中国科学院大学
University of Chinese Academy of Sciences

操作系统课程 B0911010Y-01

操作系统
课程简介

中国科学院大学计算机与控制学院
中国科学院计算技术研究所

2019-08-26

内容提要

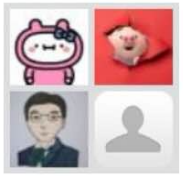
- 课程信息
- 什么是系统？
- 什么是操作系统？
- 怎么学习操作系统？

课程信息

- 课程网站：http://sep.ucas.ac.cn
- 授课教师：
  - 陈明宇 中科院计算所 cmy@ict.ac.cn
  - 蒋德钧 中科院计算所 jiangdejun@ict.ac.cn
- 助教：
  - 李海锋 lihaifeng@ict.ac.cn
  - 崔聪祎 cuicongyi@ict.ac.cn
- 同期课程：
  - 操作系统2班：包云岗
  - 操作系统研讨课：蒋德钧

课程信息

- 微信群
  - /*操作系统2019秋季1班*/



操作系统2019秋季1班

该二维码7天内(9月2日前)有效，重新进入将更新

课程教材

- 课程教材
  - 现代操作系统（第4版）Andrew S. Tanenbaum，Herbert Bos 著
    - 英文版：机械工业出版社 ISBN:9787111581659
    - 中文版：陈向群、马洪兵等译，机械工业出版社 ISBN: 9787111573692
- 参考教材（推荐）
  - Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

    http://pages.cs.wisc.edu/~remzi/OSTEP/
- 参考课件
  - Princeton COS 318

实例分析

- 实例分析课程内容
  - 通过阅读实际的操作系统代码，加深对操作系统原理和设计的理解
- 实例分析课时安排
  - 共4次，每次4个课时
  - 2课时详细分析XV6代码
    - 课前分组阅读代码
    - 课上抽签，请几个组给大家分析交流一下对代码的理解
  - 2课时介绍操作系统前沿研究（根据课时进度安排）

实例分析

- 实例分析课分组方式
  - 3~4个人一组，自由组合
- 实例分析课抽签方式
  - 每次抽4~5个组
    - 提前一周抽组号以及各组负责主讲的部分
    - 提前两天抽组内主讲人
      - 主讲人负责在课上介绍自己组负责的部分
      - 如果最终不是抽到的同学讲代码的话，这部分成绩按80%计算
      - 遇到生病或者有事无法讲的情况，需要提前至少一天告诉助教，并重新抽主讲人

实例分析

- XV6：在x86处理器上用ANSI标准C重新实现的Unix第六版
  - https://pdos.csail.mit.edu/6.828/2018/xv6.html
- Linux: 4.12.10源码
  - https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.12.10.tar.xz
- 阅读部分
  - （1）进程与线程管理；（2）并发、同步和通信机制；（3）虚拟内存；（4）文件系统
  - XV6和Linux 4.12.10源码相结合
- 代码阅读工具
  - SourceInsight，UltraEdit，Vim + ctag + cscope

上课时间

- 教室：阶二5
- 时间： 单周一、三；双周只上周一
  - 特殊情况注意通知

2019－2020学年秋季学期本科教学日历

年度	2019																		2020	
月份	九月					十月					十一月				十二月				一月	
周次	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
星期一	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30	6
星期二	27	3	10	17	24	1 国庆节	8	15	22	29	5	12	19	26	3	10	17	24	31	7
星期三	28	4	11	18	25	2	9	16	23	30	6	13	20	27	4	11	18	25	1 元旦	8
星期四	29	5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9
星期五	30	6	13 中秋节	20	27	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10
星期六	31	7	14	21	28	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11
星期日	1	8	15	22	29 补 周五课	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12

考核方法

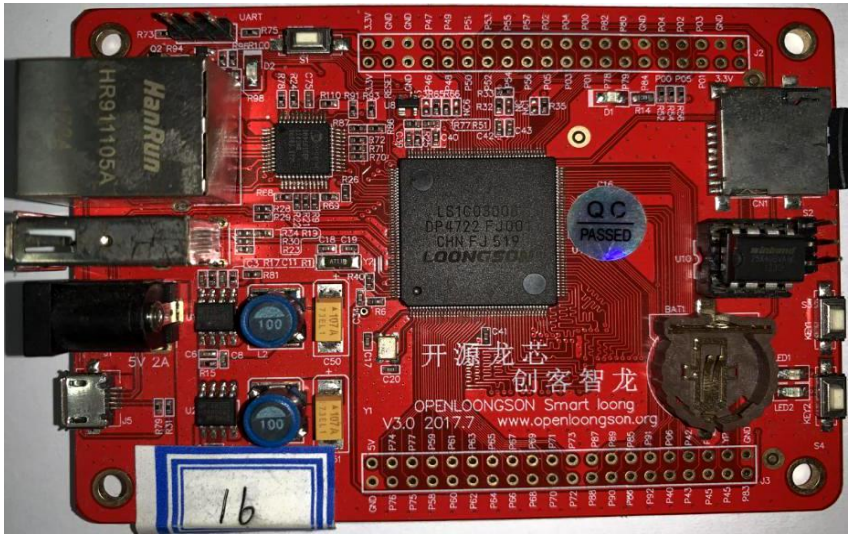
- 平时作业：20%
- 实例分析：20%
- 期中考试：20%
- 期末考试：40%

前序课程

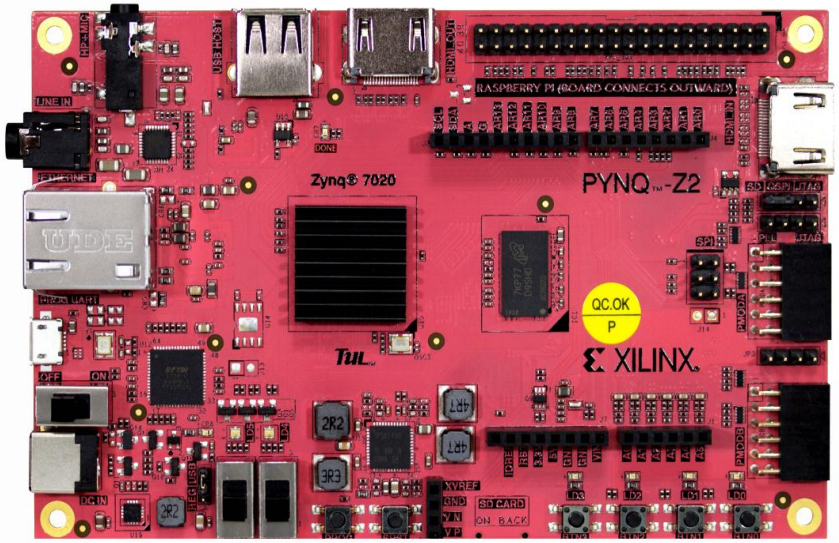
- 前序基础课程
  - 程序设计（C语言）
  - 数据结构
  - 计算机组成原理

研讨课

- 实现一个小型操作系统
  - MIPS版本
  - RISC-V版本



- 采用成熟稳健的龙芯处理器
- MIPS 32位指令集
- 成熟稳定



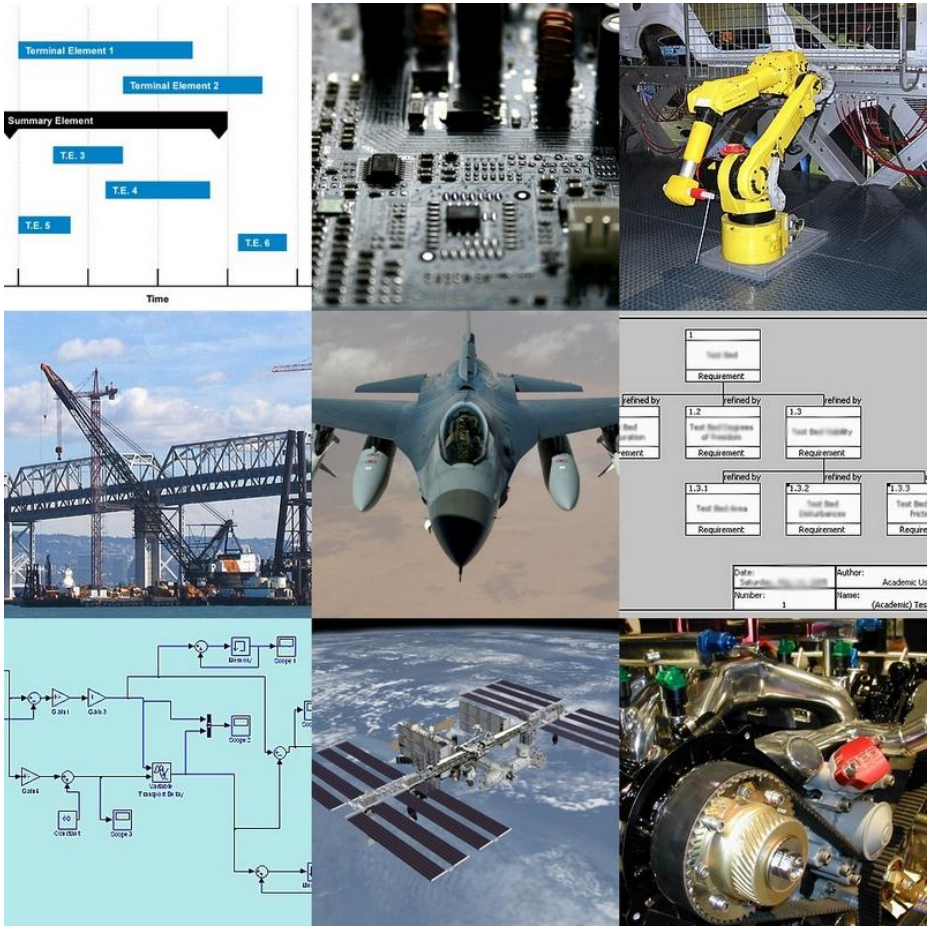
- 采用新兴的RISC V处理器
- RISCV 64位指令集
- 挑战与机遇并存

内容提要

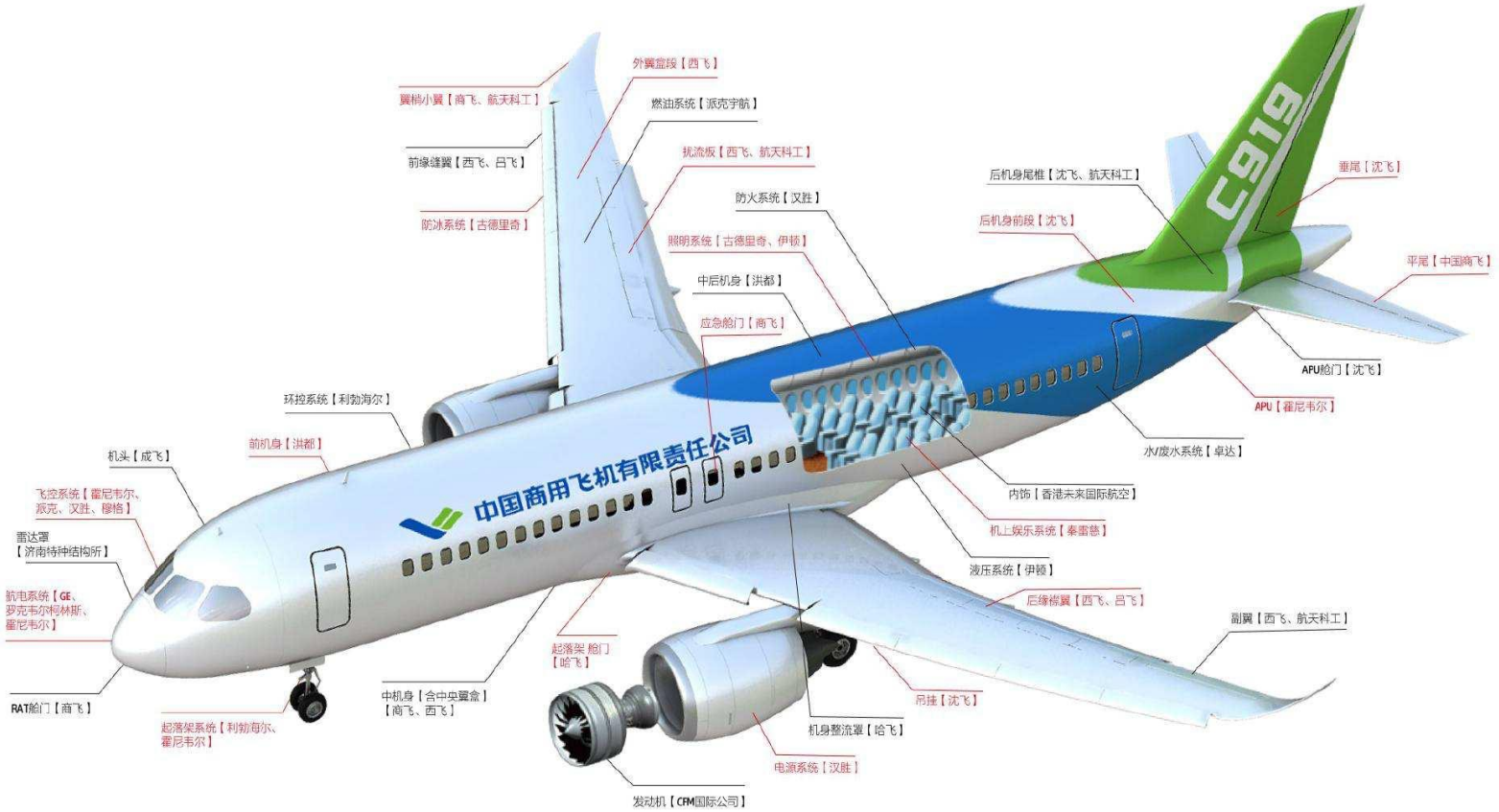
- 课程信息
- 什么是系统？
- 什么是操作系统？
- 怎么学习操作系统？

系统（System）

- 由一群有关联的个体组成，根据某种规则运作，能完成个别元件不能单独完成的工作的群体



飞机系统（全系统）



[图片来源: Baidu]

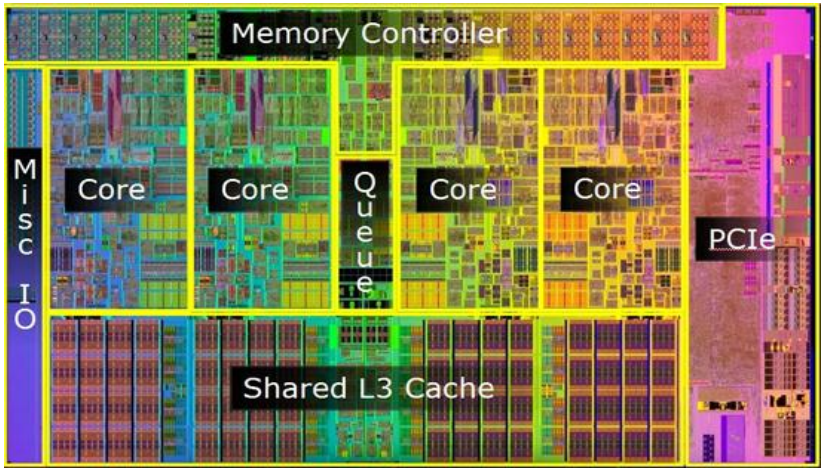
飞机系统（组件）

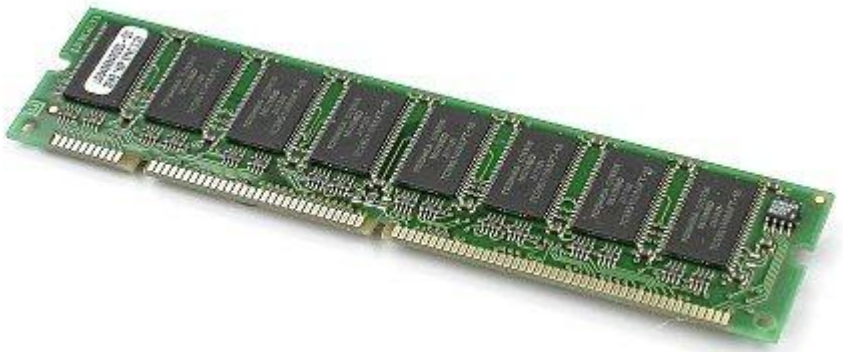
[图片来源: Internet]

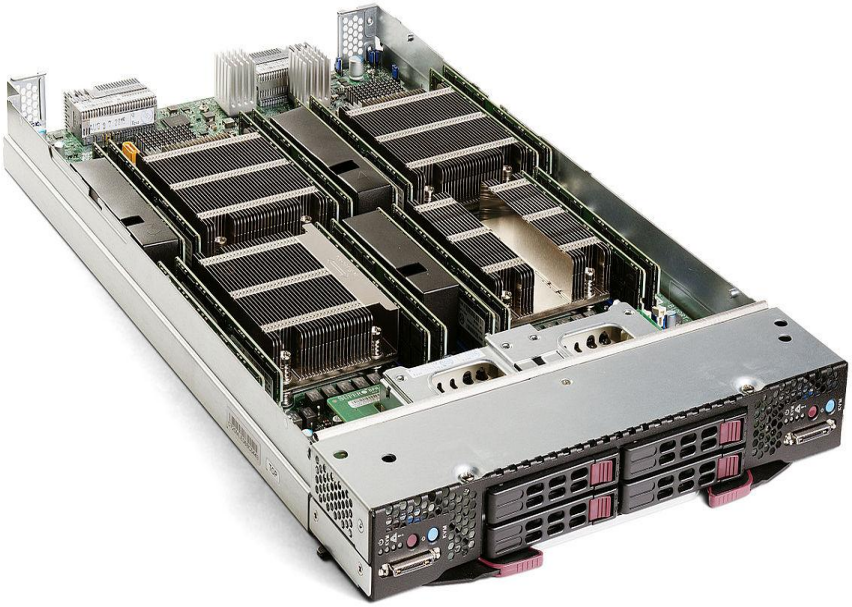
飞机系统（规模）

[图片来源: Youtube]

计算机系统（硬件）





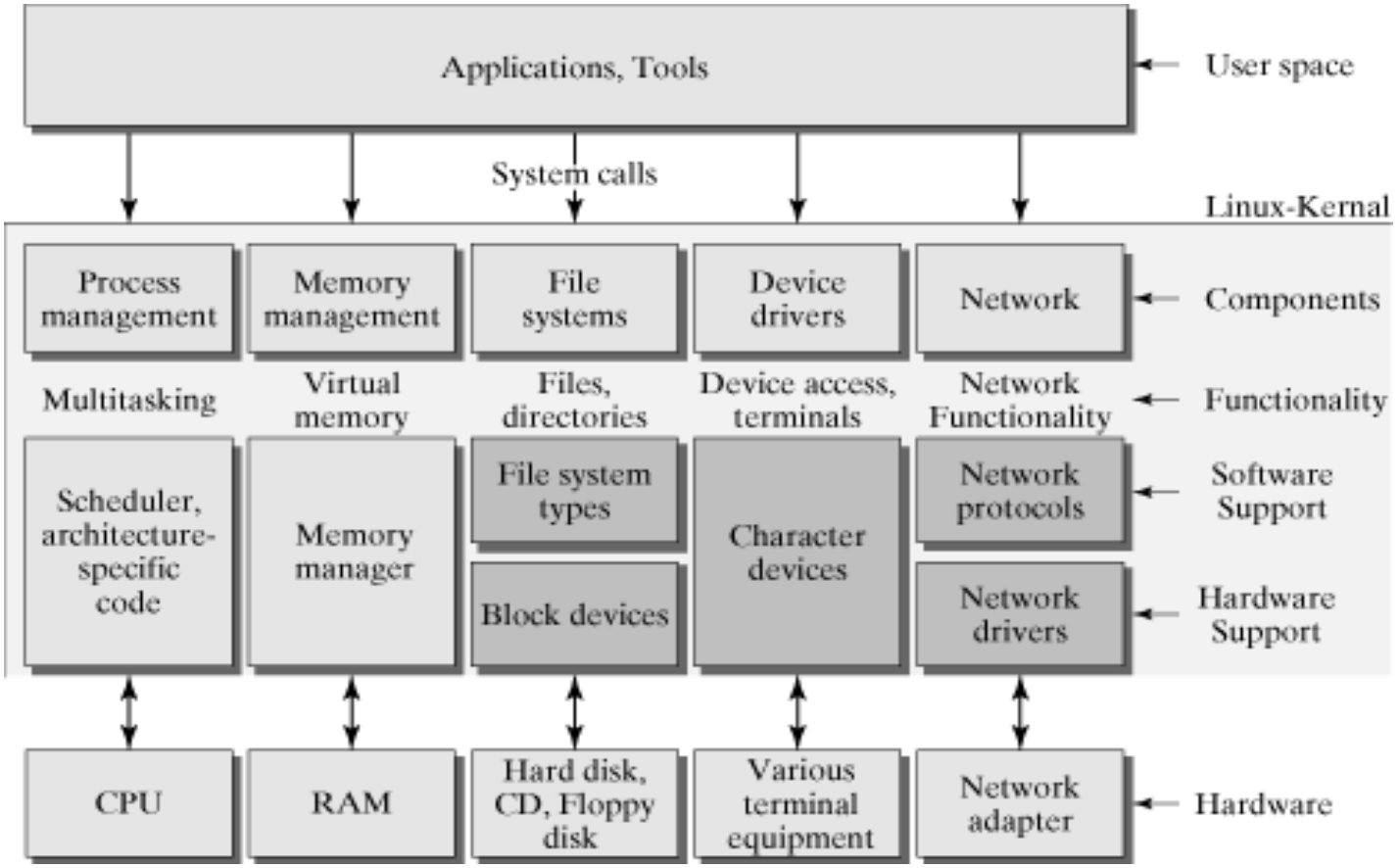


[图片来源: Wikipedia]

计算机系统（硬件）

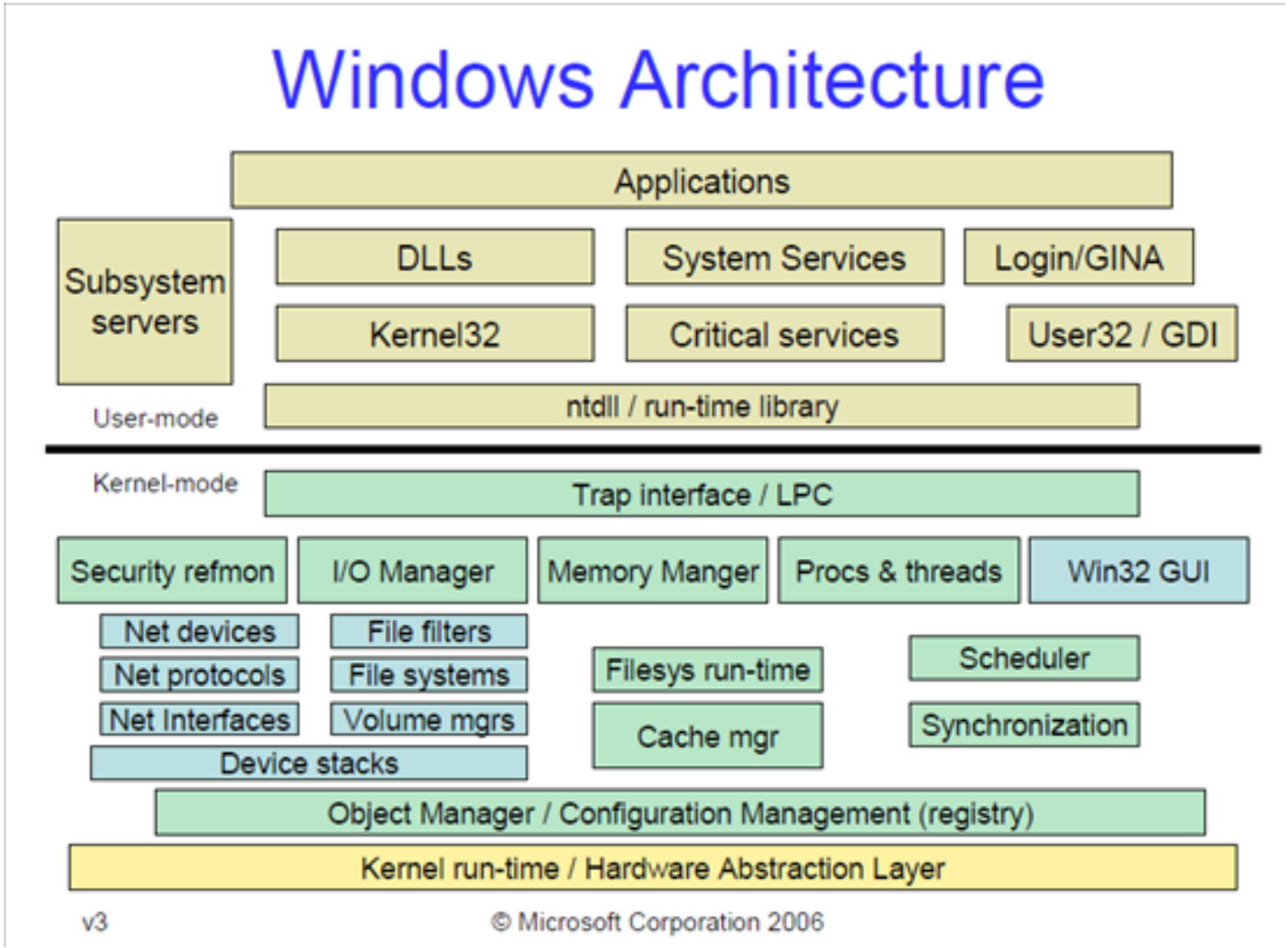
[图片来源: Baidu]

计算机系统（软件）



[图片来源：linux-india]

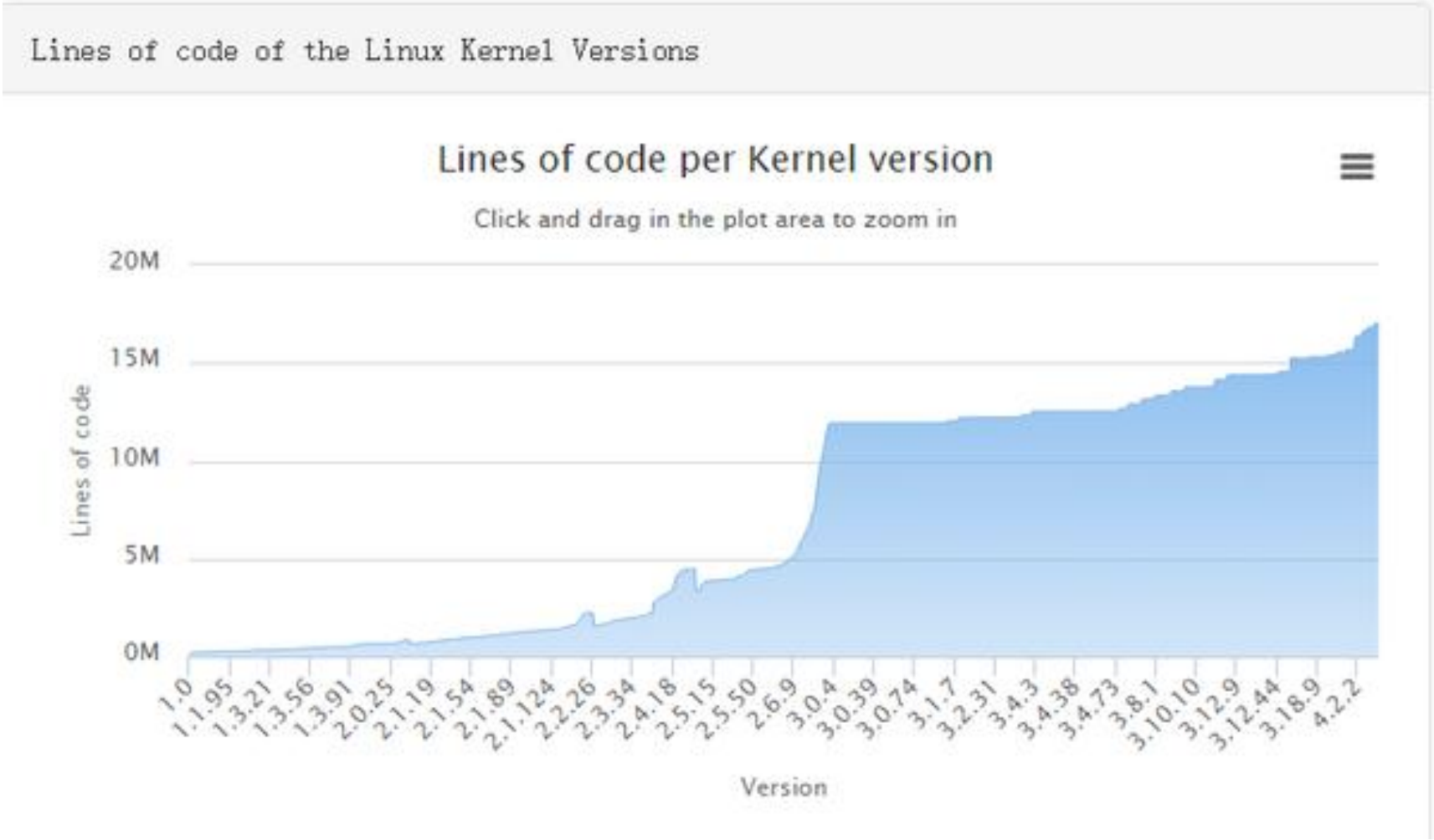
计算机系统（软件）



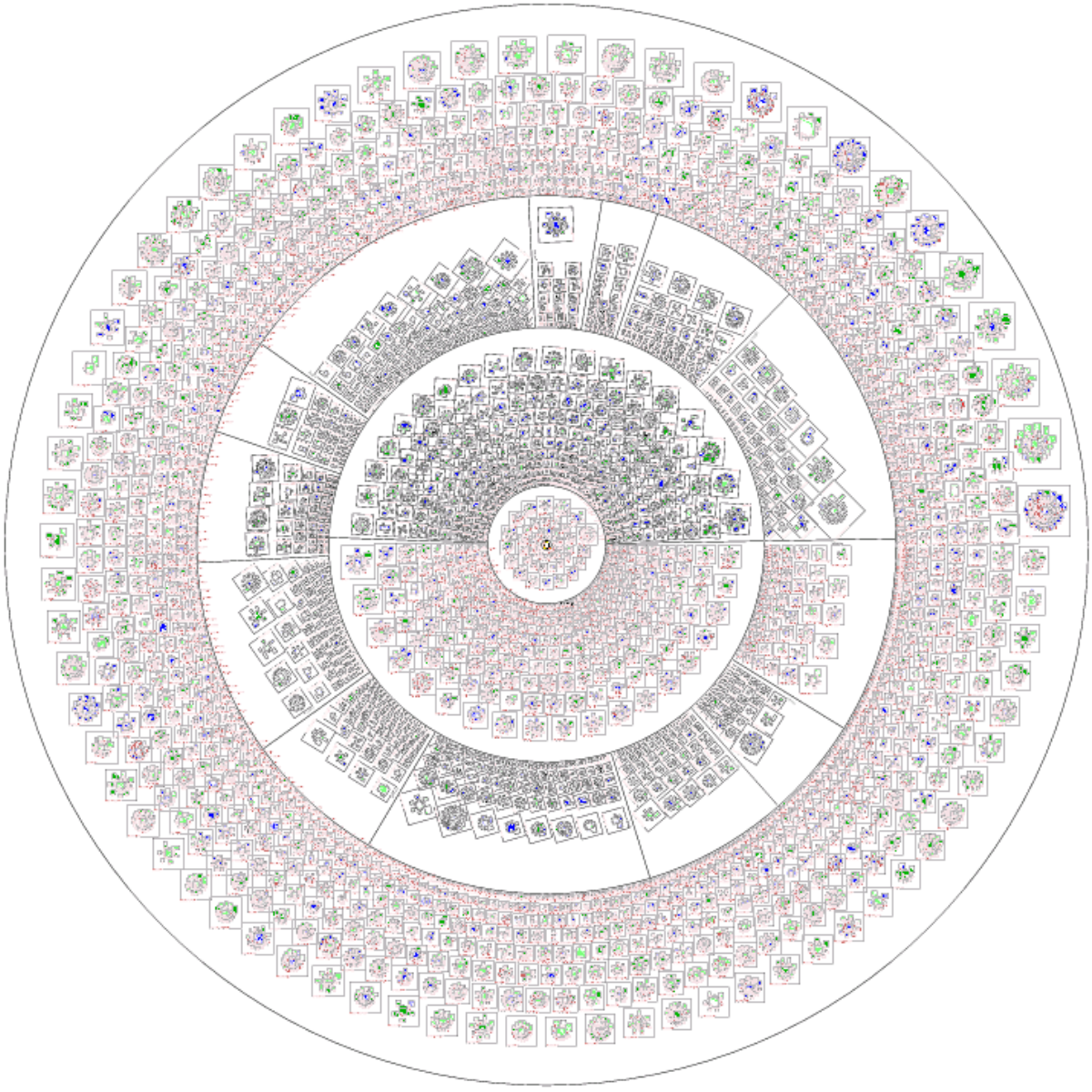
[图片来源: infosec]

操作系统

- 现代操作系统已经超过千万行代码



Linux Kernel v2.4.0 调用图



Linux Kernel 调用图放大

系统特征

- 系统是复杂的
  - 不同组件相互交互
  - 每一个组件自身也很复杂
- 系统设计是多目标的
  - 功能
  - 性能
  - 规模
  - 安全
  - 可靠性
  - 成本
  - …

系统特征

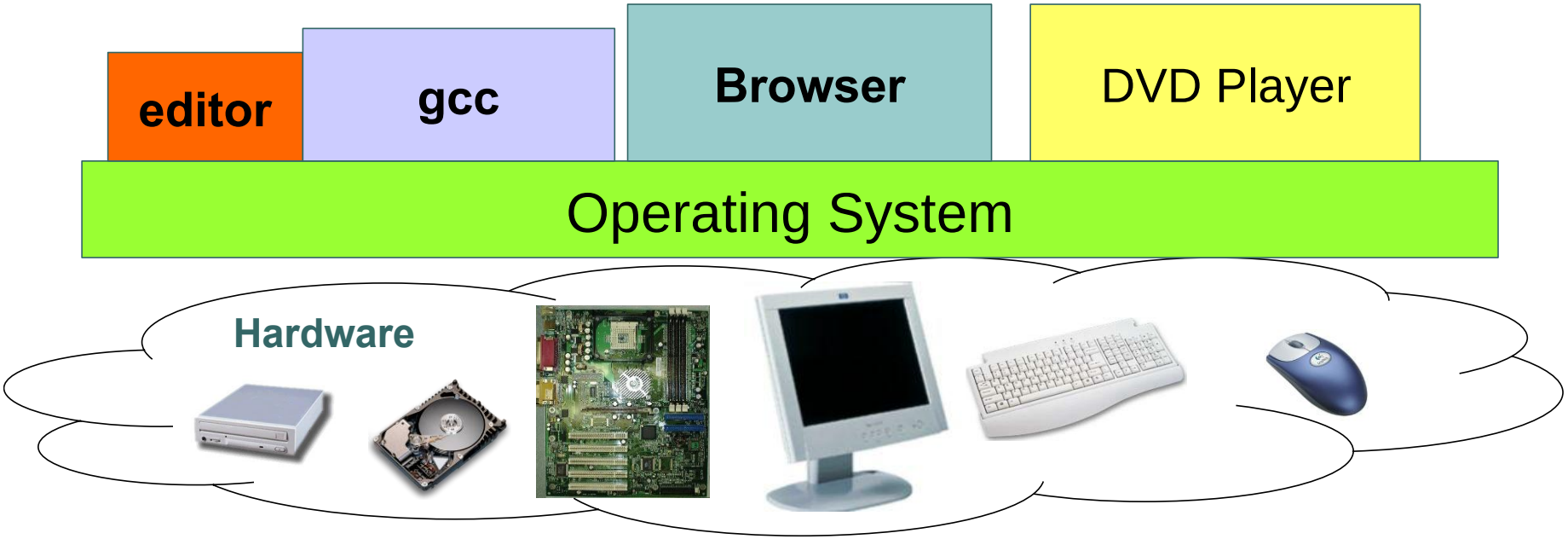
- 系统工作 vs. 理论工作
  - 理论工作：寻找极限（lower bound, upper bound）
    - 数学“孪生素数猜想”（张益唐，2013）：是否存在孪生素数间最大间隔的常数
    - 物理学“标准模型”：描述强力、弱力及电磁力这三种基本力及组成所有物质的基本粒子的理论
  - 系统工作：追求均衡（trade-off）
    - 缓存：空间和时间的trade-off
    - CAP理论：副本一致性、服务可用性和网络分区容忍性的trade-off
  - 在实际实现系统前，明确你的系统设计目标

内容提要

- 课程信息
- 什么是系统？
- 什么是操作系统？
- 怎么学习操作系统？

什么是操作系统？

- 承上启下
  - 在应用和硬件之间的一层软件
  - 对上层软件提供硬件抽象、实现共用功能
  - 对底层硬件进行管理、实现共享、保证隔离



操作系统能做什么？

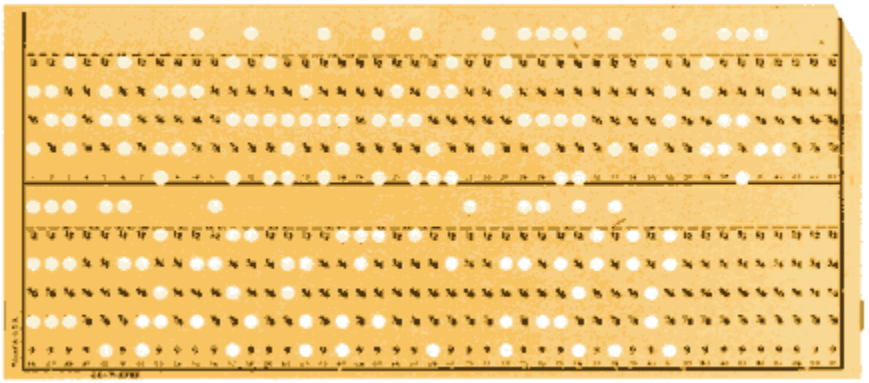
- 一个程序的运行

```
#include <stdio.h>

int main(int argc, char *argv[]) {
   int num = 0, base = 3;
   int result = 0;

   printf("Please input a number: ");
   scanf("%d", &num);

   result = base + num;
   printf("Hello %d\n", result);

   return 0;
}
```

操作系统能做什么？

- 抽象
  - 让硬件可用
    - 接收输入: scanf
    - 计算：使用CPU
    - 产生输出: printf
  - 让软件好写
    - 不用每个程序自己编写输入、输出函数

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



操作系统能做什么？

- 多个程序的运行

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
   char *str = argv[1];

   for(int i=0;i<3;i++) {
      printf("Hello %s\n", str);
      sleep(1);
   }

   return 0;
}
```

```
./a.out U & ./a.out C & ./a.out A
& ./a.out S &

Hello U
Hello C
Hello A
Hello S
Hello U
Hello S
Hello A
Hello C
Hello A
Hello C
Hello S
Hello U
```

操作系统能做什么？

- CPU资源共享与管理
  - 一个CPU core，多个程序同时运行
    - 让多个程序共享使用一个CPU core
    - 分配资源，调度程序执行
    - 有程序需要长时间使用CPU，是否允许？
    - 有程序需要优先使用CPU，是否允许？
    - ...
  - 多个CPU core，多个程序同时运行
    - 分配资源，调度程序执行
    - 是否允许一个程序独占一个CPU core？
    - …

操作系统能做什么？

- 一个程序的运行

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int *array = malloc(5*sizeof(int));
    printf("The starting address of the in-memory array is %p\n", array);

    array[0] = 1;
    for(int i=1;i<5;i++) {
      array[i] = array[i-1] + i;
      printf("The %d element is %d\n", i, array[i]);
    }

    return 0;
}
```

操作系统能做什么？

- 一个程序的运行

./a.out & ./a.out &

The starting address of the in-memory array is 0x7fc83c4027d0
The starting address of the in-memory array is 0x7f87924027d0

The 1 element is 2
The 2 element is 4
The 3 element is 7
The 1 element is 2
The 4 element is 11
The 2 element is 4
The 3 element is 7
The 4 element is 11

操作系统能做什么？

- 内存资源共享、管理与隔离
  - 给程序分配内存资源
  - 让程序读写内存
  - 程序结束后回收内存资源
  - 不同程序间共享使用内存
    - 是否允许不同程序访问同一个内存地址？
    - 程序A能否访问程序B的内存数据？
    - 内存不够用时，如何处理？
    - …

操作系统能做什么？

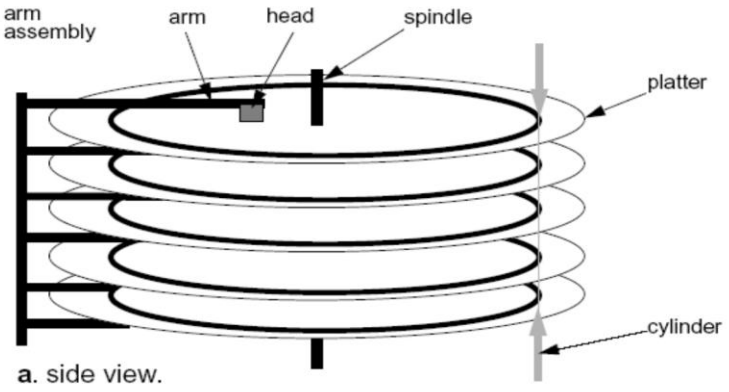
- 一个程序的运行

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

int main(int argc, char *argv[]) {
   int fd = open("/tmp/examplefile",
O_CREAT|O_WRONLY, S_IRWXU);

   int rc = write(fd, "hello world\n", 13);

   close(fd);
   return 0;
}
```

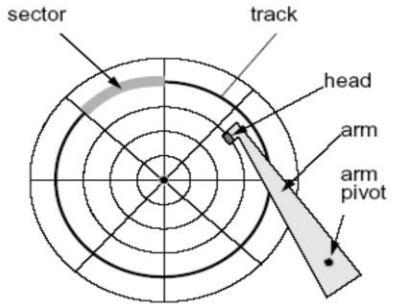
```
./a.out


cat /tmp/examplefile
hello world
```

操作系统能做什么？

- 数据持久化与管理
  - 让程序能在持久化存储设备上读写数据
  - 数据隐私保护、数据可靠性保护
  - 和不同的存储设备交互，驱动管理
  - 屏蔽存储设备硬件细节，同时管理存储设备



a. side view.



b. top view.

操作系统能做什么？

- 一个程序的运行

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int counter = 0;
int loops;

void *worker(void *arg) {
   for (int i=0;i<loops;i++) {
      counter++;
   }
   return NULL;
}
```

```
int main(int argc, char *argv[]) {
   loops = atoi(argv[1]);
   printf("Initial value: %d\n", counter);

   pthread_t p1, p2;

   pthread_create(&p1, NULL, worker,NULL);
   pthread_create(&p2, NULL, worker,NULL);

   pthread_join(p1, NULL);
   pthread_join(p2, NULL);

   printf("Final value: %d\n", counter);

   return 0;
}
```

操作系统能做什么？

- 并发访问管理
  - 控制多个程序访问相同资源时的正确性
    - 能否把程序访问变成串行？
    - 如何保证程序操作的原子性？
    - 如何让其他程序知道当前程序操作已完成？

./a.out 1000
Initial value: 0
Final value: 2000

./a.out 10000
Initial value: 0
Final value: 20000

./a.out 100000
Initial value: 0
Final value: 115664

操作系统能做什么？

- 还有什么？
  - 数据如何传输给远程计算机
  - 一个程序崩溃不能引起整个系统崩溃
  - 虚拟机、容器
  - 安全保护

什么是操作系统？

- 操作系统的发展（mainframe，大型机）
  - IBM 701计算机 1954
    - OS是一个通用函数库
    - 用户自行提交程序
    - Each user was allocated a minimum 15-minute slot, of which time he usually spent 10 minutes in setting up the equipment to do his computation . . . By the time he got his calculation going, he may have had only 5 minutes or less of actual computation completed-wasting two thirds of his time slot.

      [Hansen, The Evolution of Operating Systems, 2000]
  - IBM 709计算机 1959
    - 支持批处理

什么是操作系统？

- 操作系统的发展（minicomputer，小型机）
  - 1960s
    - 支持MultiProgramming，多个程序同时运行
    - CPU、内存、存储资源共享与隔离
  - Multics 1965、UNIX 1974
    - Time-sharing Operating System

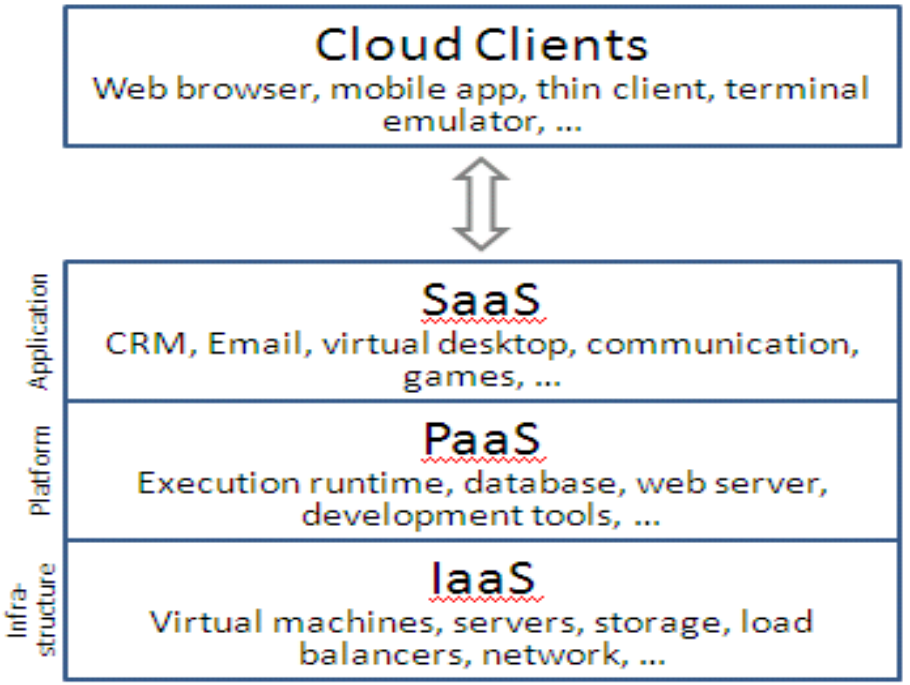
什么是操作系统？

- 操作系统的发展（personal computer，个人计算机）
  - Alto operating system
    - 64K内存，2.5MB磁盘
    - 图形显示
    - 以太网
    - 健壮的文件系统
    - 顺序执行的单用户系统
  - DOS
  - Windows
  - Mac OS
  - Linux

第一台PC Alto(Xerox PARC)

什么是操作系统

- 集群
  - 网络连接的计算机
  - 商用操作系统+分布式框架（如MapReduce，Spark）
- 云计算
  - 数据中心是一个计算机
  - 按需使用资源
  - 数据中心操作系统
    - Mesos
    - 阿里飞天系统



什么是操作系统

- 物联网
  - 万亿规模的物端设备
  - 物端应用需求差异化：“昆虫纲现象”
  - 嵌入式OS
    - RT-Thread
  - 手机OS
    - iOS, Android, 鸿蒙
  - 物端操作系统？

内容提要

- 课程信息
- 什么是系统？
- 什么是操作系统？
- 怎么学习操作系统？

学习目标

- 了解计算机操作系统发展
- 了解操作系统的基本概念与主要模块（What）
  - OS是计算机的核心部分
  - 很多研究领域的基础：网络、分布式系统、数据库等
- 熟悉OS设计原理与关键机制（How）
  - 理解复杂系统如何工作
  - 成为一个懂系统的同学
- 培养系统观（Why）
  - OS的设计思想可以用于其他系统或领域

怎么学习操作系统？

- “我曾经给本科生教操作系统课，几个星期下来，有的学生就对系统方向的研究上瘾了，觉得特有意思，而另一批学生就特别反感，认为系统过于复杂，需要考虑的东西太多。”

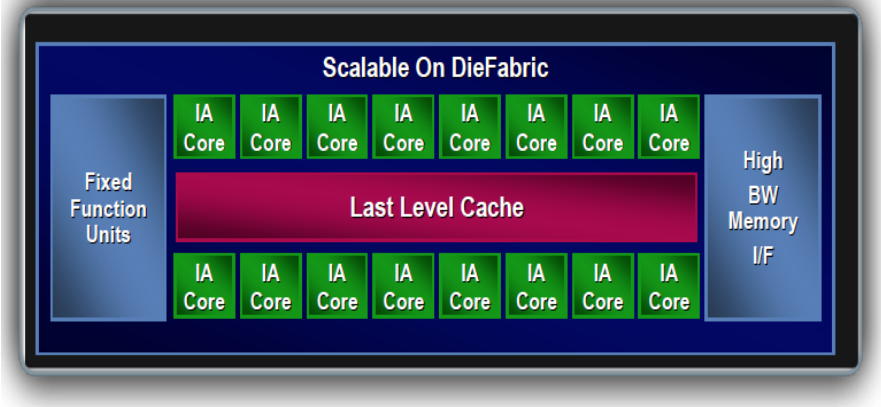
周源源教授 UCSD

怎么学习操作系统？

- 听课、阅读教材、做习题
- 阅读代码
  - 阅读并理解经典操作系统代码：结构、流程、技巧
  - 阅读你感兴趣的系统代码
- 开发代码
  - 实现一个小的操作系统（操作系统研讨课）
  - 自行开发一些程序
    - 多线程、缓存、调度算法
    - 设备驱动
- “理论联系实际”

怎么学习操作系统？

- 操作系统仍然在变化
  - 底层硬件不断在变化
  - 上层应用需求不断在变化
    - 云计算、大数据、AI、边缘计算



多核、众核

持久化内存

怎么学习操作系统？

- 操作系统仍然在变化
  - 操作系统的结构也随之相应变化
  - 没有严格的界限
    - 应用、库函数、内核模块
    - 系统调用、设备驱动、智能硬件
    - 用户态、核心态
    - 本地资源、远程资源
  - 设计原则没有变
    - 没有什么算法和结构是普适的，特定的场景下做出合适的选择
    - Trade-off

课程计划

- 讲授内容：
  - 操作系统结构
  - 处理器：进程和线程
  - 并发：同步、通信
  - 内存：虚存管理
  - 设备：中断与驱动
  - 存储：文件系统
- 实例分析：
  - XV6 实例分析（报告和讨论）
  - 主流操作系统实例