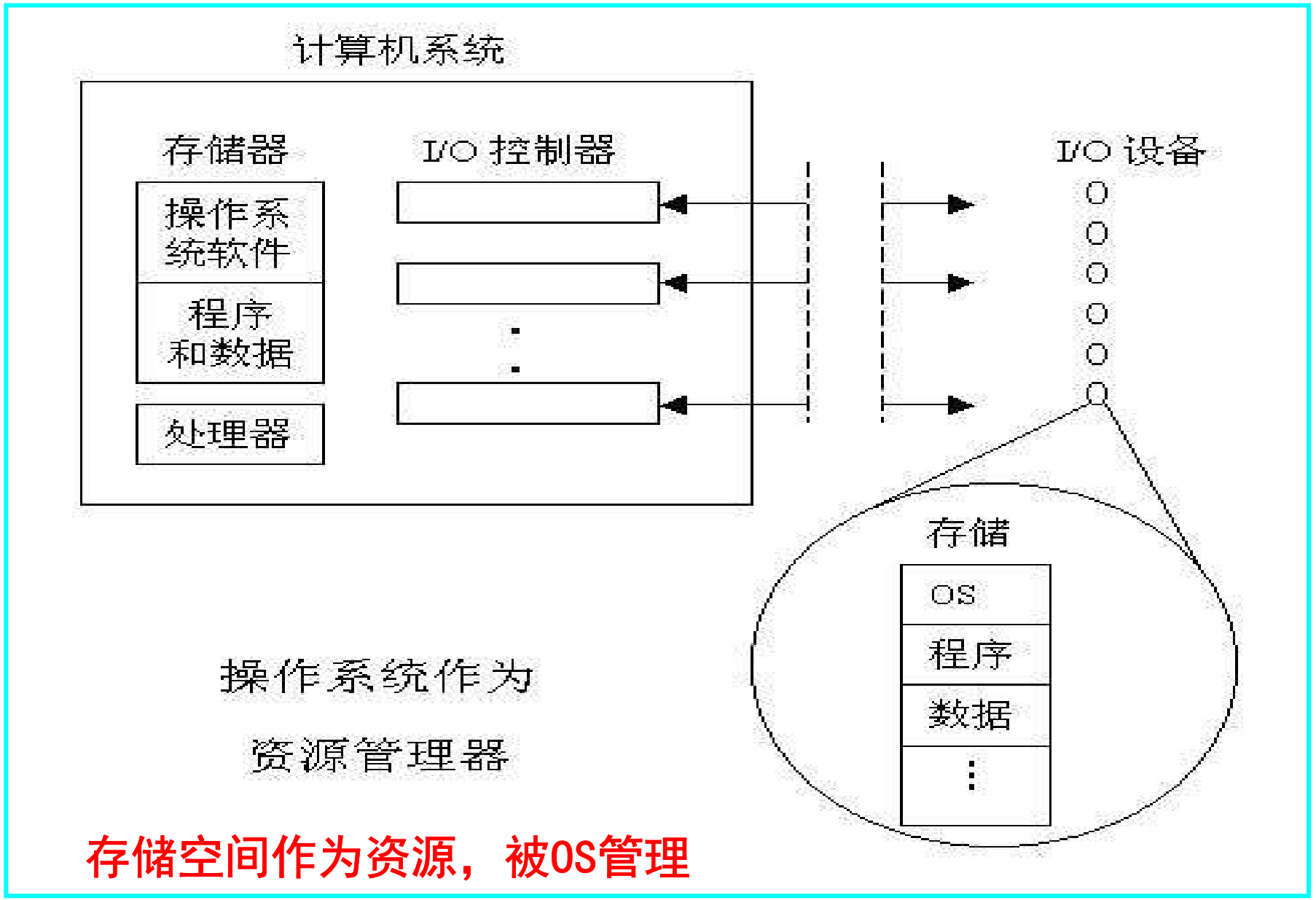
Lecture 14: Virtual Memory I

虚拟存储器

主 要 内 容

- 存储管理技术的发展过程
- 虚拟存储器的基本概念
  - 按需调页
  - 虚拟地址空间
- 虚拟存储器方式
  - 三种方式：页式、段式、段页式
  - 逻辑地址--物理地址的转换
  - 页表、缺页处理
- 替换策略
- 快表
- 存储保护
  - 地址越界检查
  - 存取权限检查

存储器资源的管理由操作系统来实现



存储空间作为资源，被OS管理

存储器资源的管理由操作系统来实现

- 操作系统(OS)通过合理地管理、调度计算机的硬件资源，使其高效被利用。
- 存储器作为一种空间资源，由OS来管理
- CPU执行的程序总是在操作系统和用户程序之间切换。
- 主存中同时存储OS和用户程序。磁盘中也存储OS和用户程序
- CPU执行指令时，涉及到存储器操作，因此，CPU中专门有一个存储器管理部件MMU （Memory Management Unit）协助OS完成存储器访问

OS为“进程” 分配存储器资源，所以，先了解一下进程的概念。

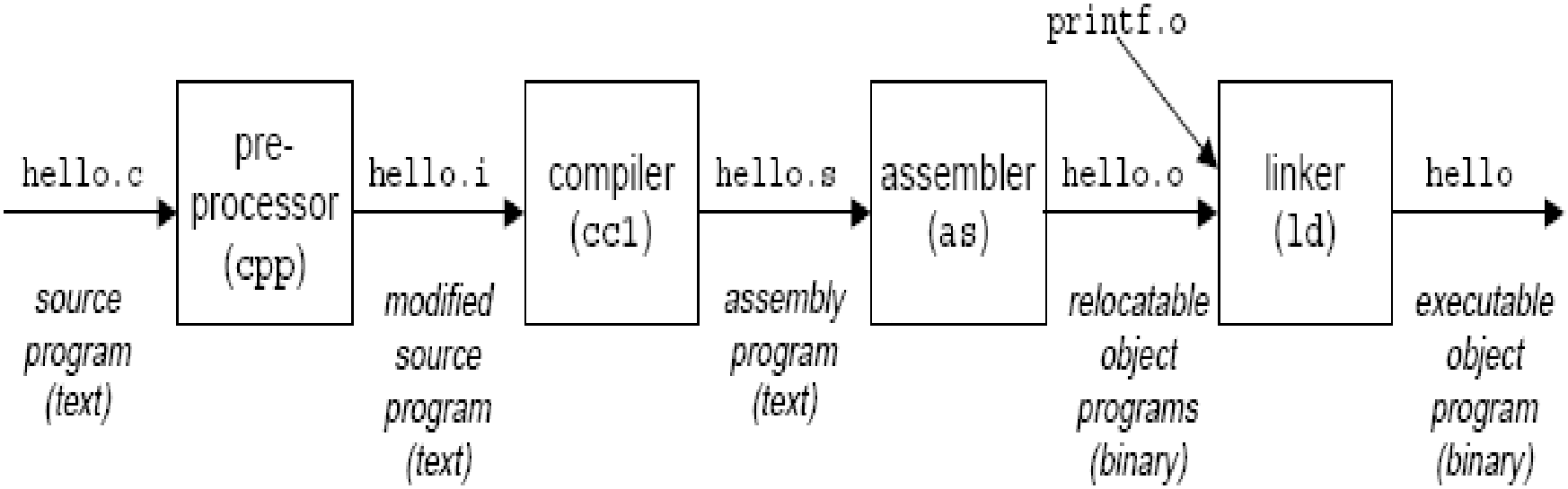
复习：一个典型程序的转换处理过程

经典的“ hello.c ”C-源程序

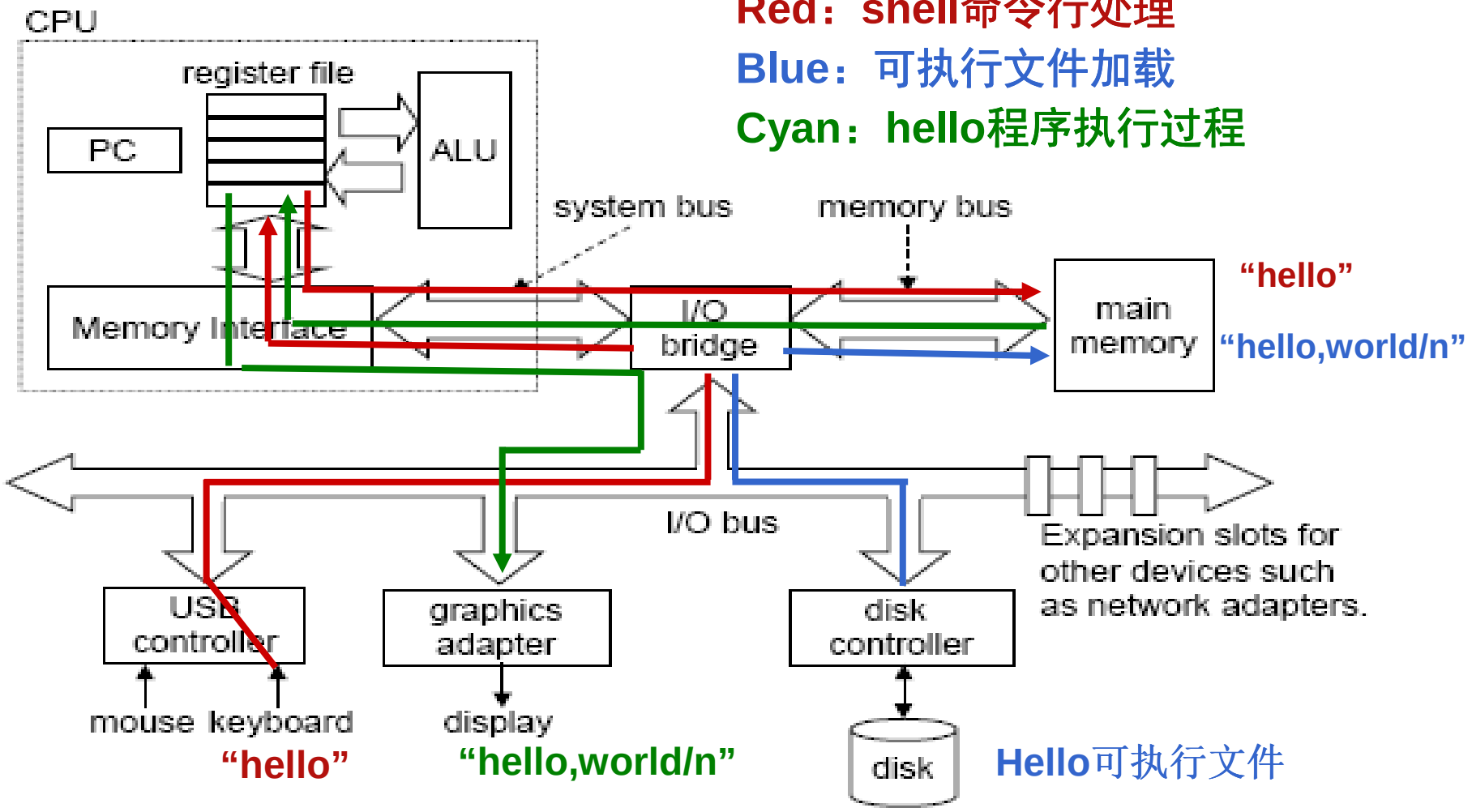
```
1 #include <stdio.h>
2
3 int main()
4 {
5 printf("hello, world\n");
6 }
```

hello.c的ASCII文本表示

```
# i n c l u d e <sp> < s t d i o .
35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
h > \n \n i n t <sp> m a i n ( ) \n {
104 62 10 10 105 110 116 32 109 97 105 110 40 41 10 123
\n <sp> <sp> <sp> <sp> p r i n t f ( " h e l
10 32 32 32 32 112 114 105 110 116 102 40 34 104 101 108
l o , <sp> w o r l d \ n " ) ; \n }
108 111 44 32 119 111 114 108 100 92 110 34 41 59 10 125
```



复习：Hello程序的数据流动过程



Red：shell命令行处理

Blue：可执行文件加载

Cyan：hello程序执行过程

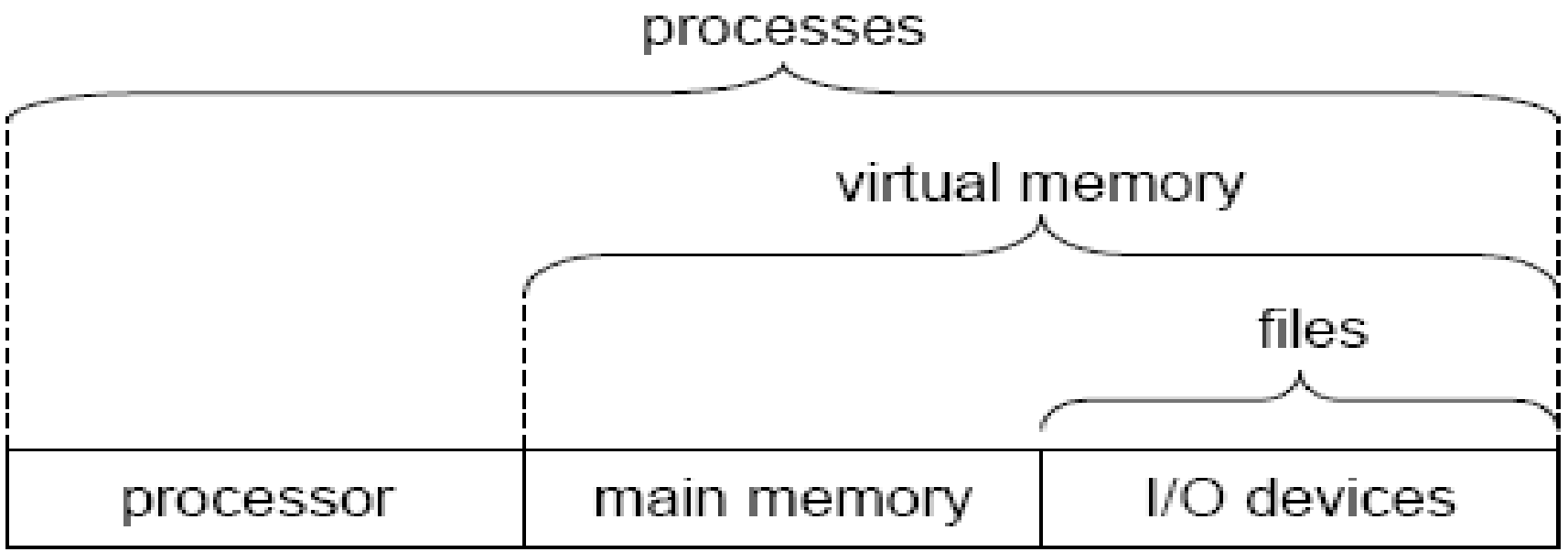
问题：hello程序何时被装？谁来装入？被谁启动？每次是否被装到相同的地方？Hello程序是否知道还有其他程序同时运行？是否能直接访问硬件资源？

操作系统在程序执行过程中的作用

- 在Hello程序执行过程中，Hello程序本身没有直接访问键盘、显示器、磁盘和主存储器这些硬件资源，而是依靠操作系统提供的服务来间接访问。 例如，调用printf（）函数访问硬件。
- 操作系统是在应用程序和硬件之间插入的一个中间软件层。
- 操作系统的两个主要的作用：
  - 硬件资源管理，以达到以下两个目的：
    - 统筹安排和调度硬件资源，以防止硬件资源被用户程序滥用
    - 对于广泛使用的复杂低级设备，为用户程序提供一个简单一致的使用接口
  - 为用户使用系统提供一个操作接口

操作系统在程序执行过程中的作用

- 操作系统通过三个基本的抽象概念（进程、虚拟存储器、文件）实现硬件资源管理
  - 文件（files）是对I/O设备的抽象
  - 虚拟存储器（Virtual Memory）是对主存和磁盘I/O的抽象
  - 进程（processes）是对处理器、主存和I/O设备的抽象

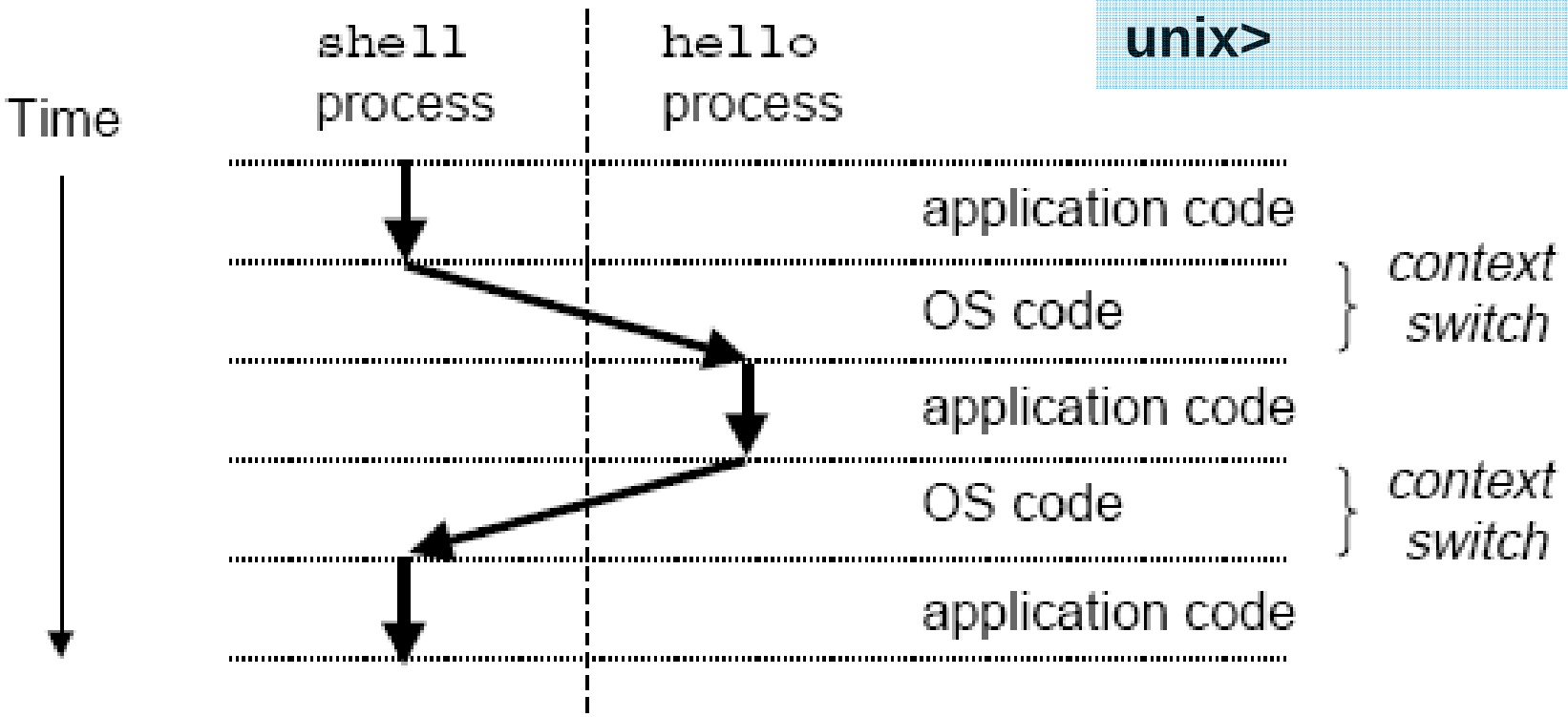


进程（processes）

- Hello程序运行时，Hello程序会以为（错觉）：
  - 所有系统资源都被自己独占使用
  - 处理器始终在执行本程序的一条条指令
- 进程是操作系统对运行程序的一种抽象
  - 一个系统可同时运行很多进程，但每个进程都好像自己独占使用系统
  - 实际上，操作系统让处理器交替执行很多进程中的指令
  - 操作系统实现交替指令执行的机制称为“上下文切换（context switching）”
- 进程的上下文
  - 指进程运行所需的所有状态信息，例如：PC、寄存器堆的当前值、cache内容、段/页表等
  - 系统中有一套状态单元，用以存放当前运行进程的上下文
- 上下文切换过程（任何时刻，系统中只有一个进程正在运行）
  - 上下文切换指把正在运行的进程换下，换一个新进程到处理器执行，上下文切换时，必须保存换下进程的上下文，恢复换上进程的上下文

进程（processes）

```
unix> ./hello [Enter]
hello, world
unix>
```



开始，Shell进程等待命令行输入
输入“Hello”后Shell进行系统调用
OS保存shell上下文，创建并换入hello进程
Hello进程中止时，进行系统调用
OS恢复shell进程上下文，并换入shell进程

由于在一个进程的整个生命周期中，有不同进程在CPU上交替运行，所以运行时间很难准确、重复测量

存储管理(Memory Management)

- 早期采用单道程序，系统的主存中包含：
  - 操作系统（常驻监控程序）
  - 正在执行的一个用户程序

  所以无需进行存储管理，即使有也很简单。
- 现在都采用多道程序，系统的存储器中包含：
  - 操作系统
  - 若干个用户程序

  如果在存储器中进程数很少，则由于进程花费很多时间来等待I/O，常使处理机处于空闲状态。因此，存储器需要进行合理分配，尽可能让更多进程进入存储器。
- 在多道程序系统中，存储器的“用户”部分须进一步划分以适应多个进程。划分的任务由OS动态执行，这被称为存储管理(memory management)。

Memory Management Schema

- 使系统中尽量多地存储用户程序的解决办法有：

  (1) 扩大主存(程序越来越长、主存贵，不是根本办法)

  (2) 采用交换(Exchange)方式和覆盖(Overlap)技术

  存储器中无处于就绪状态的进程（例如：某一时刻所有进程都在等待I/O）时，处理器将一些进程调出写回到磁盘，然后OS再调入其他进程执行，或新的作业直接覆盖老作业的存储区。

  分区(Partitioning)和分页(Paging)是交换的两种实现方式

  “交换”和“覆盖”技术的缺点：对程序员不透明、空间利用率差

  (3) 虚拟存储器(Virtual Memory)

  类似上述分页方式，但不是把所有页面一起调到主存，而是采用“按需调页Demand Paging”，在外存和主存间以固定页面进行调度。

  虚拟存储器方式下，引入了虚拟地址空间的概念。

  SKIP