

欢迎志同道合的朋友一起玩技术.....

输入关键字

果冻想

Q

[🏠 首页 \(http://www.jellythink.com\)](#)    [</> 编程语言 \(http://www.jellythink.com/archives/category/language\)](#)

[📁 系列专栏 \(http://www.jellythink.com/archives/category/columns\)](#)    [👤 关于果冻 \(http://www.jellythink.com/about\)](#)    Q

# Linux进程基础知识总结 (http://www.jellythink.com/archives/900)

2015-01-16    分类：Linux开发 (http://www.jellythink.com/archives/category/linuxdevelopment) / Linux程序设计 (http://www.jellythink.com/archives/category/linuxdevelopment/linuxprogramming)

阅读(984)    评论(9)

## 进程

进程表示一个正在运行的程序实例，它是分配资源的最小单位，这种说法特别官方。

进程是一个非常重要的东西，我们运行的系统中同时跑着N个进程，这些进程都在默默的工作着，我们编写的代码，经过编译、运行，也会生成一个进程。这个进程由程序代码、数据、变量（占用着系统内存）、打开的文件（文件描述符）和环境组成。一般来说，对于Linux系统，系统会在进程之间共享程序代码和系统函数库，所以在任何时刻，内存中都只存在代码的一份副本。

由于进程这个东西如此的重要，这篇文章就对Linux中的进程进行一个基本的总结。

## 创建一个进程

故事还是先从创建一个进程开始。看看下面的代码：

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main()
{
    pid_t pid;
    char *msg;
    int n;

    printf("Fork program starting\n");
    pid = fork(); // 创建一个进程

    // 创建完成以后，父进程和子进程都从这里开始运行
    switch (pid)
    {
        case -1:
            perror("Fork failed.");
            exit(EXIT_FAILURE);

        case 0:
            // 当fork返回值为0时，表示这是子进程
            msg = "This is the child";
            n = 2;
            break;

        default:
            // 当fork不为0和-1时，表示这是父进程
            msg = "This is the parent";
            n = 3;
            break;
    }

    for (; n > 0; --n)
    {
        printf("%s\n", msg);
        sleep(1);
    }

    exit(EXIT_SUCCESS);
}
```

我们使用 fork 函数创建一个子进程， fork 函数的声明如下：

```
#include <unistd.h>
pid_t fork(void);
```

fork 函数返回一个pid\_t类型的值， pid\_t类型就是一个int类型。当调用 fork 创建进程成功以后，子进程和父进程同时都从 fork 函数之后的代码开始运行。那么既然都从同一个地方开始运行，如何区分是子进程还是父进程呢？

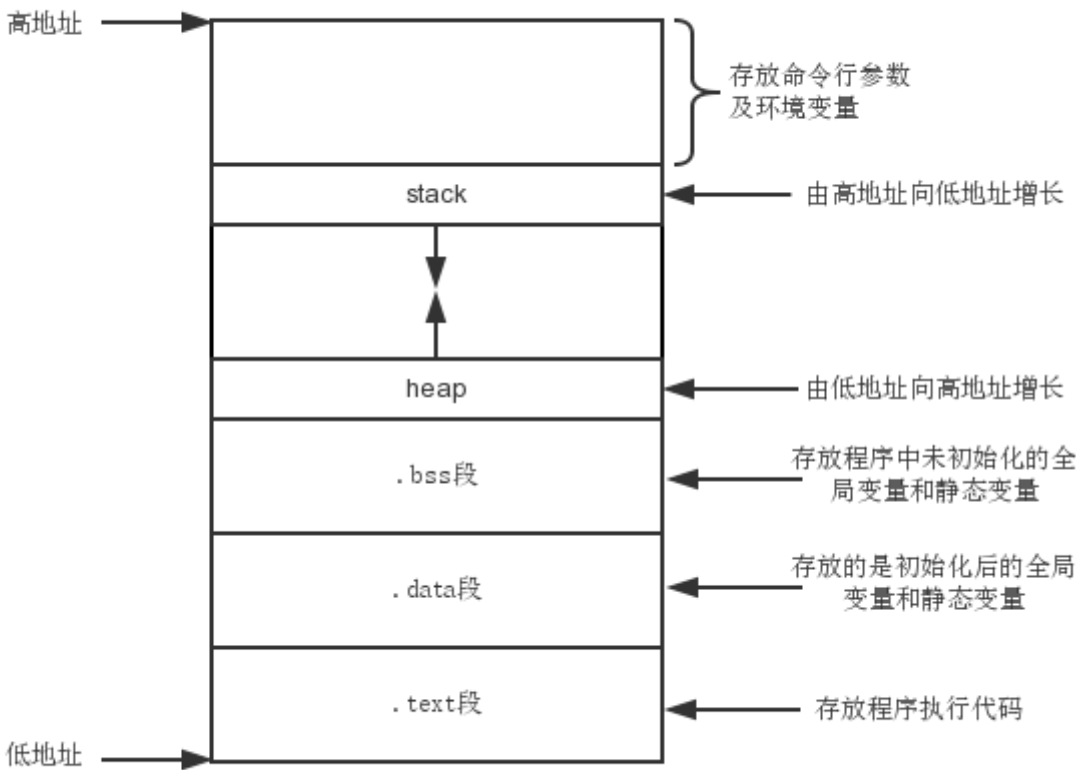
Linux中是使用 fork 的返回值来进行区分的，方法如下：

- 当fork返回pid\_t的值为-1时，就表示创建进程出现了错误；
- 当fork返回pid\_t的值为0时，就表示是子进程；
- 当fork返回pid\_t的值不为0和-1时，就表示是父进程。

在上面的代码中，我就是根据 fork 的返回值来判断是子进程还是父进程的。怎么都觉的怪怪的。

## 进程的结构

再说一次，进程很重要，我们需要去理解，那么对于进程的结构就不得不说了。那么我们使用 fork 创建了一个进程以后，该进程在内存内部的结构是如何的呢？我们来看看。



如上图所示，我们在实际编码时并没有考虑到图中所说的这些内容，而上图所展示的东西，却是理解父进程和子进程的关键，关于进程的很多问题，我们也是基于上图出发进行思考的。在后面，我会专门拿出几个问题来分析为什么进程内存映像如此重要。

## 进程调度

在单个处理器上，同一时间只能有一个进程可以运行，其它进程都处于等待运行状态。但是，我们实际的感觉是同一时刻有多个进程在“同时”进行运行，这是为什么呢？

操作系统会分给每个进程一定的运行时间，叫做“时间片”。进程在这个“时间片”内运行，由于“时间片”非常的短，这样就给人一种多个程序在同时运行的假象。操作系统是如何调度进程的呢？进程调度的规则有很多，比如根据优先级进行调度算法，先进先出调度算法等。

Linux内核进程调度器是根据进程的优先级来进行进程调度的。优先级高的进程运行得更为频繁。

## 进程状态

Linux上进程有5种状态：

1. 运行(正在运行或在运行队列中等待)
2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)
3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)

- 4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)
- 5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行)

当我们使用 `ps -aux` 命令查看进程状态时，那些标识进程的字母又是什么意思呢？，具体的进程状态的字符标识如下表所示：

状态标志	状态描述
D	不可中断
R	运行
S	中断
T	停止
Z	僵死

但是，有的时候，我们还会看到一些其它的标识，例如：

状态标志	状态描述
W	僵死
<	高优先级进程
N	低优先级进程
L	内存锁页

看到上面又是中断啊，不可中断啊，停止啊，僵死啊，直接晕死，那么到底如何理解这些概念呢？

1. 运行状态

在Linux中，仅等待CPU时间的进程称为就绪进程，它们被放置在一个运行队列中，一个就绪进程的状态标志位为TASK\_RUNNING。一旦一个运行中的进程时间片用完，Linux内核的调度器会剥夺这个进程对CPU的控制权，并且从运行队列中选择一个合适的进程投入运行。

2. 睡眠状态

Linux中的进程睡眠状态有两种：

- 一种是可中断的睡眠状态，其状态标志位TASK\_INTERRUPTIBLE；
- 另一种是不可中断的睡眠状态，其状态标志位为TASK\_UNINTERRUPTIBLE。可中断的睡眠状态的进程会睡眠直到某个条件变为真，比如说产生一个硬件中断、释放进程正在等待的系统资源或是传递一个信号都可以是唤醒进程的条件。不可中断睡眠状态与可中断睡眠状态类似，但是它有一个例外，那就是把信号传递到这种睡眠状态的进程不能改变它的状态，也就是说它不响应信号的唤醒。不可中断睡眠状态一般较少用到，但在一些特定情况下这种状态还是很有用的，比如说：进程必须等待，不能被中断，直到某个特定的事件发生。  
当向进程发送一个SIGSTOP信号，它就会因响应该信号而进入TASK\_STOPPED状态（除非该进程本身处于TASK\_UNINTERRUPTIBLE状态而不响应信号）。（SIGSTOP与SIGKILL信号一样，是非常强制的。不允许用户进程通过signal系列的系统调用重新设置对应的信号处理函数。）  
向进程发送一个SIGCONT信号，可以让其从TASK\_STOPPED状态恢复到TASK\_RUNNING状态。

3. 僵死状态

进程在退出的过程中，处于TASK\_DEAD状态。在这个退出过程中，进程占有的所有资源将被回收，除了task\_struct结构（以及少数资源）以外。于是进程就只剩下task\_struct这么个空壳，故称为僵尸。之所以保留task\_struct，是因为task\_struct里面保存了进程的退出码、以及一些统计信息。而其父进程很可能会关心这些信息。比如在shell中，\$?变量就保存了最后一个退出的前台进程的退出码，而这个退出码往往被作为if语句的判断条件。  
当然，内核也可以将这些信息保存在别的地方，而将task\_struct结构释放掉，以节省一些空间。但是使用task\_struct结构更为方便，因为在内核中已经建立了从pid到task\_struct查找关系，还有进程间的父子关系。释放掉task\_struct，则需要建立一些新的数据结构，以便让父进程找到它的子进程的退出信息。

父进程可以通过wait系列的系统调用（如wait4、waitid）来等待某个或某些子进程的退出，并获取它的退出信息。然后wait系列的系统调用会顺便将子进程的尸体（task\_struct）也释放掉。

子进程在退出的过程中，内核会为其父进程发送一个信号，通知父进程来“收尸”。这个信号默认是SIGCHLD，但是在通过clone系统调用创建子进程时，可以设置这个信号。

通过下面的代码能够制造一个EXIT\_ZOMBIE状态的进程：

```
if (fork()) while(1) sleep(100);
```

使用 `ps -aux` 就会看到如下一条僵死进程的信息。

只要父进程不退出，这个僵尸状态的子进程就一直存在。那么如果父进程退出了呢，谁又来给子进程“收尸”？

当进程退出的时候，会将它的所有子进程都托管给别的进程（使之成为别的进程的子进程）。托管给谁呢？可能是退出进程所在进程组的下一个进程（如果存在的话），或者是1号进程。所以每个进程、每时每刻都有父进程存在。除非它是1号进程。

## 总结

哦，这篇文章总结了不少内容，大体上都是一些你在大学操作系统课就应该学会的东西。好吧，我大学操作系统课都在睡觉，导致现在还要写这么一篇文章来复习这些东西，罪过，罪过。

2015年1月13日 于深圳。

未经允许不得转载：果冻想 (<http://www.jellythink.com>) » Linux进程基础知识总结 (<http://www.jellythink.com/archives/900>)

分享到：更多 ()

标签： Linux (<http://www.jellythink.com/archives/tag/linux>)

## 相关推荐

- Linux expect详解 (<http://www.jellythink.com/archives/1470>)
- Linux netstat命令详解 (<http://www.jellythink.com/archives/1466>)
- 聊聊Linux Shell (<http://www.jellythink.com/archives/1323>)
- 玩玩awk (<http://www.jellythink.com/archives/1252>)
- awk中的函数 (<http://www.jellythink.com/archives/1248>)
- 玩玩Sed (<http://www.jellythink.com/archives/1145>)

## 评论 9



你的评论可以一针见血

提交评论

昵称

昵称 (必填)

邮箱

邮箱 (必填)

网址

网址



支持个，学习中。  
偏方大全qianjinpianfang.com (<http://qianjinpianfang.com/>) 1年前 (2015-01-19)

#5



我喜欢你的文章！加油~  
seo (<http://www.02942.cn/>) 1年前 (2015-04-11)

#4



谢谢支持。  
果冻想 (<http://www.jellythink.com/>) 1年前 (2015-04-11)




大神，请教一个问题，怎么感觉很多服务器开发都是用C的，我知道C速度是比较快，难道C++的面向对象难道只用在windows软件开发上与服务器的某些模块上吗？  
浩yan (<http://weibo.com/1974789652>) 1年前 (2015-05-29)

#3



开发的时候，选择什么语言，与项目的历史遗留问题有关；与你们的技术老大有关，与项目成员的技术能力有关，并不是说只用某中语言。比如电信系统，那是多少年前开发的，那个时候都是用的C语言，现在在这个基础上开发，基本都是继续使用C语言，至少我们这里是；使用什么语言，是多方面因素决定的，并没有绝对的原因。所以说，你喜欢用什么，大胆用就好了。

果冻想 (<http://www.jellythink.com/>) 1年前 (2015-05-29)

- 

不错的网站，很喜欢，期待互访  
最励志官网 (<http://www.zuilizhi.net/?ds>) 9个月前 (09-19)
- #2



留个脚印!  
陈小子 (<http://www.diefishfish.com/?ds>) 9个月前 (09-19)

#1

# 在这里玩技术，享受技术带来的疯狂

捐赠名单 (<http://www.jellythink.com/donations/>)

关于果冻 (<http://www.jellythink.com/about/>)

