# 虚拟化 & 并发：复习

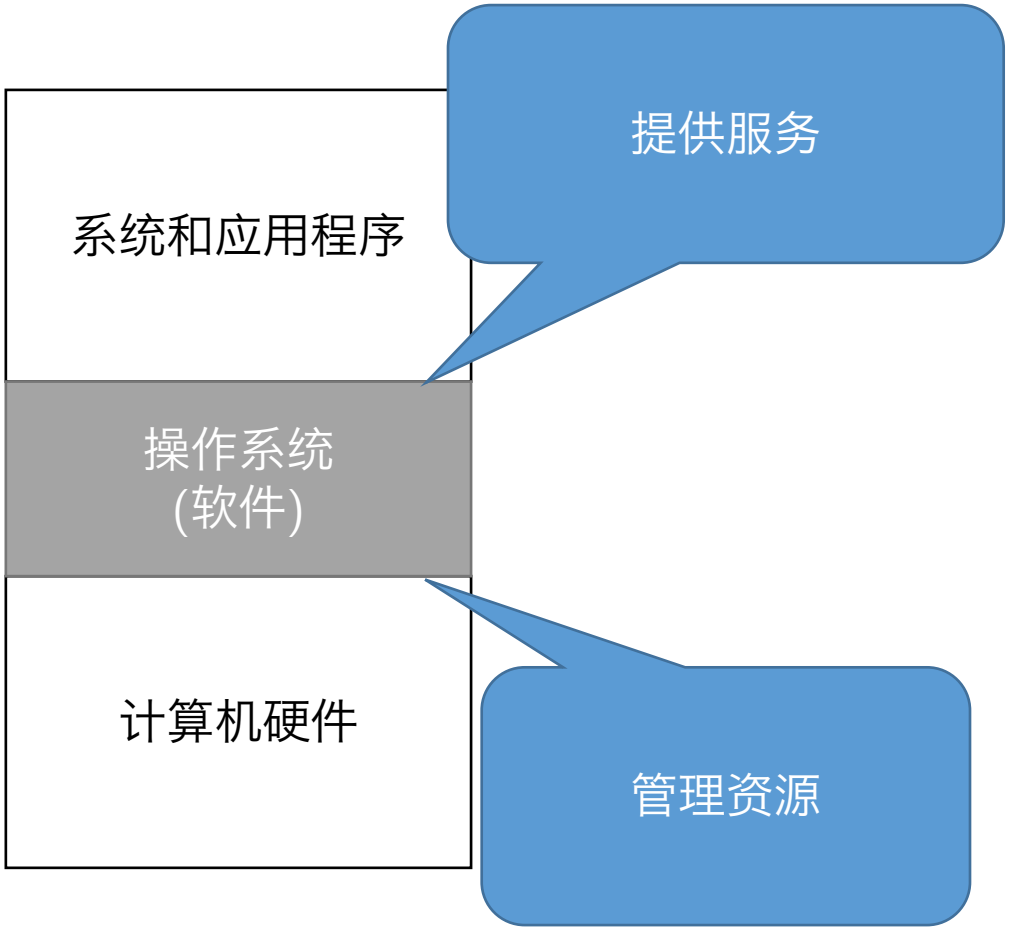
蒋炎岩

南京大学 | 计算机软件研究所 | 系统与软件分析研究组


# 本次复习课

- 复习前半学期的重要内容
- 操作系统相关面试题实战
  - 感谢助教(曹阳 <cycpp0002@163.com>)整理了一些面试中常见的操作系统问题
- 一些facts
  - 课程并没有覆盖其中所有的内容(和概念)
  - 但当你接近操作系统的真相，这些内容和概念都能被推导出来

# 操作系统概述



# 什么是计算机硬件？什么是程序？

- 硬件的例子
  - x86, MIPS, ARM, ...
- 程序的例子
  - 编译、链接后的代码(ELF文件)
  - busybox, Xorg, firefox, ...
- Three Easy Pieces
  - 虚拟化、并发、持久化(后半学习开始)

# 面试题(概念)

- 请简述你常用的Linux命令(真有人被这么问了，还问懵了)
  - 文件/目录管理：ls, cd, mkdir, mkfifo, chmod, chown, find, ...
  - 文本处理：cat, cut, head, tail, grep, sed, awk, ex(vi), ...
  - 进程/系统管理：ps, kill, top, vmstat, df, du, lsof, tcpdump, netstat, iptables, ...
  - 编程：make, gcc, ctags, gdb, strace, perf, stap, ...
  - 其他：xargs, parallel, python, ...

- 体现一件趣事：有没有想“再完美一点”？还是就用自己旧的方式解决一切问题?
  - 说出的命令体现了个人的追求

# 虚拟化(进程抽象)

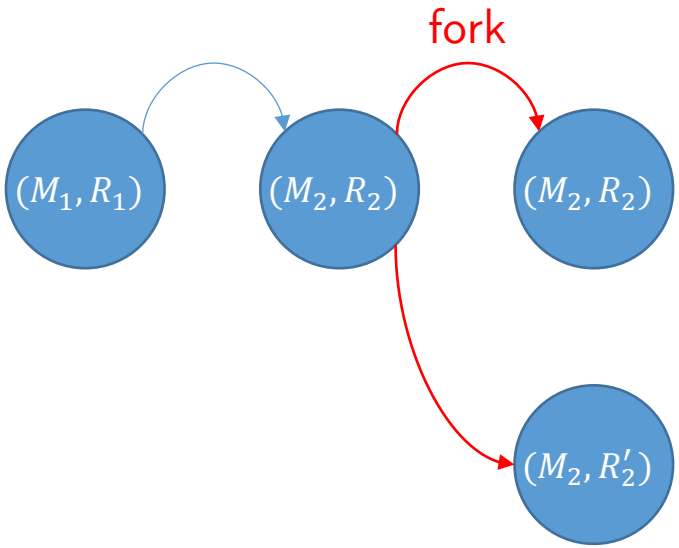
# 进程抽象

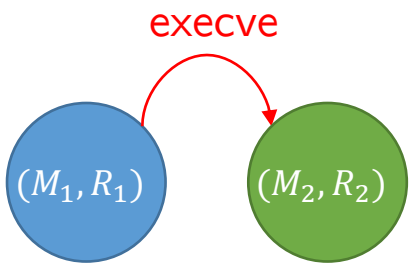
- 运行的程序
  - C, Java, Python, ...程序从操作系统眼里看都是一样的
  - 在内存/寄存器的状态机上执行确定行为的状态转移
  - 通过系统调用访问操作系统资源

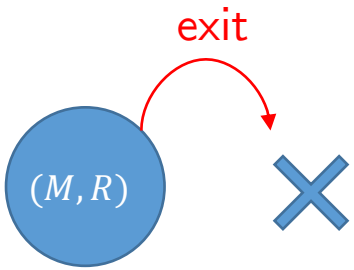
- 顺着这个主线理解进程
  - 状态机里有什么?
  - 单独的状态机 → 进程完全独立 → 操作系统管理一切资源
    - 资源有什么? CPU、内存、I/O设备 (everything is a file).....

# UNIX进程API

- fork, execve, exit
  - 创建/替换/终止进程
  - 还有其他一些API (kill, waitpid, …)



# 面试题(虚拟化)

- 简述进程的“状态转换模型”
  - TASK_RUNNING
  - TASK_INTERRUPTIBLE // TF**kM: read可以被interrupt哦!
  - TASK_KILLABLE // 介于上下二者之间，但除了面试Linux工程师，面试题不会考那么细
  - TASK_UNINTERRUPTIBLE
  - TASK_STOPPED
  - TASK_ZOMBIE
  - TASK_TRACED // 如果你知道strace，这就不难理解了
- 系统调用和函数调用的区别？何时处于何种状态？

# 面试题(虚拟化)

- 简述信号“作用”？
- 简述信号“产生方式”？
- 简述信号“生命周期”？
- 简述信号“处理方式”？

- 课堂上没有详细讲过
  - 但演示过kill; 使用过waitpid; 我们用Ctrl-C终止程序...
  - 这些都是通过信号实现的

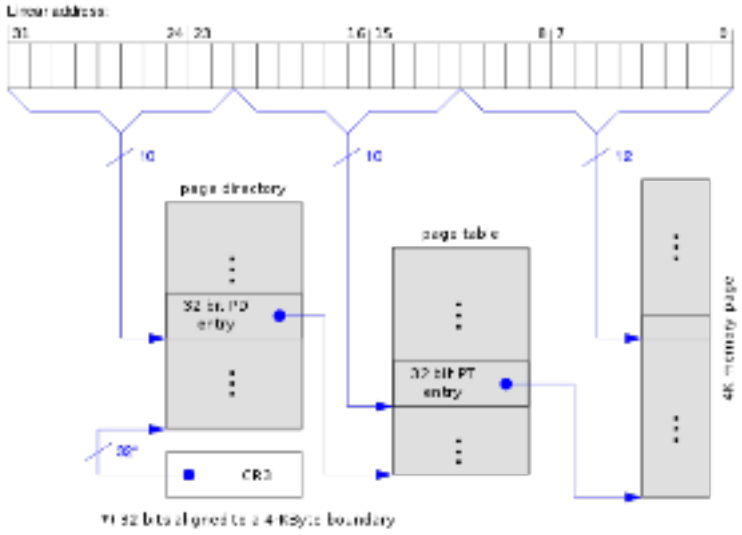
# 面试题(虚拟化)

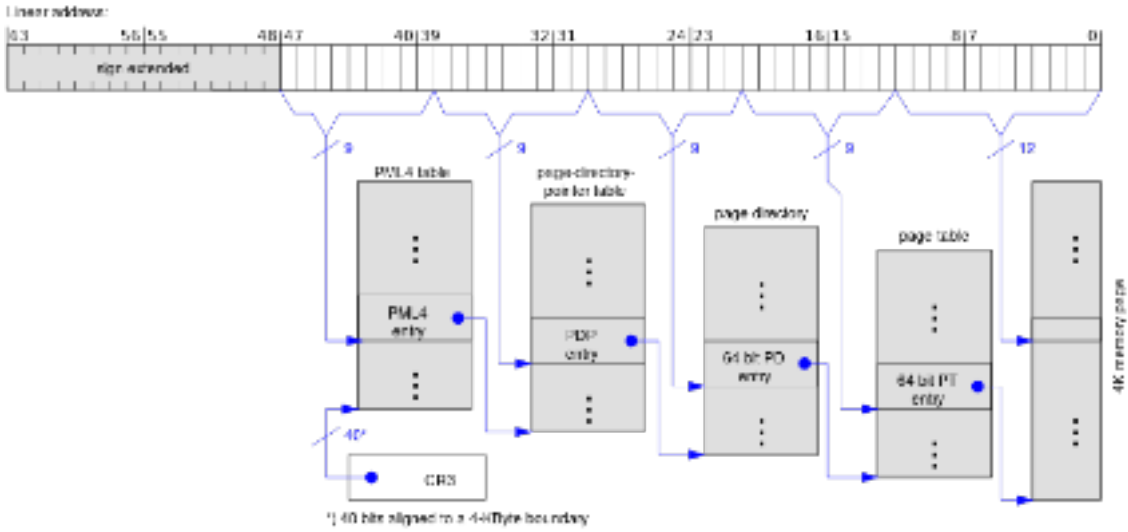
- 简述什么是“用户态”/“核心态”？
- 为何操作系统需要分出两者?
- “用户态”至“核心态”的切换方式?

# 虚拟化(虚存抽象)

# 硬件提供了虚拟存储管理机制

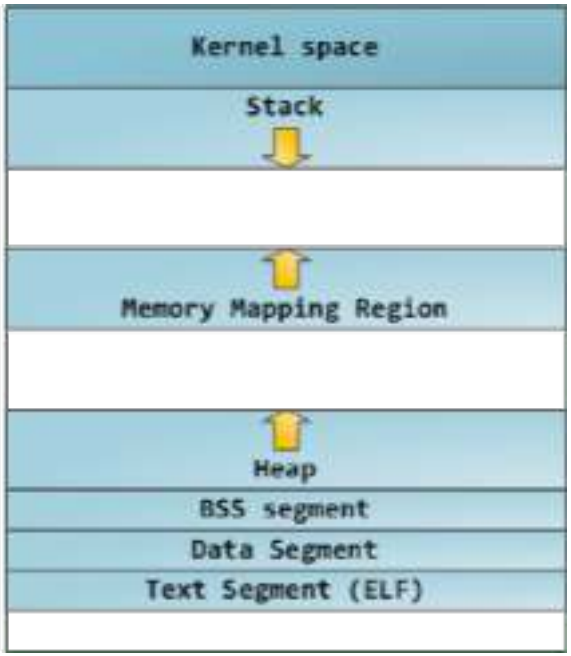
- 这样不用给每个进程分配物理内存，而只需要分配一个映射$VM(x)$，把虚拟地址映射到物理地址



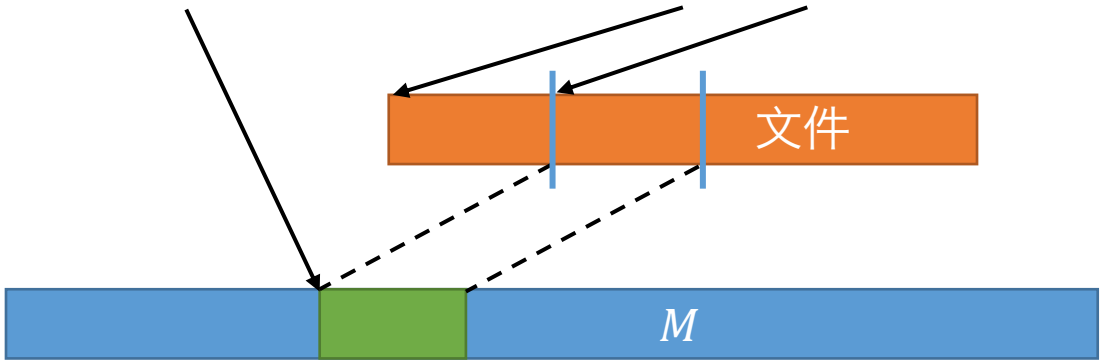



# 进程的地址空间管理

- 通过虚拟存储管理机制实现
  - 问题：如何查看进程地址空间里有什么？




- mmap(addr, length, prot, flags, fd, offset)



# 面试题(虚拟化)

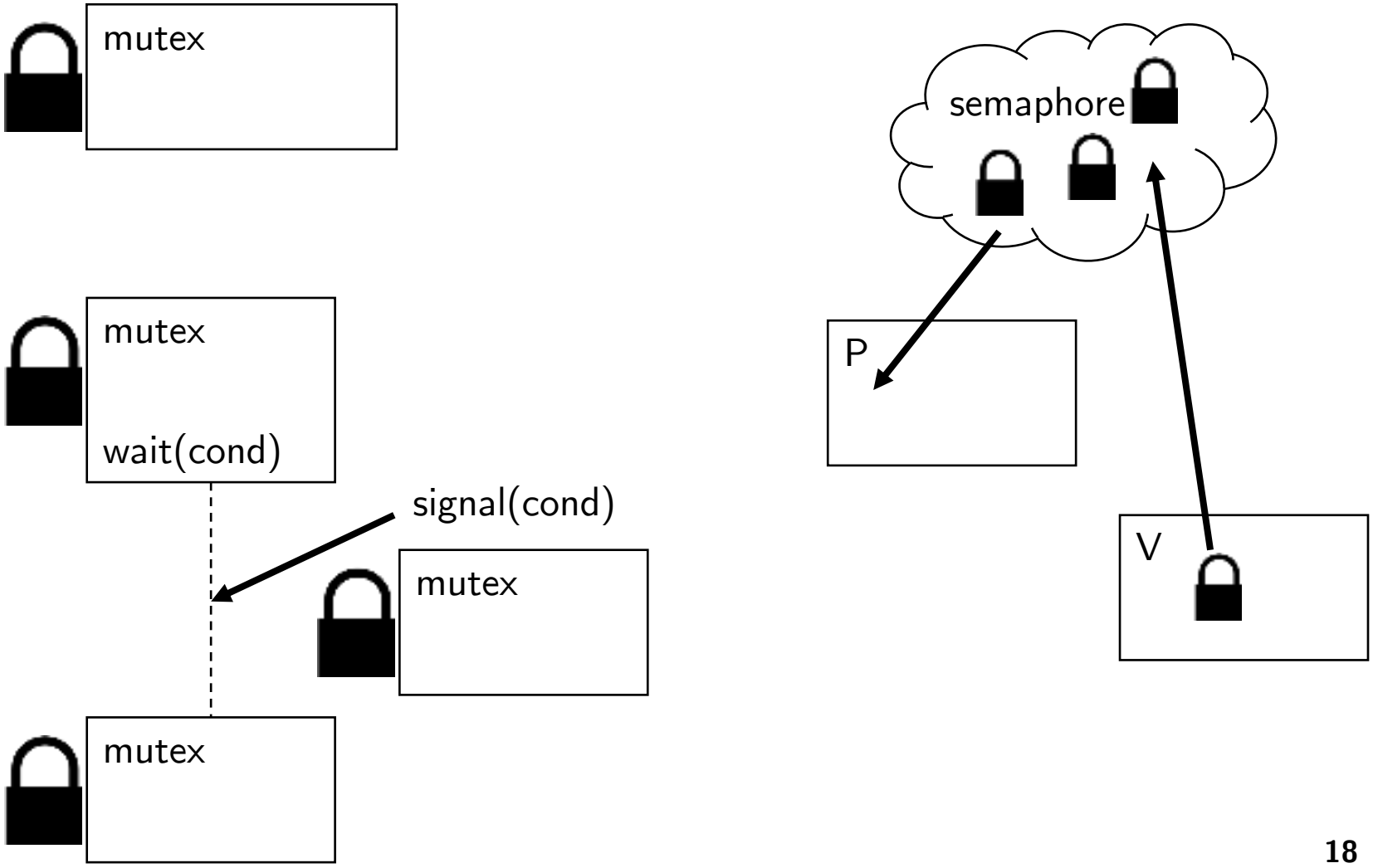
- malloc 申请动态内存之后，free 释放之时，操作系统会立即收回那块内存吗？为什么？

# 并发

# 面试题(虚拟化/并发)

- 进程和线程的区别？同一进程内，哪些资源是线程独有、哪些资源是线程共享？

# 复习：互斥、CV、信号量



# 生产者/消费者问题

```
void produce() {
  lock(&mutex);
  while (count == N)
    wait(&empty, &mutex);
  printf("("); count++;
  signal(&fill);
  unlock(&mutex);
}

void consume() {
  lock(&mutex);
  while (count == 0)
    wait(&fill, &mutex);
  printf(")"); count--;
  signal(&empty);
  unlock(&mutex);
}
```

```
sem_t empty = SEM_INIT(N);
sem_t fill = SEM_INIT(0);

void produce() {
  P(&empty);
  printf("(");
  V(&fill);
}

void consume() {
  P(&fill);
  printf(")");
  V(&empty);
}
```

# 面试题(并发)

- 简述什么是进程间同步/异步？分别有什么应用场景？

# 面试题(并发)

- 同步和互斥有什么区别和联系?

- 常见的进程间通信/同步的方式?

# 面试题(并发)

- 什么是死锁？死锁产生的条件是什么？

# 复习建议

# 建议

- 题型：8个简答题
  - 把Mini Labs弄懂
  - fork/execve/exit必考
  - 虚拟存储必考
  - 同步和互斥(条件变量)必考

- 另外：弄清楚操作系统/程序的行为很重要
  - 系统调用的语义(例如fork/execve/exit)
  - 并发原语的语义
  - 并发程序执行各种可能的情况
  - ……