Linux进程相关系统调用

Linux进程基本概念

- 进程是程序的一次执行过程，是资源分配的最小单位
- 在Linux下，进程也CPU调度的最小单位
- 在Linux下，线程是利用**轻量级进程机制**实现的
- 在Linux下，任务（task）等价于进程
- Linux利用数据结构task_struck(进程控制块 PCB)记录进程的描述信息、控制信息和资源信息

Linux线程基本概念

- 它是进程内独立的一条运行路线，处理器调度的最小单元，也可以称为轻量级进程
- 线程可以对进程的内存空间和资源进行访问，并与同一进程中的其他线程共享
- 线程的上下文切换的开销比创建进程小得多

关系

种类

1. 用户级线程
2. 轻量级进程
3. 内核线程

Linux中的线程基于进程实现

> 相对于UNIX，进程创建开销小
>
> 用户线程对象与内核线程对象的关系是“一对一”

进程标识PID

Linux内核通过惟一的进程标识符PID来标识每个进程。PID存放在进程描述符task_struct的pid字段中。

```
pid_t  getpid();
pid_t  getppid();
```

Linux进程状态

Linux 将**进程状态主要分为五种**

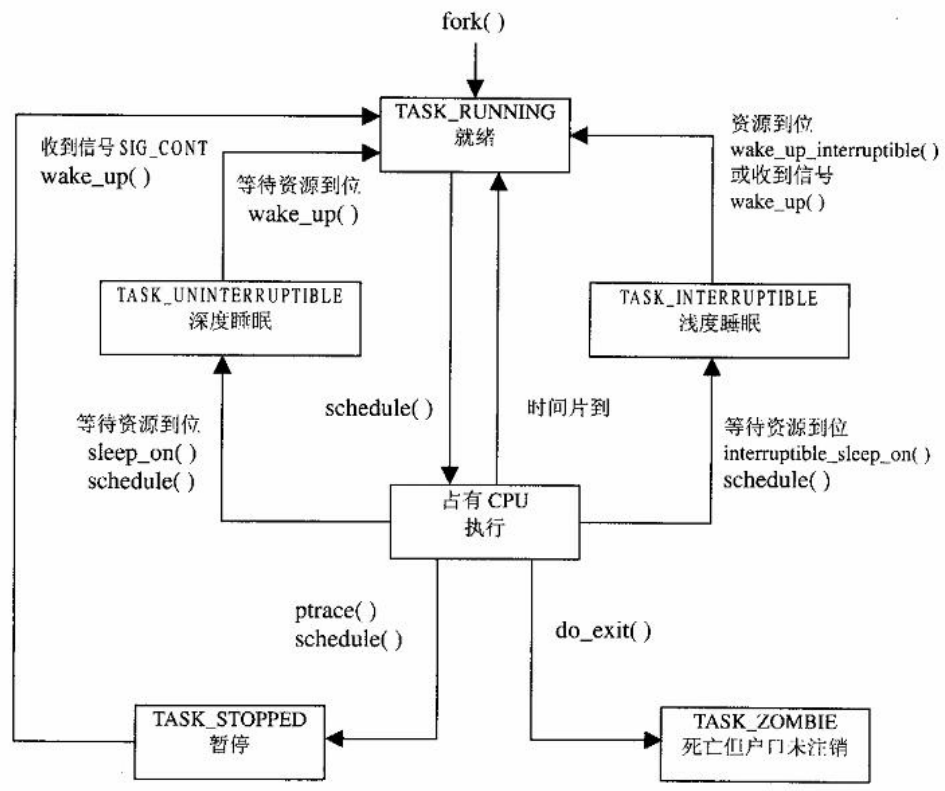
1. 运行状态
2. 阻塞状态
3. 暂停状态

4. 跟踪状态
5. 僵尸状态

细分如下

- 运行状态（TASK_RUNNING)R
- 可中断的阻塞状态(TASK_INTERRUPTIBLE)S
- 不可中断的阻塞状态(TASK_UNINTERRUPTIBLE)D
- 可终止的阻塞状态(TASK_KILLABLE)
- 暂停状态(TASK_STOPPED)T
- 跟踪状态(TASK_TRACED)T
- 僵尸状态(TASK_ZOMBIE)Z
- 僵尸撤销状态(EXIT_DEAD)X

进程状态转换



进程启动

- 手工启动
  - 前台启动：shell中输入命令：program
  - 后台启动：shell中输入命令：program&
- 调度启动
  - 利用at命令在指定时刻启动
  - 利用cron命令定期启动

在程序中创建与终止进程

进程的创建和执行

创建

fork()函数族

fork()函数的使用

子进程复制了父进程的哪些资源

- 用户ID、用户组ID、进程组ID、会话ID
- 当前工作目录、根目录、环境变量
- 文件访问权限、资源访问权限
- 信号屏蔽位
- 打开的文件描述符
- 进程地址空间（数据段、代码段、堆栈段）

对比Linux进程和Windows进程

- Fork函数（Linux）的应用逻辑—孙悟空逻辑
- CreateProcess（Windows）的应用逻辑—黑社会逻辑

执行

exec函数族负责读取可执行文件并将其载入地址空间开始运行

exec函数🐵

exec函数用于创建一个新的进程，新进程以另一个可执行程序为执行脚本。exec创建的新进程“占用了”原进程的绝大部分资源，进程地址空间中装入了新的可执行程序。exec执行成功之后，原进程就“消失了。

exec函数族使用区别

- 查找方式
- 参数传递方式
- 环境变量
- 真正的系统调用是execve()

exec与fork配合使用的效率问题

在Linux时代，fork函数实现中引入了“写时拷贝Copy On Write”技术，fork+exec的配合效率也很高

MMU为共享内存及写时拷贝的支持

> **虚拟内存（Virtual Memory）**
>
> **页表（Page Table）**
>
> **地址转换过程**
>
> **内存保护和权限**

进程的终止

进程终结也需要做很多繁琐的收尾工作，系统必须保证进程所占用的资源回收，并通知父进程

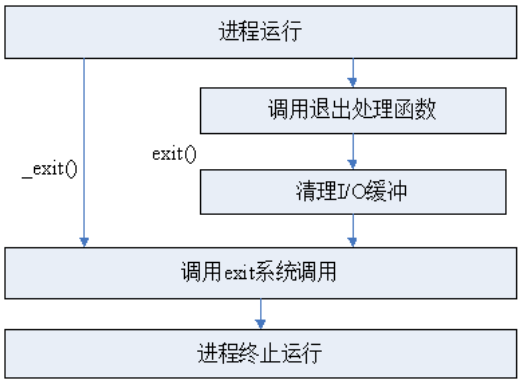
1. Linux首先把终止的进程设置为**僵尸状态**，它的存在只为父进程提供信息，申请死亡
2. 父进程得到信息后，开始调用wait函数族，最终赐死子进程
3. 子进程占用的所有资源被全部释放

使用exit()族函数

exit()和_exit()函数😻

1. exit()和*exit()函数都是用来终止进程的。当程序执行到exit()或*exit()时，进程会无条件地停止剩下的所有操作，清除包括各种数据结构，并终止本进程的运行
2. atexit用于注册一个或多个退出函数

exit()和_exit()的执行过程



僵尸进程😽

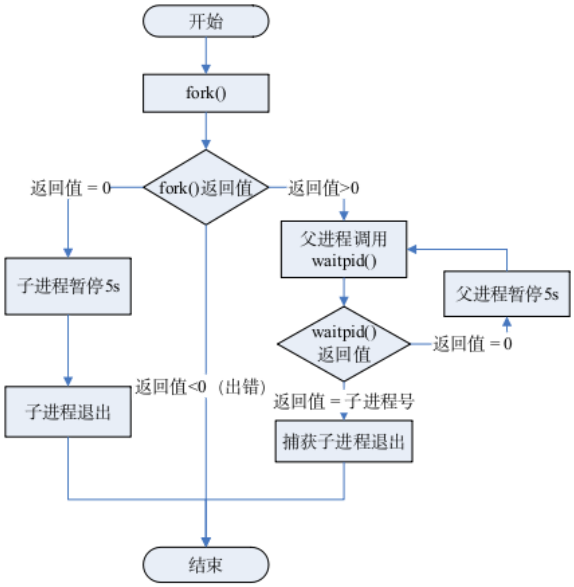
僵尸进程产生条件

1. 子进程执行完毕
2. 父进程没有回收其状态

子进程退出前会向父进程发送SIGCHLD信号，父进程用wait和waitpid回收

wait()和waitpid()

- wait()函数是用于使父进程（也就是调用wait()的进程）阻塞，直到一个子进程结束或者该进程接到了一个指定的信号为止。如果该父进程没有子进程或者他的子进程已经结束，则wait()就会立即返回
- waitpid()的作用和wait()一样，但它并不一定要等待第一个终止的子进程，它还有若干选项，如可提供一个非阻塞版本的wait()功能



守护进程

守护进程，也就是通常所说的Daemon进程，是Linux中的后台服务进程。它是一个生存期较长的进程，通常独立于控制终端并且周期性的执行某种任务或等待处理某些发生的事件

1. 在系统引导装入时启动，在系统关闭时终止。不能正常被kill关闭
2. 很多守护进程，大多数服务都是用守护进程实现的
3. 守护进程能够突破这种限制，它从被执行开始运转，直到整个系统关闭才会退出。当控制终端被关闭时，相应的进程都会被自动关闭。

守护进程编写规范

编写守护进程（完整过程）

1. 创建子进程，父进程退出
2. 在子进程中创建新会话
   - 进程组
   - 会话期：多个进程组的集合

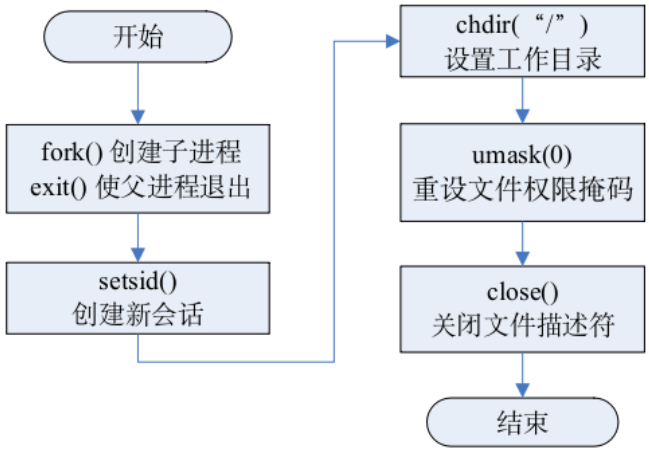
   `setsid()`函数

   > linux内核——会话、进程组、线程组

3. 改变当前目录为根目录
4. 重设文件权限掩码

   通常的使用方法为`umask(0)`
5. 关闭文件描述符

Linux守护进程创建流程



守护进程出错处理

syslog是Linux中的系统日志管理服务，通过守护进程syslogd来维护。

> 该机制提供了3个`syslog`相关函数，分别为`openlog()`、`syslog()`和`closelog()`。`openlog()`函数用于打开系统日志服务的一个连接；`syslog()`函数是用于向日志文件中写入消息，在这里可以规定消息的优先级、消息输出格式等；`closelog()`函数是用于关闭系统日志服务的连接。