目录

[一、 进程 1](#_Toc160137451)

[1. 基本概念 1](#_Toc160137452)

[2. 结构特征 1](#_Toc160137453)

[3. 查看进程信息 1](#_Toc160137454)

[二、 创建进程 2](#_Toc160137455)

[1. 进程复制 2](#_Toc160137456)

[2. 子进程“偷梁换柱” 2](#_Toc160137457)

[三、 进程退出 4](#_Toc160137458)

[1. 正常退出 4](#_Toc160137459)

# 进程

## 基本概念

程序：静态文件

进程：运行着的实体

进程是一个实体。每一个进程都有它自己的地址空间，一般情况下，包括文本区域（text region）、数据区域（data region）和堆栈（stack region）。文本区域存储处理器执行的代码；数据区域存储变量和进程执行期间使用的动态分配的内存；堆栈区域存储着活动过程调用的指令和本地变量。

进程是一个“执行中的程序”。程序是一个没有生命的实体，只有处理器赋予程序生命时（操作系统执行之），它才能成为一个活动的实体，我们称其为进程。

## 结构特征

进程由程序、数据和进程控制块（PID）三部分组成。

## 查看进程信息

进程关系：pstree（进程之间存在“父子关系”“兄弟关系”这样的亲缘关系）

进程的身份证PID：ps -ef

# 创建进程

## 进程复制

函数原型：pid\_t fork(void);

fork() 这个函数很特殊，成功创建子进程后居然有两个返回值，给父进程返回子进程pid，给子进程返回 0，如果创建失败那么就返回 -1。

fork函数执行完后会复制一份原来的进程（创建进程）。

fork函数后面的代码会执行两遍。fork之后父进程先执行还是子进程先执行不确定，取决于内核所使用的调度算法。

## 子进程“偷梁换柱”

当用fork函数创建新的子进程后，子进程往往要调用一种exec函数以执行另一个程序。当程序调用一种exec函数时，该进程执行的程序完全替换为新程序，而新程序则从其main函数开始执行。因为调用exec并不创建新进程，所以前后的进程ID并未改变。exec只是用磁盘上的一个新程序替换了当前进程的正文段、数据段、堆段和栈段。

有7种不同的exec函数可供使用，它们常常被统称为exec函数。

**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** execvpe(**const** **char** \*file, **char** \***const** argv[],**char** \***const** envp[]);

\*path：可执行文件的路径名。

\*arg：可执行程序所带的参数，第一个参数为可执行文件名字，没有带路经且arg必须以NULL结束。

\*file：如果参数file中包含/，则就将其视为路径名，否则就按PATH环境变量，在它所指定的各目录中搜寻可执行文件。

函数名中的字符可以帮助理解和分辨：

l（list）表示以参数列表的形式调用（需要指定绝对路径执行）。

v（vector）表示以参数数组的方式调用

e（environment）表示用户提供自定义环境变量，当调用execve函数时，操作系统首先根据filename指定的路径和名称找到对应的可执行文件。然后，操作系统创建一个新的进程，并将该可执行文件加载到新进程的内存空间中。接下来，操作系统将新进程的参数和环境变量设置为argv和envp指定的内容。最后，操作系统启动新进程的执行，从新程序的入口点开始执行代码。

p（path）表示PATH中搜索执行的文件，如果给出的不是绝对路径就会去PATH搜索相应名字的文件，如PATH没有设置， 则会默认在/bin,/usr/bin下搜索。

* 返回值：

如果执行成功则函数不会返回，执行失败则直接返回-1，失败原因存于errno 中。

* 注意事项

l、v后缀必须任选其一使用，p、e任选其一。

有可能执行失败：文件路径错误；没有加NULL结尾；新程序没有执行权限

# 进程退出

## 正常退出

exit()函数用于正常退出进程。当调用exit()函数时，它会执行一些清理操作（如关闭打开的文件描述符）并终止进程。它还通过返回一个退出状态码将控制权返回给操作系统。exit()函数可用于任意函数中，通过调用它可以使整个进程退出。

\_exit()系统调用是直接终止进程。当调用\_exit()函数时，进程立即终止，而不执行任何清理操作。与exit()不同，\_exit()函数不返回任何状态给操作系统，而是直接终止进程。通常，\_exit()函数用于异常情况或在需要立即终止进程而不进行清理操作的情况下。

return语句用于退出函数或方法，并将控制权返回到调用该函数或方法的位置。return语句只能用于函数或方法的内部，而不能使整个进程退出。当函数或方法中的所有代码都执行完毕或遇到return语句时，该函数或方法的执行将结束，控制权将返回给调用者。