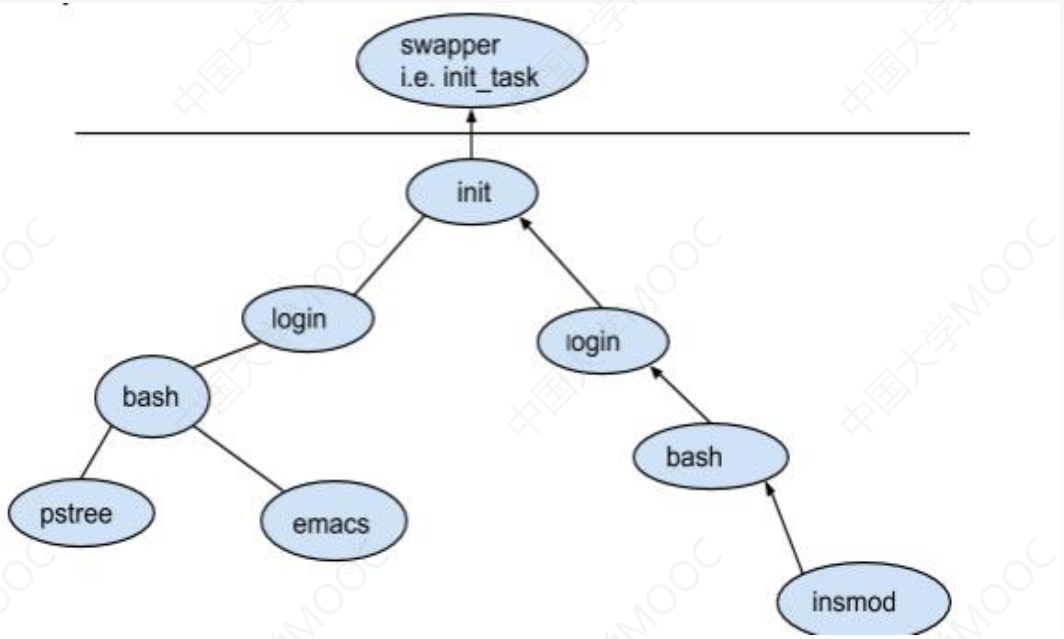
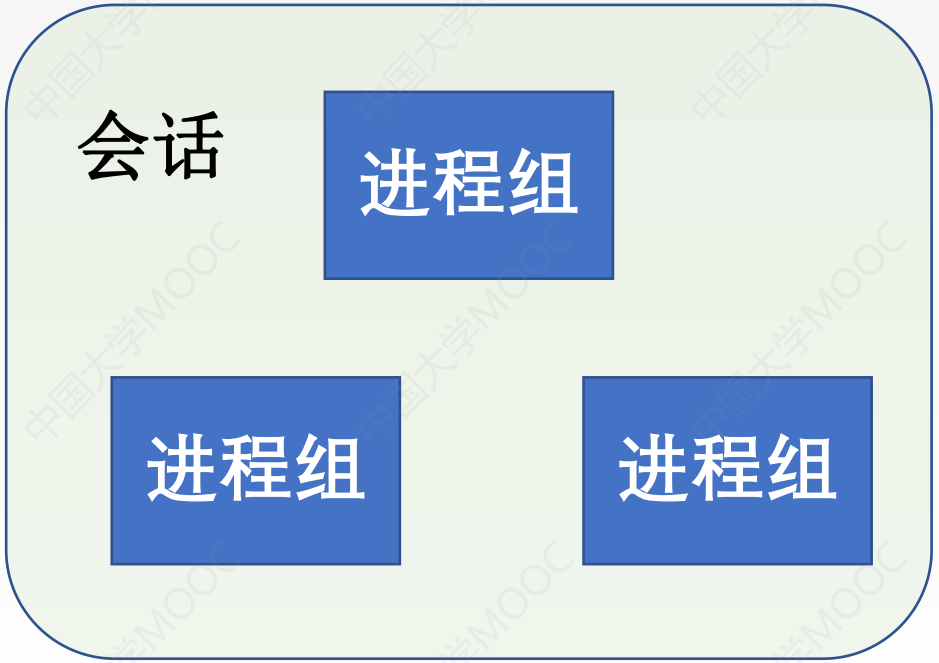



第3章 进程管理

——进程属性、守护进程

进程的组织属性


swapper
i.e. init_task
init
login
login
bash
bash
pstree
emacs
insmod


会话
进程组
进程组
进程组

进程组

- **集合：** 进程组是一个或多个进程的集合，通常与同一作业相关联，接收来自同一终端的各种信号。
- **有编号：** 每个进程属于一个进程组，有进程组号，即进程组长的进程号。
- **组长权力大：** 进程组长可以创建一个进程组，可以创建该组中的进程。
- **进程组的存在：** 存在与否与进程组长是否存在无关。
- **访问进程组号：** 可使用getpgid获取进程组号、setpgid设置进程组号

getpgid

getpgid	
功能	获取进程的进程组号
头文件	/usr/include/unistd.h
函数原型	pid_t getpgid(pid_t pid);
参数	pid 进程号（为0时，表示当前进程）
返回值	>0 pid进程的进程组号
	-1 出错

setpgid

setpgid		
功能	加入或创建一个进程组 将pid进程的组ID设为pgid	
头文件	/usr/include/unistd.h	
函数原型	int setpgid(pid_t pid, pid_t pgid);	
参数	pid	待处理的进程号
	pgid	进程组号
返回值	0	成功
	-1	失败

说明：

pid=pgid： 由pid指定的进程变成进程组长

pid=0： 则使用调用进程的PID作为进程号

pgid=0： 由pid指定的进程的进程号将用做进程组ID

会话(session)是一个或多个进程组的集合。如下图所示：

bash: proc1 | proc2 &

bash: proc3 | proc4 | proc5

会话

管道命令“|”将几个进程编成一组

会话概念与示例

会话(session)是一个或多个进程组的集合。如下图所示：

终端1

```
root@ubuntu:4# ./endless_loop|sleep 100|sleep 200
```

终端2　进程组ID　会话ID　终端

```
root@ubuntu:~# ps axjf|grep -E 'endless_loop|sleep|bash'
PID   PID   PGID  SID   TTY      TPGID STAT UID TIME COMMAND
48334 48372 48372 48372 pts/17   48541 Ss   0   0:00  |   \_ -bash
48372 48542 48541 48372 pts/17   48541 S+   0   0:00  |      \_ grep --color=auto
-E endless_loop|sleep|bash
48482 48521 48521 48521 pts/18   48538 Ss   0   0:00      \_ -bash
48521 48538 48538 48521 pts/18   48538 R+   0   0:10          \_ ./endless_loop
48521 48539 48538 48521 pts/18   48538 S+   0   0:00          \_ sleep 100
48521 48540 48538 48521 pts/18   48538 S+   0   0:00          \_ sleep 200
```

setsid	
功能	创建一个新会话
头文件	/usr/include/unistd.h
函数原型	pid_t setsid(void)
返回值	>0 成功 返回会话首进程的ID
	-1 出错

如果调用此函数的进程并非某进程组的组长，则会有以下变化：

1.创建一个新会话（创建会话）

2.调用此函数的进程成为新会话首进程（成为首进程）

3.调用进程成为一个新进程组的组长进程，此新进程组的ID是调用进程的ID（变为组长）

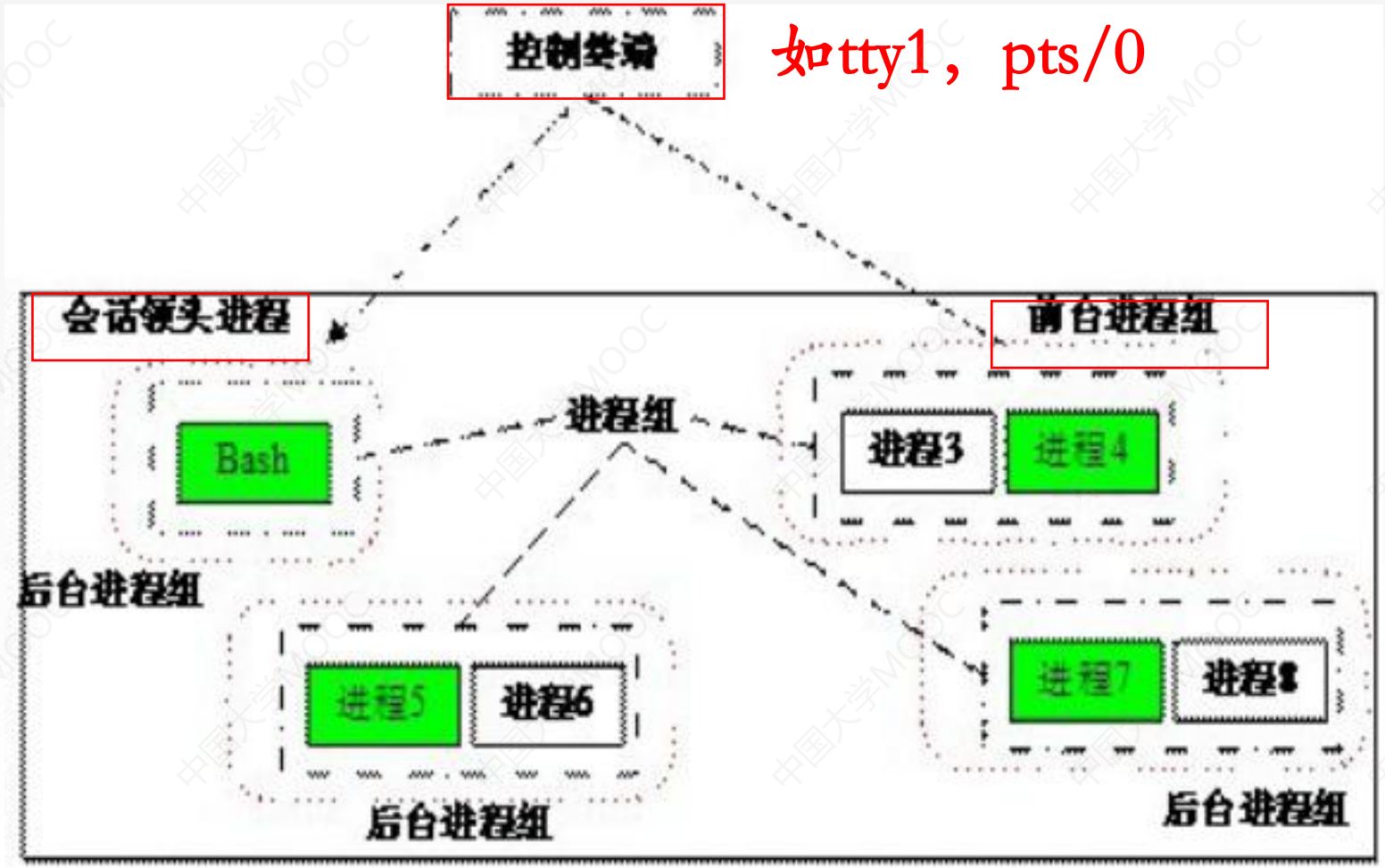
4.该进程没有控制终端（脱离控制终端）

1.如果调用进程本身是某个进程组的组长，则调用setsid会出错。
通常用以下代码来保证调用setsid的进程不是进程组长

```
if((pid=fork())>0)   exit();
    setsid();
```

2.会话没有ID号，通常把会话首进程的进程组ID号视为会话ID号，使用getsid函数将会返回会话首进程的进程组ID号。

控制终端



后台进程组接收不到[ctrl]+c中断键等信号

控制终端

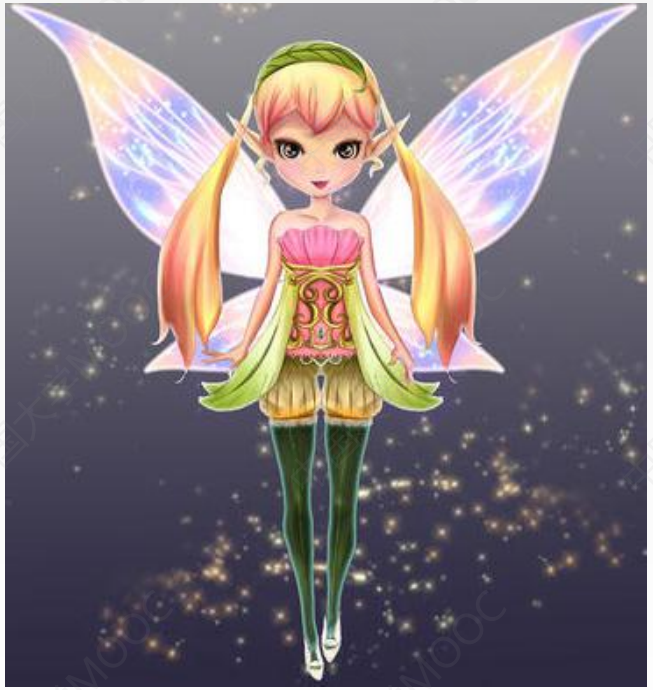
会话和进程组有以下一些特点：

1. 一个会话可以有一个控制终端，建立与控制终端连接的会话首进程被称为控制进程
2. 一个会话中的几个进程组可被分成一个前台进程组和几个后台进程组
3. 如果一个会话有一个控制终端，则它有一个前台进程组。
4. 无论何时键入终端的中断键（Ctrl+C）、退出键（Ctrl+\），都会将中断信号、退出信号发送给前台进程组的所有进程，
5. 如果终端检测到调制解调器（或网络）已经断开连接，则将挂断信号发送给控制进程（会话首进程）。

通常，在登录时，将自动建立控制终端，我们不必关心。

如果希望程序能读写控制终端，则可以打开文件/dev/tty。如果程序没有控制终端，则打开此设备失败。

守护进程



- 守护进程（Daemon）也称为精灵进程，是一类独立于控制终端，运行在后台，执行日常事务的特殊进程。通常在系统自举时启动，关闭时终止。
- 守护进程周期性地执行某种任务或等待处理某些发生的事件，Linux的大多数服务器就是用守护进程实现的。

常见守护进程： 日志进程syslogd、 web服务器httpd、 数据库服务器mysqld

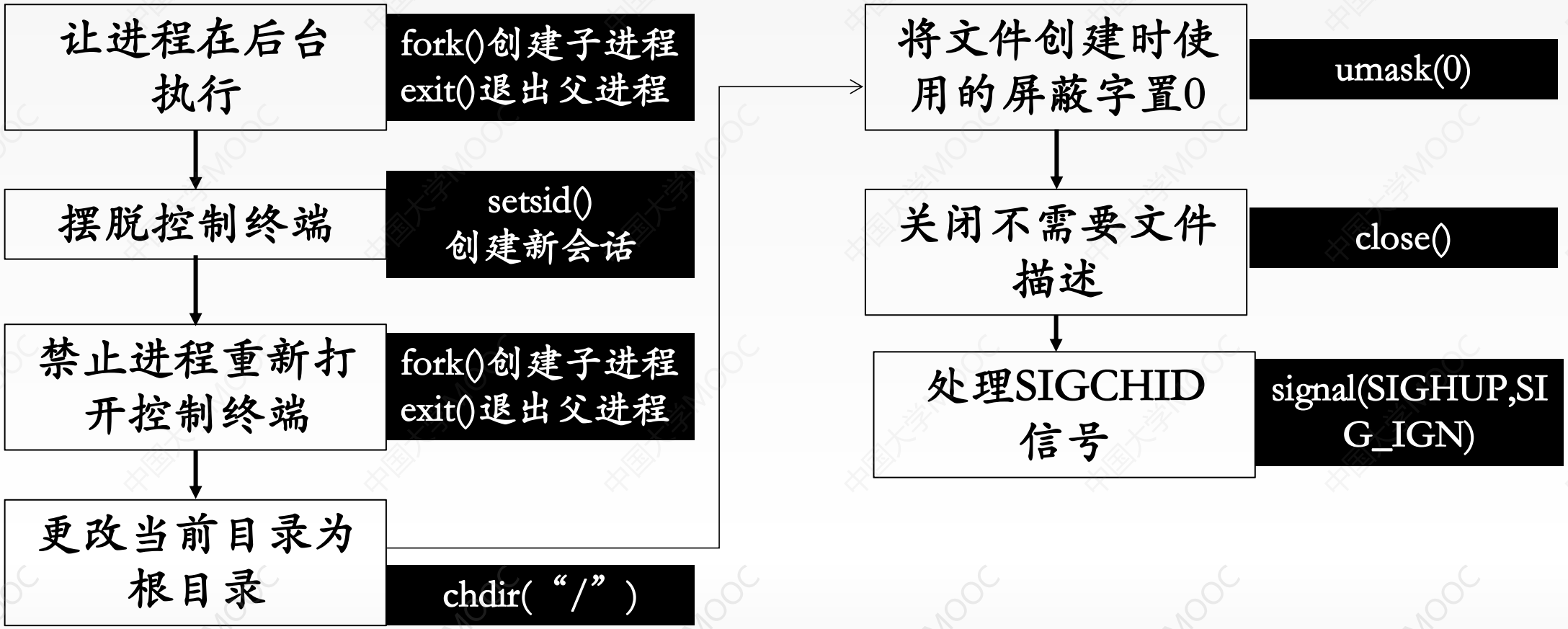
```
hr@hr-VirtualBox:~$ ps -axj
Warning: bad ps syntax, perhaps a bogus '-'? See http://procps.sf.net/faq.html
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
    0     1     1     1 ?           -1 Ss       0   0:02 /sbin/init
    0     2     0     0 ?           -1 S        0   0:00 [kthreadd]
    2     3     0     0 ?           -1 S        0   0:00 [ksoftirqd/0]
    2     4     0     0 ?           -1 S        0   0:00 [kworker/0:0]
    2     5     0     0 ?           -1 S        0   0:00 [kworker/u:0]
    2     6     0     0 ?           -1 S        0   0:00 [migration/0]
    2     7     0     0 ?           -1 S        0   0:00 [watchdog/0]
    2     8     0     0 ?           -1 S<       0   0:00 [cpuset]
    2     9     0     0 ?           -1 S<       0   0:00 [khelper]
    2    10     0     0 ?           -1 S        0   0:00 [kdevtmpfs]
    2    11     0     0 ?           -1 S<       0   0:00 [netns]
    2    12     0     0 ?           -1 S        0   0:00 [sync_supers]
    2    13     0     0 ?           -1 S        0   0:00 [bdi-default]
    2    14     0     0 ?           -1 S<       0   0:00 [kintegrityd]
    2    15     0     0 ?           -1 S<       0   0:00 [kblockd]
    2    16     0     0 ?           -1 S<       0   0:00 [ata_sff]
    2    17     0     0 ?           -1 S        0   0:00 [khubd]
    2    18     0     0 ?           -1 S<       0   0:00 [md]
```

守护进程举例

- init进程，它是内核在自举时启动的用户层命令，它是系统守护进程，负责启动各运行层次特定的系统服务。
- inetd守护进程负责侦听系统网络接口，取得来自网络的对各种网络服务进程的请求。
- cron守护进程在指定的日期和时间执行指定的命令。

守护进程的编程设计思路

创建守护进程，主要是避免产生不必要的交互。



总结

- **进程组**：一个或多个进程的集合，常与同一作业关联。
- **会话**：一个或多个进程组的集合。
- **控制终端**：与会话关联，实现交互，本地终端设备或远程伪终端设备。
- **前台进程和后台进程**：是否与用户交互。
- **守护进程**：后台运行，独立于控制终端，执行日常事务的特殊进程。
- **守护进程创建**：主要是避免产生不必要的交互。

谢谢大家！