



# Operating Systems Concept

施 吉 昇

► Daniel Shih/Chung-Wei Lin  
► National Taiwan University

Spring 2022

# Introduction to the course

# Who am I?

- ▶ 施吉昇 Chi-Sheng (Daniel) Shih
  - PhD, University of Illinois at Urbana-Champaign
- ▶ Research Interests:
  - Real-Time Scheduling Theory
  - Embedded System Software
  - Multi-dimension Resource Management
  - Hardware/Software Co-Design for SoC
  - IoT/M2M and Cyber-Physical Systems
- ▶ My Lab:
  - Embedded Systems and Wireless Networking Lab
  - <http://newslabx.csie.ntu.edu.tw>
- ▶ My website: <http://www.csie.ntu.edu.tw/~cshih>

# Enrollment

- ▶ Submit your request on-line: <https://0rz.tw/flGmd>
- ▶ Due: 11:59PM, Feb. 16, 2022 (Wednesday)



- ▶ Switching the enrollment between sections is not allowed
  - ▶ You are free to directly attend another section
- ▶ Auditing in-person lectures is not allowed
  - ▶ NTU students can submit requests (above) and access videos

# Administration Misc.

- ▶ Course Materials
  - NTU COOL: <https://cool.ntu.edu.tw/courses/11767>
  - You will find the following on the site:
    - Course slides:
    - Course announcement
    - MP discussions
- ▶ Machine Problems will be announced on NTU Cool.
- ▶ Report submission and grading
  - ▶ Gradescope: the link will be provided with a following MP

# Syllabus

- ▶ Online syllabus
  - ▶ The two classes will have same exams, and same programming assignments.
  - ▶ Both classes will use the same required textbook and cover the required topics.
  - ▶ Final grade may be adjusted at different classes/sections.

# Textbook

► Required Textbook

- Operating system concepts by “Galvin, Peter B., Greg Gagne, and Abraham Silberschatz, John Wiley and Sons, ISBN 978-1-118-06333-0. It is distributed by Ten-Long (天瓏圖書) in Taiwan.

► Reference Books:

- “Xv6, a simple Unix-like teaching operating system” by R. Cox, M.F. Kaashoek, & R. Morris, 2011, <http://pdos.csail.mit.edu/6.828/2012/xv6.html>.
- “Advanced Programming in the Unix Environment” third edition by W. Richard Stevens and Stephen A. Rago, Addison-Wesley, 2013. It is distributed by 開發圖書有限公司.
- Understanding Unix/Linux Programming: A Guide to Theory and Practice, Molay, Prentice Hall, ISBN: 0130083968

- ▶ Source codes shown in the text-book

- <https://github.com/greggagne/osc10e> or
- Our website

# Class Schedule

Week #	Date	Topic	Textbook: OSC 10e	References : xv6	Lab Assignment Due
1	22/02/15	Course Introduction		Ch0	
2	22/02/22	Overview	Ch1 and Ch2		
3	22/03/01	Processes, Threads, and Concurrency	Ch3	Ch1	MP0: Setup xv6 due
4	22/03/08		Ch4		MP1 Announce
5	22/03/15	Memory Management	Ch9	Ch2	
6	22/03/22		Ch10	Ch2	MP1: Process and Thread Due/MP2 Announce
7	22/03/29	I/O Systems	Ch12		
8	22/04/05	Spring Break	Ch12		MP2 Memory Due
9	22/04/12	Midterm			
10	22/04/19	CPU Scheduling	Ch5	Ch7	
11	22/04/26				MP3 Announce
12	22/05/03	Mass-Storage and File System	Ch11, Ch13	Ch8	
13	22/05/10		Ch14, Ch15		MP3: Scheduling Due/MP4 Announce
14	22/05/17	Synchronization	Ch6 and Ch7	Ch6	
15	22/05/24				MP4: File Systems Due
16	22/05/31	Final Exam			



# Grading

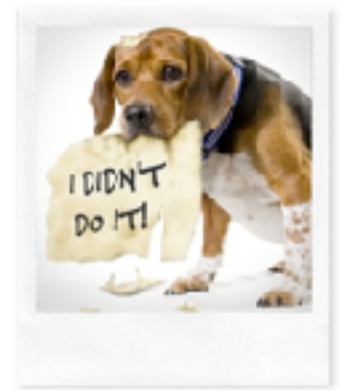
► Grading:

- Mid-term exam: 28 points,
- Final exam: 28 points, and
- 1+4 programming assignments: 44 points
  - MP0: 4 points
  - MP1 ~ 4: 10 points for each
- Bonus:
  - Submit a short self-review report with the MP4 report
  - Provide "evidence" (e.g., a post on NTU COOL) how you helped others

# Machine Problems

- ▶ TA: 林祥瑞, 劉昕佑, 曾奕青, 林家佑, 周良冠, 賈本皓
- ▶ MP0: set up xv6.
- ▶ In these machine problems, you will learn how the fundamental OS functions are implemented.
  - MP1: Thread
  - MP2: Shared Memory
  - MP3: Scheduling
  - MP4: File Systems
- ▶ Bonus and Penalty:
  - Early bird bonus MAY be applied if submitted 5 days before deadline.
    - bonus: 20%-more per day (subject to change)
  - Late penalty will be applied after submission deadline.
    - Penalty: 20%-off per day

# Plagiarism



- ▶ There is NO tolerance for plagiarism.
- ▶ You can discuss the assignments with your classmates and/or friends. However, you MUST write the codes by yourself.
- ▶ It is YOUR responsibility to protect your own codes so that please do not leave your codes on the table or screen, or unlocked directory.
  - On github, you will create 'PRIVATE' repository for your assignments and should not change it to 'PUBLIC'
  - Change the permission of your home directory on workstations.
- ▶ We will cross check your assignment with others'.

Options -l c -m 10

---

[ [How to Read the Results](#) | [Tips](#) | [FAQ](#) | [Contact](#) | [Submission Scripts](#) | [Credits](#) ]

---

File 1	File 2	Lines Matched
<a href="#">b9190xxxx.c (97%)</a>	<a href="#">b9190xxx.c (97%)</a>	81
<a href="#">b9190xxxx.c (89%)</a>	<a href="#">b9190xxxx.c (89%)</a>	53
<a href="#">b9190xxxx.c (96%)</a>	<a href="#">b9190xxxx.c (96%)</a>	52
<a href="#">b9190xxxx.c (77%)</a>	<a href="#">b9190xxxx.c (77%)</a>	42
<a href="#">b9190xxxx.c (60%)</a>	<a href="#">b9190xxxx.c (58%)</a>	45
<a href="#">b9190xxxx.c (74%)</a>	<a href="#">b9190xxxx.c (73%)</a>	34
<a href="#">b9190xxxx.c (47%)</a>	<a href="#">b9190xxxx.c (45%)</a>	17
<a href="#">b9090xxxx.c (43%)</a>	<a href="#">b9090xxxx.c (44%)</a>	31
<a href="#">b9190xxxx.c (47%)</a>	<a href="#">b9190xxxx.c (51%)</a>	35
<a href="#">b9190xxxx.c (28%)</a>	<a href="#">b9190xxxx.c (33%)</a>	10
<a href="#">b9190xxxx.c (56%)</a>	<a href="#">b9190xxxx.c (35%)</a>	30
<a href="#">b9190xxxx.c (35%)</a>	<a href="#">b9190xxxx.c (56%)</a>	27
<a href="#">b9190xxxx.c (45%)</a>	<a href="#">b9190xxxx.c (35%)</a>	18
<a href="#">b9190xxxx.c (34%)</a>	<a href="#">b9190xxxx.c (32%)</a>	28

b9190XXXX.c (97%)		b9190XXXX.c (97%)	
9-83		10-90	

## b9190XXXX.c

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#define max 8192
```



```
int main(int argc, char *argv[])
{
    int i, j;
    int fdin, fdout;
    long lenCurrent; /* saving open file's offset */
    int lenRead; /* bytes of read and write */
    char BI[max]; /* save in read buff */
    char BO[max]; /* save in write buff */

    if(argc != 3)
    {
        fprintf(stderr, "Failure\n");
        exit(1);
    }

    if((fdin = open(argv[1], O_RDONLY)) == -1) /* c
```

## b9190XXXX.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#define BUFSIZE 8192
```



```
int main(int argc, char *argv[])
{
    int i, j;
    int fdin, fdout;
    long CurrentLength; //save open file's current
    int ReadLength; //used as read & write's n
    char ByteIn[BUFSIZE]; //used to save in read
    char ByteOut[BUFSIZE]; //used to save in write

    if(argc != 3)
    {
        fprintf(stderr, "Usage:Copy Infile Outfile\n");
        exit(1);
    }

    //open file
    if((fdin = open(argv[1], O_RDONLY)) == -1)
```

# Participation

- ▶ Q & A for machine problems will be hosted on NTU Cool.
  - ▶ Both TA and students are encouraged to answer the questions.
  - ▶ Helping others can be counted toward bonus grade.

# Next

- ▶ MP0 Announcement
- ▶ 2<sup>nd</sup> Part of this lecture:
  - ▶ Live Stream via WebEx: <https://bit.ly/3sF7ax6>
  - ▶ Why Studying Operating Systems?
  - ▶ Write your first OS



# Why Studying Operating Systems?





Alan Mathison Turing  
23 June 1912 – 7 June 1954



Alan Mathison Turing  
23 June 1912 – 7 June 1954

# Inventor of Turing Machine

**Turing Machine is the theoretic foundation of modern computers.**

# Turing Machine

By Alan Turing in 1936

- ▶ They were first named 'Turing machines' by Alonzo Church in a review of Turing's paper (Church 1937)



# Turing Machine

By Alan Turing in 1936

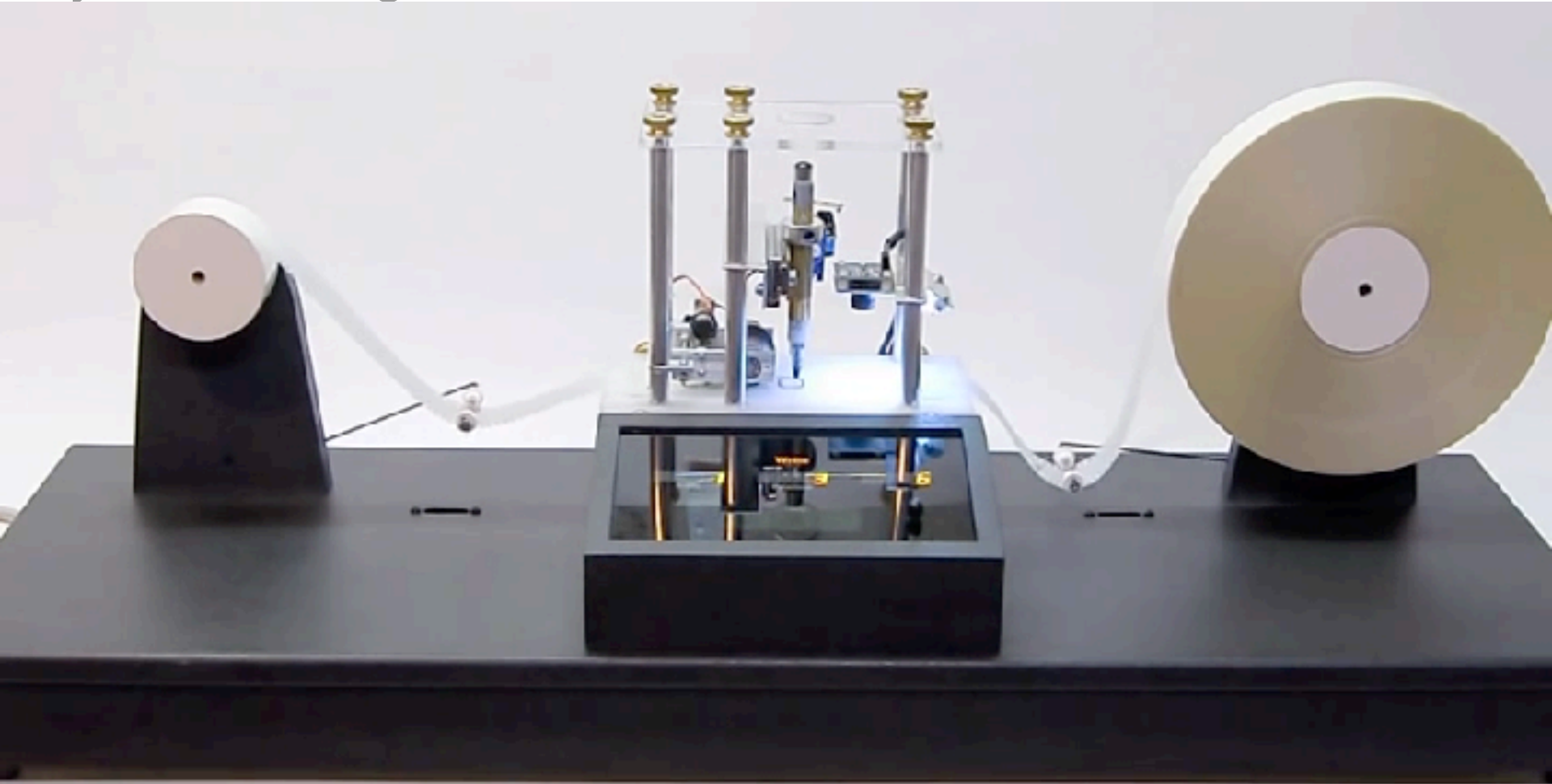
- ▶ They were first named 'Turing machines' by Alonzo Church in a review of Turing's paper (Church 1937)



- ▶ Definition:
  - ▶ Infinite memory as input
  - ▶ State machine as thinking process
  - ▶ Pen/Erase as output of the thinking process

# Turing Machine

By Alan Turing in 1936





Alan Mathison Turing  
23 June 1912 – 7 June 1954

# Did Alan Turing know Operating Systems?

## Why Should I?

我只想寫 **挖礦** 程式，  
為何逼我學作業系統？是要逼死誰？



花雖美。但是，有股淡淡的憂傷！！





懂 OS 的人，可以輕易地讓你的心血  
化為烏有！！

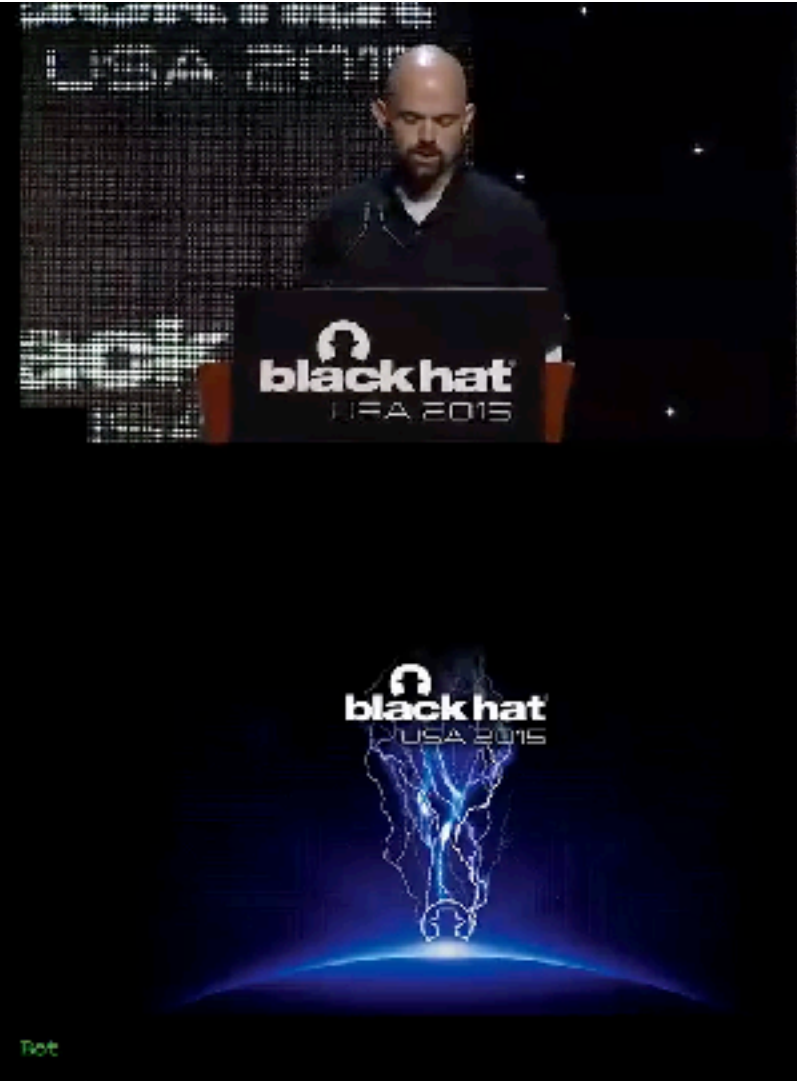


# Difficult to steal the root?

```
3 #include <unistd.h>
4 #include <sched.h>
5
6 int main(void)
7 {
8     long long a;
9     long long b;
10    long long i;
11
12    cpu_set_t mask;
13    CPU_ZERO(&mask);
14    CPU_SET(0,&mask);
15    sched_setaffinity(0, sizeof(mask), &mask);
16
17    for (i=0; i<0x100000000; i++) {
18        a=0x19a0f5019cc762cLL;
19        b=a;
20    }
21
22    exec1("/bin/sh", "/bin/sh", NULL);
23
24    return 0;
25 }
```

"escalate.c" 25L, 358C

16,22-28 Rot



Watch the full video on YouTube: searching for “The Memory Sinkhole - Unleashing “

# One more - Steal your computer



# Who implementing their own Operating Systems?

Why not downloading one from github or google?





# How difficult it is to write an OS?

- ▶ One complex and complete OS can take million lines of code.
  - ▶ MacOS 10.4@2005: 86 Millions of SLoC
  - ▶ Debian 7.0@2012: 419 Millions of SLoC
  - ▶ Linux Kernel 4.2@2015: 20.2 Millions of SLoC
- ▶ But
  - ▶ Linux Kernel 0.01@1991: 0.010239 Millions of SLoC

# Can we write one now?

# How to say 'Hello World' from your computer?

Tell the CPU how to start

Tell the CPU where to start

Create a main function

Create a print string function

Call the interrupt to display

```
printf("Hello, World.")
```



# How to say 'Hello World' from your computer?

Tell the CPU how to start

Start with 16bits mode

Tell the CPU where to start

Jump to [0x7C00](#) where boot sectors are loaded.

Create a main function

Run main function

Create a print string function

Call `printstr()` function

Call the interrupt to display

Call int 10h interrupt

`printf("Hello, World.")`

Show the character one by one

# Memory allocation after calling int 19h: load MBR after power on

(defined by BIOS)

Address	Purpose	Size
0x0000h	Interrupt Vectors	16KB
0x0400h	BIOS data area	
0x05??h	OS Load Area	
0X7C00h	Boot Sector (MBR)	512B
0x7E00h	Boot data/stack	512B
0x7FFFh	(Not used)	32HB

# Make the program to Run

- ▶ There is no operating system on the virtual machine.
- ▶ The CPU must be notified the bootable disk (image) and executable.
- ▶ Create the executable file:
  - ▶ Compile the assemble into executable.
- ▶ Prepare a bootable disk image
- ▶ Attache the bootable disk image to virtual machine.

Any  
Questions?

See You  
Next Class