**进程组**

进程组是指一个或多个进程的集合。通常与一个作业相关联，可以接收来自同一终端的信号。每个进程组有一个唯一的进程组ID，它类似于进程ID，是一个正整数（其实就是组长进程的进程ID）。

可以通过函数获得

#include<unistd.h>

pid\_t getpgrp(void);

**进程组都有一个组长，组长进程的标识是其进程组ID等于进程ID。**

组长可以创建一个组长进程组，创建该进程组的进程，然后终止。进程存在于组长存在没必要关系，只要进程组有一个进程存在，则进程组就存在。进程组生命周期为组长创建到最后一个进程离开为止。进程组中最后一个进程可以终止，或者转移到另一个进程组中。

可以通过下面函数加入一个进程组或创建一个进程组:

#include<unistd.h>

int setpgid(pid\_t pid, pit\_t pgid);

**一个进程只能为它自己或自己的子进程设置进程组ID**，在它的子进程调用了exec函数后，它就不能再改变该子进程的进程组ID了。

在大多数作业控制的shell中，在fork之后调用setpgid函数，使父进程设置其子进程的进程组ID，并且使子进程设置其自己的进程组ID。如果不这样做，那么fork之后，由于父、子进程运行先后次序的不确定，会造成在一段时间内(父、子进程只运行了其中一个)子进程组员身份的不确定(取决于哪个进程先执行)，这就产生了竞争条件。

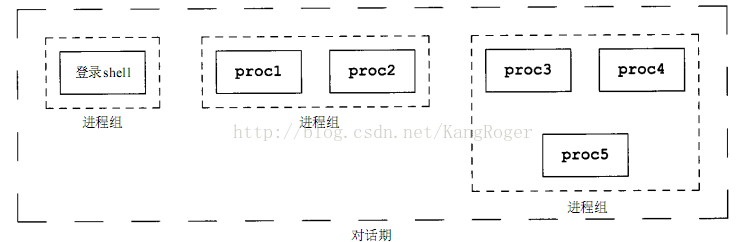
**会话（session）**

会话是一个或多个进程组的集合。shell的管道线将几个进程编成一组。

proc1 | proc2 &

proc3 | proc4 | proc5

编成的会话如下图：



进程调用setsid函数建立一个新会话

#include<unistd.h>

pid\_t setsid(void);

如果调用此函数的进程不是一个进程组的组长，则此函数会创建一个新会话，结果将发生下面3件事：

1、该进程变成新会话首进程。此时，该进程是新会话的唯一的进程。

2、该进程成为一个新进场组的组长进程。新进程组的ID是该调用进程的进程ID。

3、该进程没有控制终端。如果有，在调用后，这种联系也会被中断。

如果该进程是一个进程组的组长进程，则此函数返回出错。为了保证不会发生这种情况，通常先调用fork，然后使其父进程终止，而子进程则继续。

setsid()函数的作用：

1. 让进程摆脱原会话的作用；
2. 让进程摆脱进程组的控制
3. 让进程摆脱控制终端的控制