UC Day09

预习课

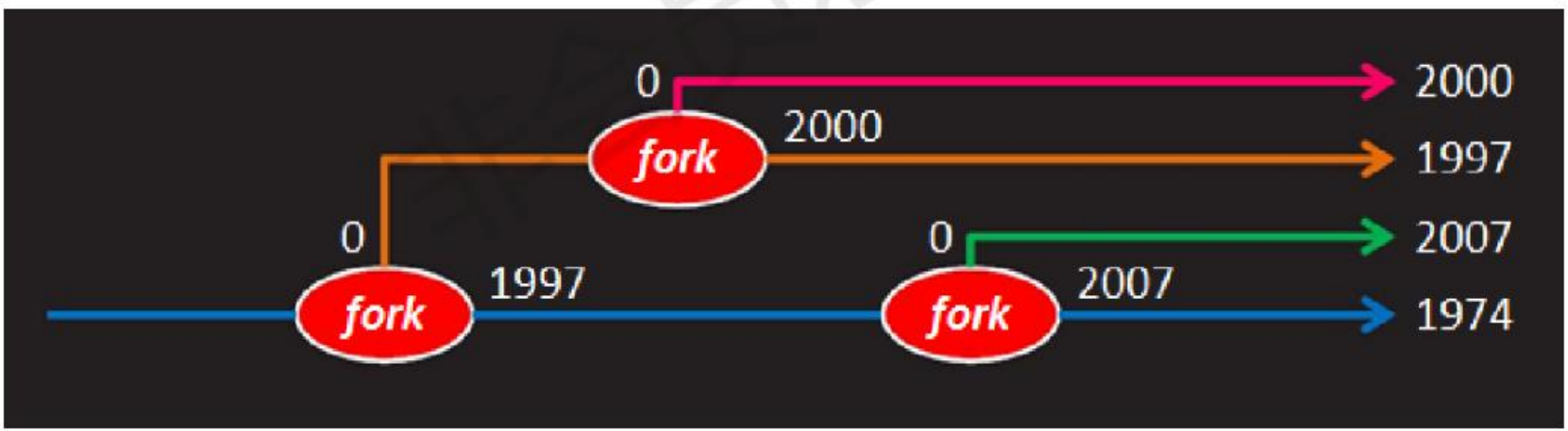
预习内容

什么是创建新进程

什么是创建新进程

什么是创建新进程

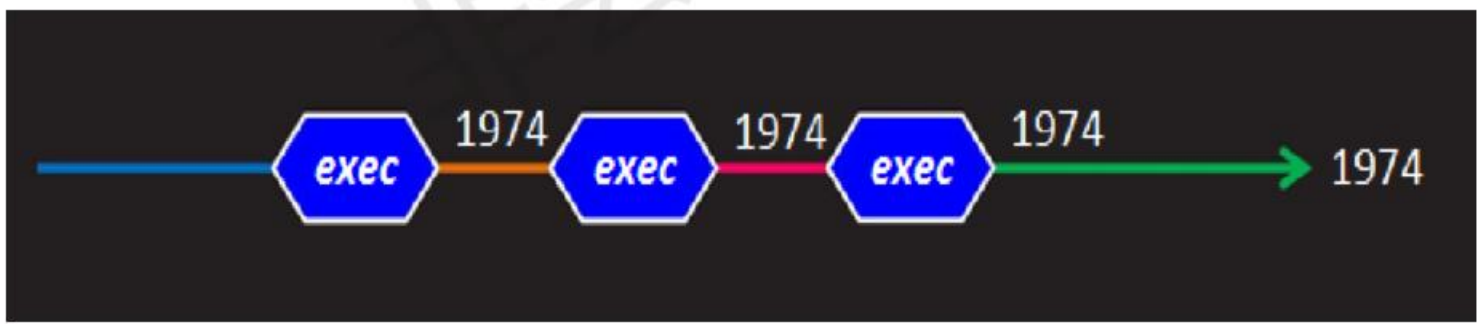
- 在前面的课程中，我们已经学习了fork函数用来创建子进程，子进程是父进程的副本，会复制父进程除代码段以外的其他数据，代码段数据和父进程共享



什么是创建新进程

知识讲解

- 而今天我们要聊的是创建新进程。与fork不同，exec函数族不是创建调用进程的子进程，而是创建一个新的进程取代调用进程自身
- 新进程会用自己的全部地址空间，覆盖调用进程的地址空间，但进程的PID保持不变



直播课见

UC

C/C++教学体系

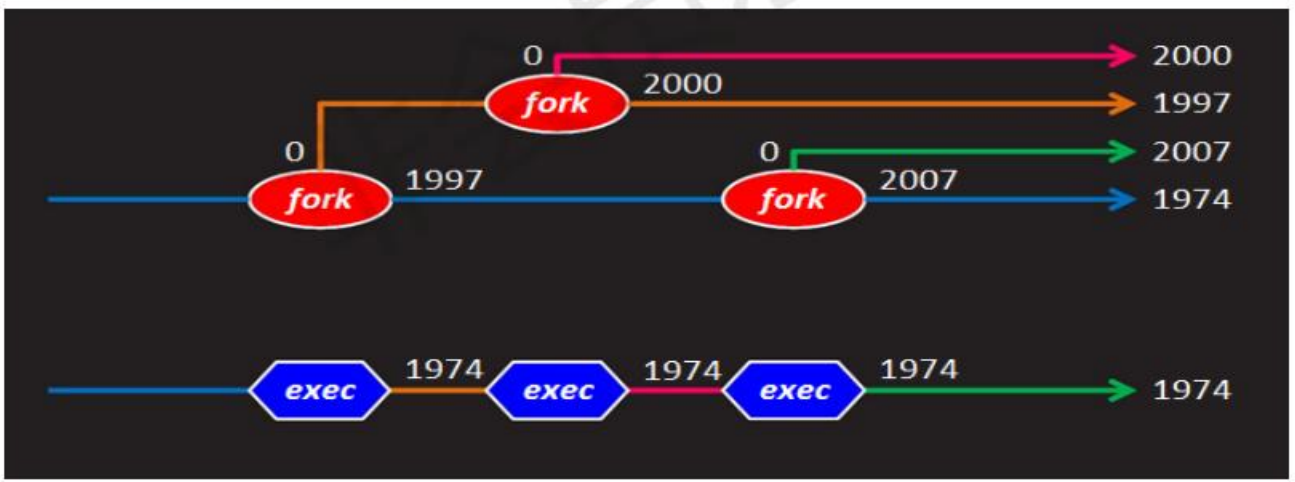
目录

新进程的创建

新进程的创建

知识讲解

- 与fork不同，exec函数不是创建调用进程的子进程，而是创建一个新的进程取代调用进程自身。新进程会用自己的全部地址空间，覆盖调用进程的地址空间，但进程的PID保持不变



新进程的创建

知识讲解

- exec不是一个函数而是一堆函数(共6个)，一般称为exec函数族。它们的功能是一样的，用法也很相近，只是参数的形式和数量略有不同
  - #include <unistd.h>
  - int execl (const char* path, const char* arg, ...);
  - int execlp (const char* file, const char* arg, ...);
  - int execle (const char* path, const char* arg, ...,char* const envp[]);
  - int execv (const char* path, char* const argv[]);
  - int execvp (const char* file, char* const argv[]);
  - int execve (const char* path, char* const argv[],char* const envp[]);

新进程的创建

知识讲解

- exec函数族一共包括6个函数，它们的函数名都是在exec后面加上一到两个字符后缀，不同的字符后缀代表不同的含义
  - l：即list，新进程的命令行参数以字符指针列表(const char* arg, ...)的形式传入，列表以空指针结束
  - p：即path，若第一个参数中不包含“/”，则将其视为文件名，并根据PATH环境变量搜索该文件
  - e：即environment，新进程的环境变量以字符指针数组(char* const envp[])的形式传入，数组以空指针结束，不指定环境变量则从调用进程复制
  - v：即vector，新进程的命令行参数以字符指针数组(char* const argv[])的形式传入，数组以空指针结束

新进程的创建

- 其实6个exec函数中只有execve是真正的系统调用，其它5个函数不过是对execve函数的简单包装

知识讲解

新进程的创建

知识讲解

- 调用exec函数不仅改变调用进程的地址空间和进程映像，调用进程的一些属性也发生了变化
  - 任何处于阻塞状态的信号都会丢失
  - 被设置为捕获的信号会还原为默认操作
  - 有关线程属性的设置会还原为缺省值
  - 有关进程的统计信息会复位
  - 与进程内存相关的任何数据都会丢失，包括内存映射文件
  - 标准库在用户空间维护的一切数据结构(如通过atexit或on_exit函数注册的退出处理函数)都会丢失

新进程的创建

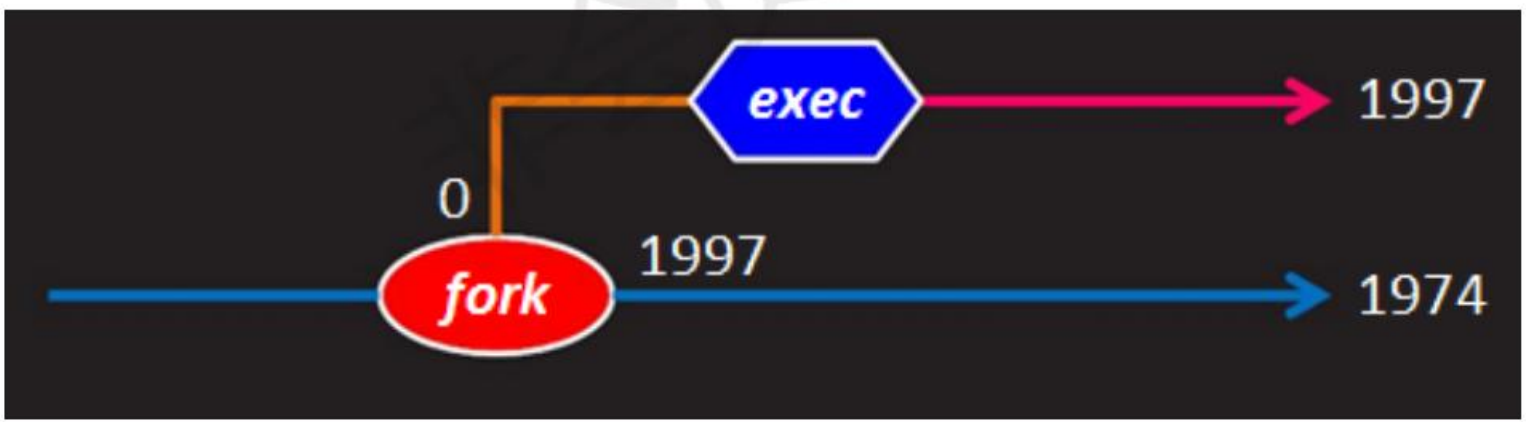
- 但也有些属性会被新进程继承下来，比如PID、PPID、实际用户ID和实际组ID、优先级，以及文件描述符等
- 注意如果进程创建成功，exec函数是不会返回的，因为成功的exec调用会以跳转到新进程的入口地址作为结束，而刚刚运行的代码是不会存在于新进程的地址空间中的。但如果进程创建失败，exec函数会返回-1

知识讲解

新进程的创建

知识讲解

- 调用exec函数固然可以创建出新的进程，但是新进程会取代原来的进程。如果既想创建新的进程，同时又希望原来的进程继续存在，则可以考虑fork+exec模式，即在fork产生的子进程里调用exec函数，新进程取代了子进程，但父进程依然存在



system

system

知识讲解

- #include <stdlib.h>
- int system (const char* command);
  - 功能：执行shell命令
  - 参数：command　shell命令行字符串
  - 返回值：成功返回command进程的终止状态，失败返回-1

  - system函数执行command参数所表示的命令行，并返回命令进程的终止状态
  - 若command参数取NULL，返回非0表示Shell可用，返回0表示Shell不可用

system

知识讲解

- 在system函数内部调用了vfork、exec和waitpid等函数
  - 如果调用vfork或waitpid函数出错，则返回-1
  - 如果调用exec函数出错，则在子进程中执行exit(127)
  - 如果都成功，则返回command进程的终止状态(由waitpid的status参数获得)

- 使用system函数而不用vfork+exec的好处是，system函数针对各种错误和信号都做了必要的处理，而且system是标准库函数，可跨平台使用

谢谢

UC Day09

复习课

设置用户ID位的运用

设置用户ID位的运用

- 在前面的课程中，我们已经了解了什么是设置用户ID位，到目前位置，我们也学完了关于进程的知识，本节课我们来实际演示下设置用户ID位对进程的影响。

知识讲解

设置用户ID位的运用

- 这是我们曾经课程中实现过的代码，用来输出进程相关的各种ID。

知识讲解

```c
//各种ID
#include<stdio.h>
#include<unistd.h>

int main(void){
    printf("进程的ID:    %d\n",getpid());
    printf("父进程的ID:  %d\n",getppid());
    printf("实际用户ID:  %d\n",getuid());
    printf("实际组ID:    %d\n",getgid());
    printf("有效用户ID:  %d\n",geteuid());
    printf("有效组ID:    %d\n",getegid());
    return 0;
}
```

设置用户ID位的运用

- 以当前tarena用户对代码进行编译，生成可执行程序，并运行。此时进程的有效用户ID是1000。

```
day09$gcc id.c -o id
day09$ls -l id
-rwxrwxr-x 1 tarena tarena 8920 9月   30 10:34 id
day09$./id
进程的ID:    15946
父进程的ID:  14649
实际用户ID:  1000
实际组ID:    1000
有效用户ID:  1000
有效组ID:    1000
day09$
```

知识讲解

设置用户ID位的运用

- 接下来我们切换用户为root，并重新编译代码，生成可执行程序，此时得到的可执行程序idroot为root用户。

知识讲解

```
day09$su root
密码：
root@tarena-virtual-machine:/home/tarena/2205/uc/day09# gcc id.c -o idroot
root@tarena-virtual-machine:/home/tarena/2205/uc/day09# ls -l idroot
-rwxr-xr-x 1 root root 8920 9月  30 10:38 idroot
root@tarena-virtual-machine:/home/tarena/2205/uc/day09#
```

设置用户ID位的运用

- 通过chmod命令，为可执行程序idroot添加设置用户ID位

知识讲解

```
root@tarena-virtual-machine:/home/tarena/2205/uc/day09# chmod u+s idroot
root@tarena-virtual-machine:/home/tarena/2205/uc/day09# ls -l idroot
-rwsr-xr-x 1 root root 8920 9月  30 10:38 idroot
root@tarena-virtual-machine:/home/tarena/2205/uc/day09#
```

设置用户ID位的运用

- 切换回tarena用户，执行idroot，此时会发现，程序运行后的有效用户ID为0，代表root用户。拥有设置用户ID位的可执行程序执行后，其进程的有效用户ID取决于文件的拥有者而非登陆用户

```
root@tarena-virtual-machine:/home/tarena/2205/uc/day09# su tarena
day09$ls -l idroot
-rwsr-xr-x 1 root root 8920 9月  30 10:38 idroot
day09$./idroot
进程的ID:   16014
父进程的ID: 16001
实际用户ID: 1000
实际组ID:   1000
有效用户ID: 0
有效组ID:   1000
day09$
```

知识讲解

下节课见