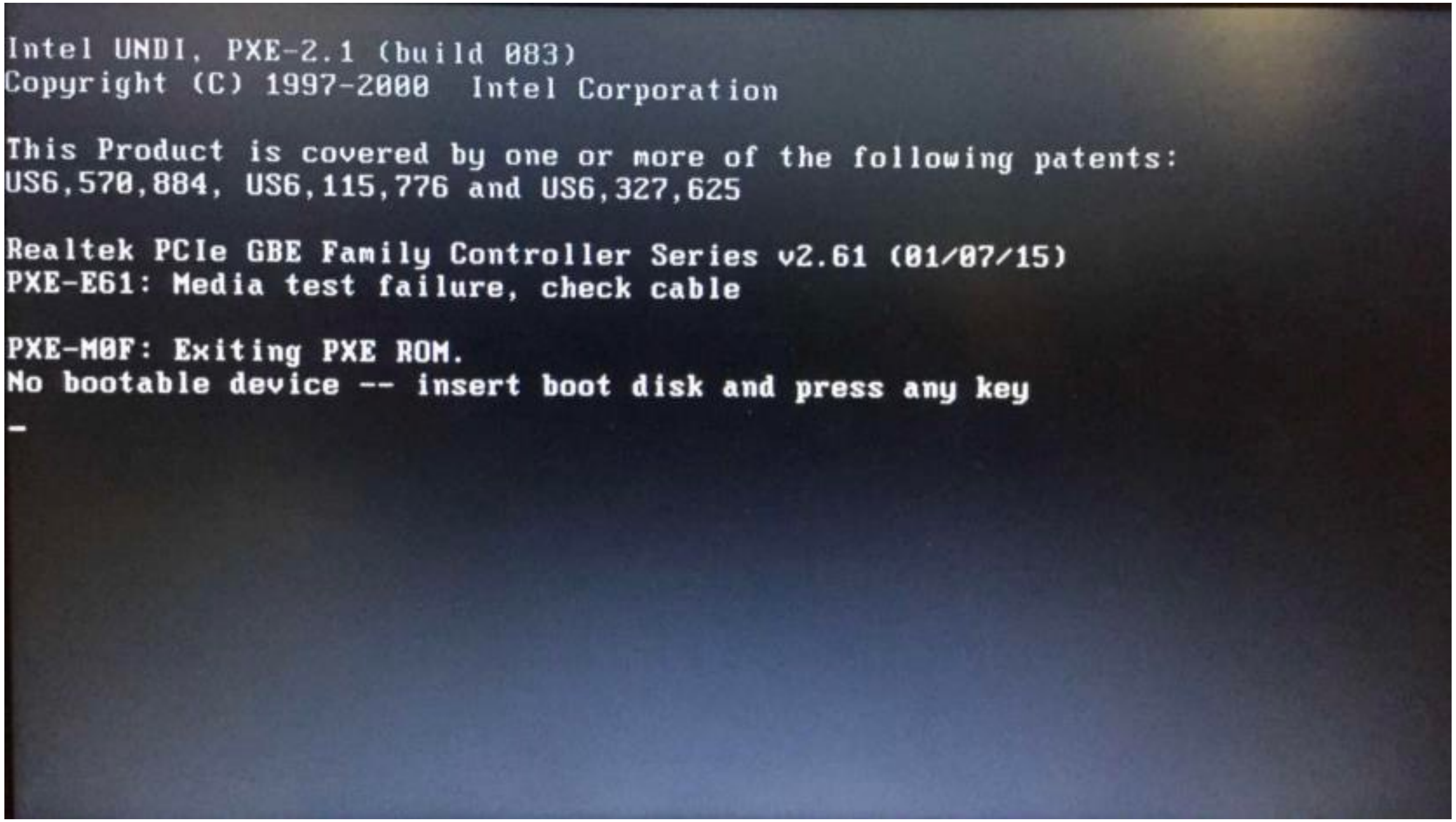


```
GNU GRUB  version 2.02~beta2-9ubuntu1

*Ubuntu
 Advanced options for Ubuntu
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
```

BIOS

- Recall what we've learned from ICS course
  - Once CPU is powered on, it fetches instructions from memory and modifies data in memory
  - The first instruction is stored in BIOS ROM
  - BIOS ROM is mapped to low-address space of RAM
  - The IBM PC starts executing at physical address `0x000ffff0`
- CPU then executes pre-defined instructions in BIOS ROM and does some self checking
- But what's next?

How your PC goes on strike…



```
Intel UNDI, PXE-2.1 (build 083)
Copyright (C) 1997-2000  Intel Corporation

This Product is covered by one or more of the following patents:
US6,570,884, US6,115,776 and US6,327,625

Realtek PCIe GBE Family Controller Series v2.61 (01/07/15)
PXE-E61: Media test failure, check cable

PXE-M0F: Exiting PXE ROM.
No bootable device -- insert boot disk and press any key
_
```

Bootloader

- We need a **bootable device**
- Of course, not all devices (disks) are bootable
- Disks have sectors
  - Hard disks have 512-byte sectors
  - The first sector is used to flag whether the disk is bootable
  - MBR (master boot record) is the first sector ends with **0x55** and **0xAA**. This is manually set.
  - The disk is bootable ⇔ MBR exists
  - Bootloader are stored in MBR
- BIOS will recognize MBR if it exists on some disk, then load bootloader into RAM and finally jump to the first instruction of it

OS loading

- In bootloader we need to load OS into RAM
- OS are stored in disk. How to load it?
  - x86 has special I/O instructions
  - Consult JOS lab 1 code for more details
- Finally, OS takes over the control flow

Lab 1 has begun

- Lab 1 due: 9/30
- Website: https://pdos.csail.mit.edu/6.828/2018/labs/lab1/
- What you need to do
  - Finish lab 1 according to instructions
  - Write lab 1 report according to template provided
- What to submit
  - Lab 1 source code
  - Lab 1 report (in PDF format)
- Where to submit
  - http://course.pku.edu.cn/
- Start early! Start early! Start early!