Technology Tarena 达内科技

Unix系统高级编程

创建子进程 Unit14

创建子进程

fork

fork

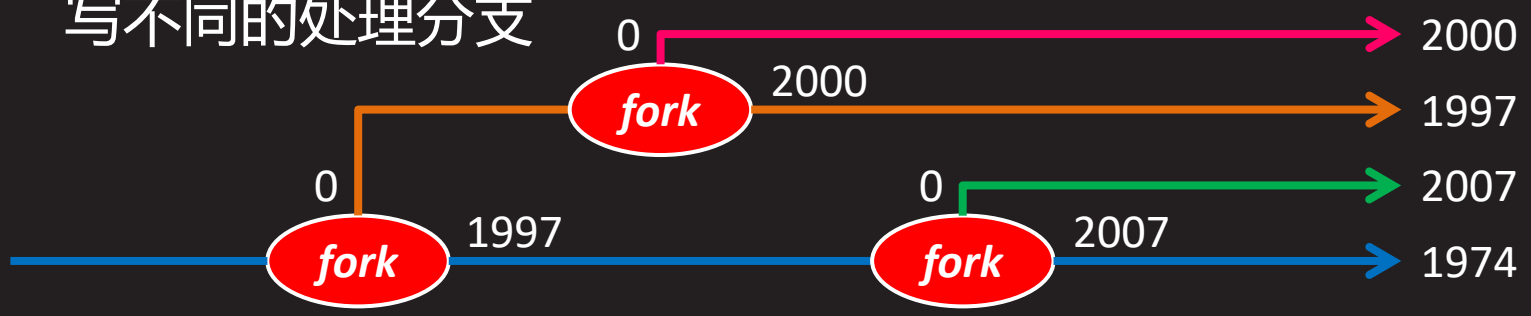
- 创建子进程

```
#include <unistd.h>

pid_t fork (void);
```

成功分别在父子进程中返回子进程的PID和0，失败返回-1

- 调用一次返回两次
  - 在父进程中返回所创建子进程的PID，而在子进程中返回0
  - 函数的调用者可以根据返回值的不同，分别为父子进程编写不同的处理分支



知识讲解

创建子进程

【参见：fork.c】

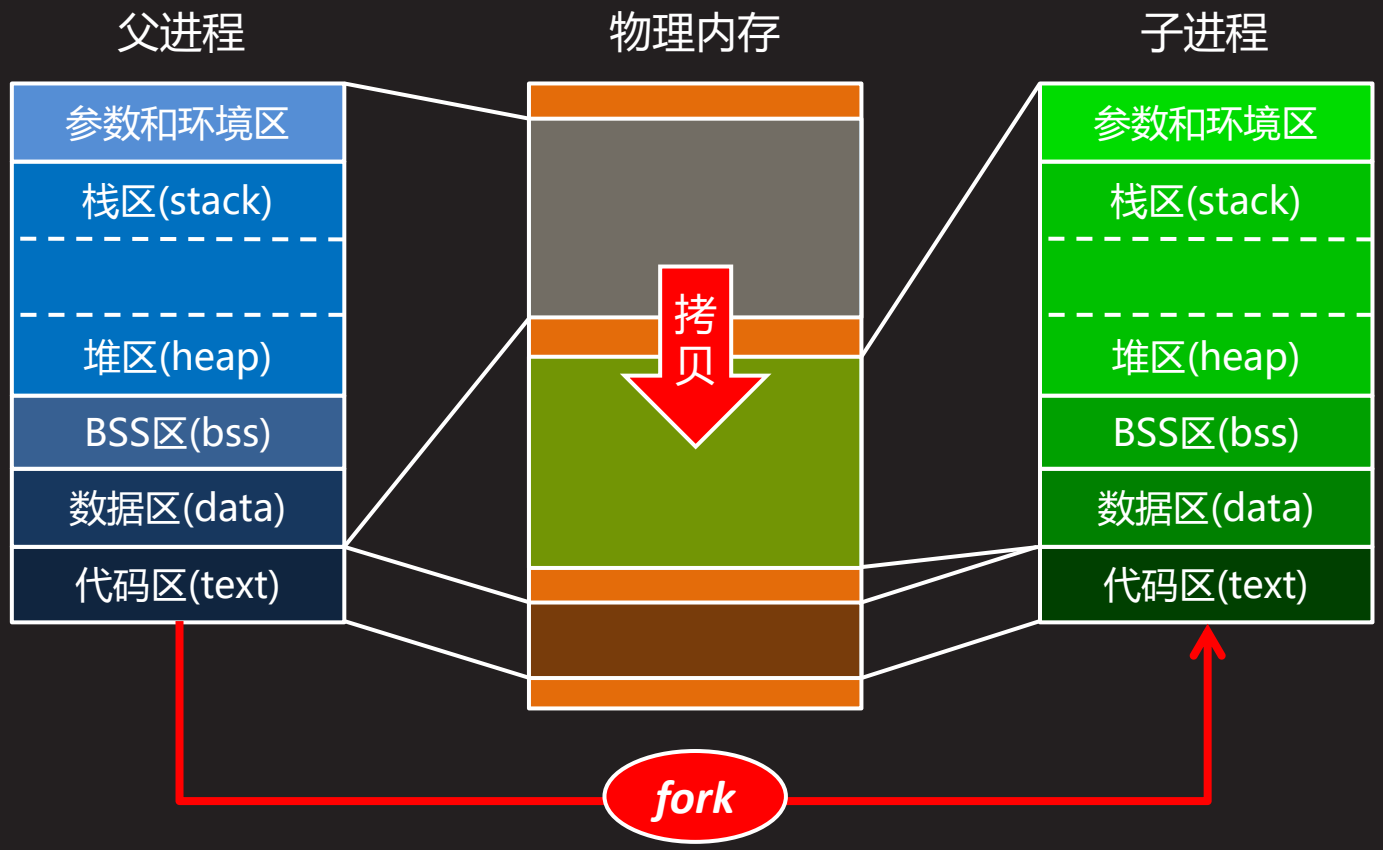
课堂练习

- 创建子进程

fork（续1）

- 子进程是父进程的不完全副本，子进程的数据区、BSS区、堆栈区(包括I/O流缓冲区)，甚至参数和环境区都从父进程拷贝，唯有代码区与父进程共享

知识讲解



子进程是父进程的副本

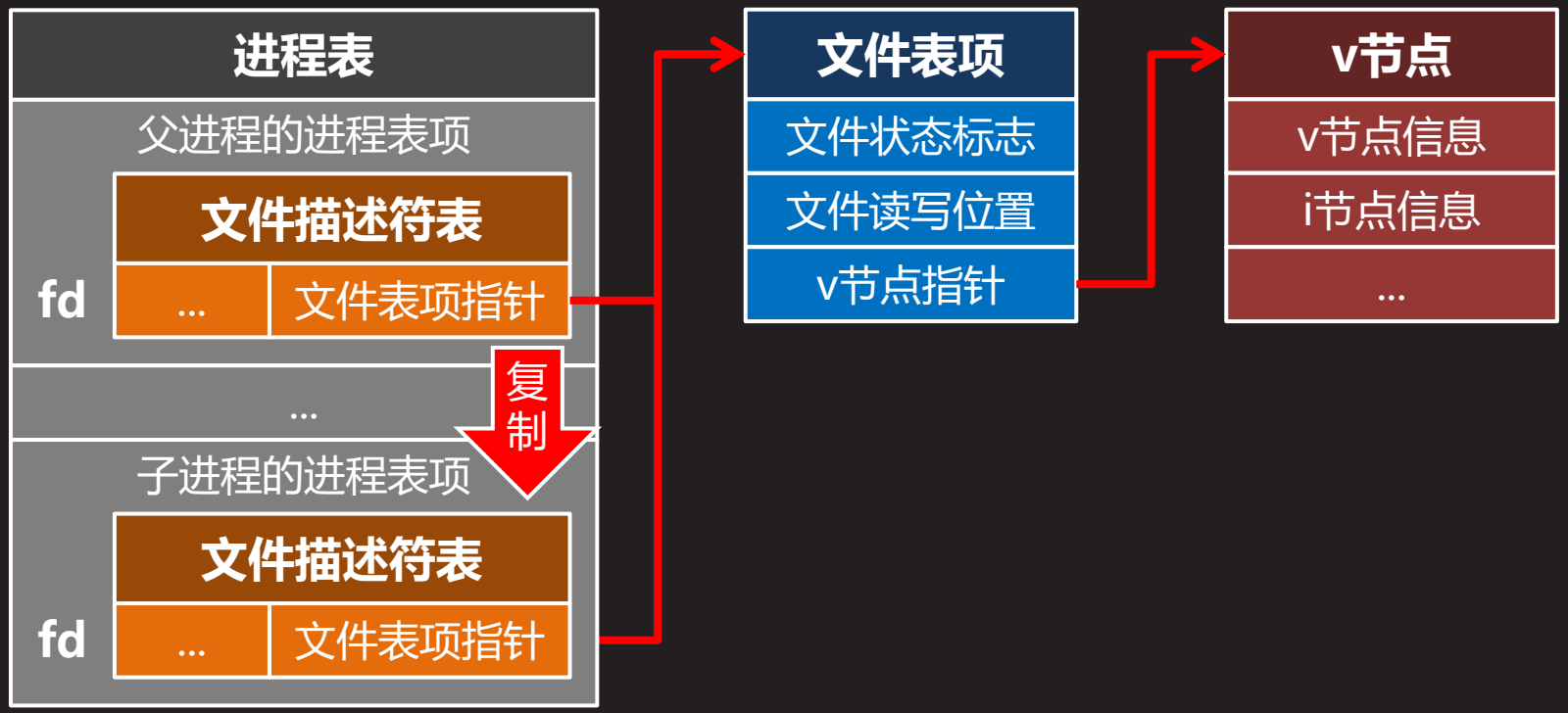
【参见：mem.c、is.c、os.c】

课堂练习

- 子进程是父进程的副本

fork（续2）

- fork函数成功返回以后，父子进程各自独立运行，其被调度的先后顺序并不确定，某些实现可以保证子进程先被调度
- fork函数成功返回以后，系统内核为父进程维护的文件描述符表也被复制到子进程的进程表项中，文件表项并不复制



知识讲解

父子进程共享文件表

【参见：ftab.c】

课堂练习

- 父子进程共享文件表

fork（续3）

- 系统总线程数达到上限(/proc/sys/kernel/threads-max)，或用户总进程数达到上限(ulimit -u)，fork函数将返回失败
- 一个进程如果希望创建自己的副本并执行同一份代码，或希望与另一个进程并发地运行，都可以使用fork函数
- 调用fork函数前的代码只有父进程执行，fork函数成功返回后的代码父子进程都会执行，受逻辑控制进入不同分支
  - pid_t pid = fork ();
    if (pid == -1) {
    perror ("fork"); exit (EXIT_FAILURE); }
    if (pid == 0) { 子进程执行的代码; }
    else { 父进程执行的代码; }
    父子进程都执行的代码;

知识讲解

孤儿与僵尸

【参见：orphan.c、zombie.c】

课堂练习

- 孤儿与僵尸

vfork

vfork

- 创建轻量级子进程

```
#include <unistd.h>

pid_t vfork (void);

        成功分别在父子进程中返回子进程的PID和0，失败返回-1
```

- vfork与fork函数的功能基本相同，只有以下两点区别
  - vfork函数创建的子进程不复制父进程的物理内存，也不拥有自己独立的内存映射，而是与父进程共享全部地址空间
  - vfork函数会在创建子进程的同时挂起其父进程，直到子进程终止，或通过exec函数启动了另一个可执行程序

知识讲解

vfork（续1）

- 终止vfork函数创建的子进程，不要使用return语句，也不要调用exit函数，而要调用_exit函数，以避免对其父进程造成不利影响
- vfork函数的典型用法就是在所创建的子进程里直接调用exec函数启动另外一个进程取代其自身，这比调用fork函数完成同样的工作要快得多
  - pid_t pid = vfork ();
    if (pid == -1) {
        perror ("vfork"); exit (EXIT_FAILURE); }
    if (pid == 0)
        if (execl ("ls", "ls", "-l", NULL) == -1) {
            perror ("execl"); _exit (EXIT_FAILURE); }

知识讲解

创建轻量级子进程

【参见：vfork.c】

课堂练习

- 创建轻量级子进程

vfork（续2）

- 写时复制(copy-on-write)
  - 传统意义上的fork系统调用，必须把父进程地址空间中的内容一页一页地复制到子进程的地址空间中(代码区除外)。这无疑是个十分漫长的过程(在系统内核看来)
  - 而多数情况下的子进程其实只是想读一读父进程的数据，并不想改变什么。更有甚者，可能连读一读都觉得多余，比如直接通过exec函数启动另一个进程的情况。漫长的内存复制在这里显得笨拙且毫无意义
  - 写时复制以惰性优化的方式避免了内存复制所带来的系统开销。在子进程创建伊始，并不复制父进程的物理内存，只复制它的内存映射表即可，父子进程共享同一个地址空间，直到子进程需要写这些数据时，再复制内存亦不为迟

vfork（续3）

- 写时复制(copy-on-write)
  - 写时复制带来的好处是，子进程什么时候写就什么时候复制，写几页就复制几页，没有写的就不复制。惰性优化算法的核心思想就是尽一切可能将代价高昂的操作，推迟到非做不可的时候再做，而且最好局限在尽可能小的范围里
  - 现代版本的fork函数已经广泛采用了写时复制技术，从这个意义上讲，vfork函数的存在纯粹只是一个历史遗留的产物，尽管它的速度还是比fork要快一点(连内存映射表都不复制)，但它的地位已远不如写时复制技术被应用到fork函数的实现中以前那么重要了

知识讲解

Technology Tarena 达内科技

总结和答疑