<https://blog.csdn.net/u013589130/article/details/45441973>

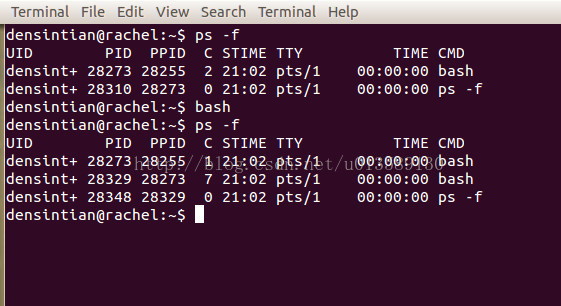
1. 基本知识

在Linux 是一个多用户多任务的操作系统。同时会有不同的用户，向操作系统发出各种命令。  
命令通过程序执行。在执行的过程中，就会产生进程。这里我们讲程序和进程之间的关系，我们可以这样理解： 程序是一些以文件的形式存储在操作系统文件系统中，包含可执行代码的静态文件。这些文件在没有被运行的时候，和其他的Word、Excel文档没有本质的区别。  
当程序被执行的时候，程序要被读入到内存中并且被执行。在执行的过程中，程序在内存中执行过程中的实例化的体现，就是进程。一个实例可能有很多个进程，每一个进程也可能有子进程。  
在进程的执行过程中，操作系统会给进程分配一定的资源（内存，磁盘和设备等），为了区别不同的进程，系统会为不同进程分配唯一的ID进行识别。  
系统内部对进程的管理包含以下的状态：  
新建—— 进程正在被创建  
运行—— 进程正在运行  
阻塞—— 进程正在等待某个命令发生  
就绪—— 进程正在等待CPU执行命令  
完成—— 进程已经结束了，操作系统正在回收资源

2. 通过Shell开列系统进程

     下面的Shell执行了3个命令，分别来解读。  
      (1) 【ps -f】命令列出当前用户的进程。 第一行bash，它的进程ID是28273，这个进程就是我们运行的Shell本身，【ps -f】 命令在这个Shell中执行，所以第二行ps -f的父进程ID就是第一行的进程ID，同时【ps -f】命令也具有自己的独立进程ID。

      (2)  【bash】命令在当前Shell中，又创建了一个Shell进程  
      (3)  【ps -f】命令在这时候列出当前用户的进程时候，这时候可以发现在开始bash进程中，打开了一个bash进程，然后执行【ps -f】命令列出进程的过程。三个进程中，上边的一个进程都是下面一个进程的父进程。



3. Linux C获得当前程序进程及用户信息

**[cpp]** [view plain](https://blog.csdn.net/u013589130/article/details/45441973) [copy](https://blog.csdn.net/u013589130/article/details/45441973)

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. \* Program Assignment : 输出进程ID和用户基本信息
3. \* Author: Densin.Tian
4. \* Date: 2015/05/02
5. \* Description:
6. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
8. #include <unistd.h>
9. #include <pwd.h>
10. #include <sys/types.h>
11. #include <stdio.h>
12. #include <stdlib.h>
14. **int** main(**int** argc, **char** \*\*argv){
15. pid\_t my\_pid, parent\_pid;
16. uid\_t my\_uid, parent\_uid;
17. gid\_t my\_gid, parent\_gid;
19. **struct** passwd \*my\_info;
21. my\_pid      = getpid();
22. parent\_pid  = getppid();
23. my\_uid      = getuid();
24. parent\_uid  = geteuid();
25. my\_gid      = getgid();
26. parent\_gid  = getegid();
28. my\_info     = getpwuid(my\_uid);
30. printf("===================================\n");
31. printf("        Process Information        \n");
32. printf("===================================\n");
33. printf("Process ID : %ld\n", (**long**)my\_pid);
34. printf("Parent  ID : %ld\n", (**long**)parent\_pid);
35. printf("User    ID : %ld\n", (**long**)my\_uid);
36. printf("Effective User  ID : %ld\n", (**long**)parent\_uid);
37. printf("Group   ID : %ld\n", (**long**)my\_gid);
38. printf("Effective Group ID : %ld\n", (**long**)parent\_gid);

41. **if**(my\_info){
42. printf("===================================\n");
43. printf("           User Information        \n");
44. printf("===================================\n");
45. printf("My Login Name : %s\n", my\_info->pw\_name);
46. printf("My Password   : %s\n", my\_info->pw\_passwd);
47. printf("My User ID    : %ld\n", (**long**)my\_info->pw\_uid);
48. printf("My Group ID   : %ld\n", (**long**)my\_info->pw\_gid);
49. printf("My Real Name  : %s\n", my\_info->pw\_gecos);
50. printf("My Home Dir   : %s\n", my\_info->pw\_dir);
51. printf("My Work Shell : %s\n", my\_info->pw\_shell);
52. }
54. **return** 0;
55. }

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u013589130/article/details/45441973