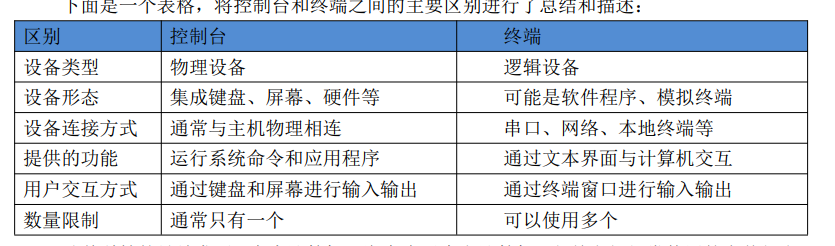
# 7.1 终端和控制台（历史演变）

早期

终端= 键盘+显示器 （个人更多使用便宜的 tty终端设备（teleTYpewriter， 电传打印机，将信息输出到白纸上）

控制器（cnosole: 与主机计算物理单元相连，是集成屏幕、键盘、指示灯和其他硬件组成的设备。功能是允许用户直接与计算机进行交互， 输入命令和数据， 查看计算机的输出和结果



逐渐发展：

控制台现在指的是系统对的主控制台或管理终端，用于系统的配置、管理和故障诊断。Linux系统为例，控制台通常由虚拟终端扮演

伪终端（pseudo terminal） 则提供一种通过网络或本地终端与另一个进程进行通信的机制。例如：

通过ssh协议连接到远程服务器，

包括ubuntu系统图形界面所使用的终端同样也是伪终端 具体原理：

通过终端设备文件和进程间通信来模拟真实终端的输入输出功能。 用户在终端模拟器中输入的命令和操作都会被转发到后台 Shell 进程中执行， Shell 进程的输出也会返回到终端模拟器中显示

# 7.2 进程组和会话

# 7.2.1 进程组

一组相关进程的集合，具有相同的进程组ID(PGID)。

一个进城组通常由一个父进程和若干个子进程组成。（父进程创建子进程时， 可以将子进程放入与父进程相同的进程组中， 也可以将子进程放入其他进程组中）

进程组的 ID 由系统分配，它是一个非负整数， 通常等于进程组中的第一个进程的进程 ID

进程可以使用系统调用 setpgid()来设置自己的进程组 ID， 使用 getpgid()获取自己所属的进程组ID

ps -afo pid,pgid,cmd or pstree -p 用于显示进程之前的关系。

进程组的作用：

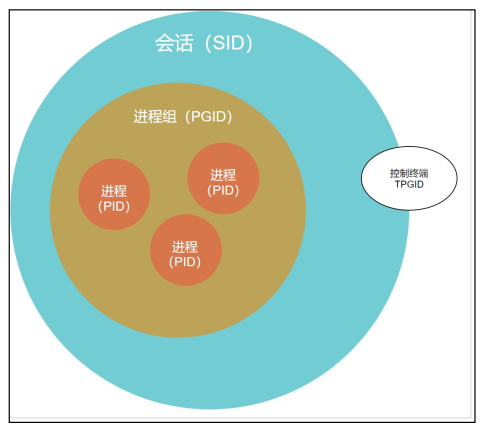
* 提供进程间通信的机制，进程组的所有进程可以相互发送信号
* 控制进程的运行和管理 父进程使用 waitpad() wait 等系统调用等待子进程退出，kill命令来终止进程运行。
* 终端控制 一个进程组在前台运行时， 它会占用控制终端，并能够接收键盘输入。 在后台运行的进程组不会占用控制终端， 但可以接收信号和输出信息
* 向进程组发消息，所有属于该进程组的进程都会收到消息。

## 7.2.2 会话（session）

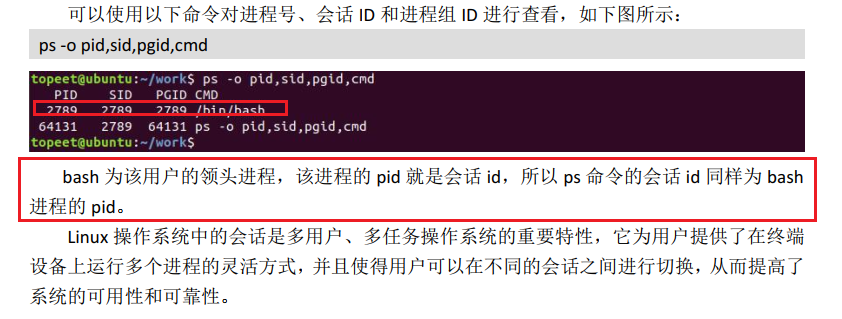
一组相关的进程集合，它们在同一个终端设备上运行， 并由同一个进程启动， 共享同一个控制终端

在一个会话中， 至少存在一个进程组， 而一个进程组可以属于多个会话。 通常情况下， 会话的领头进程（session leader） 是第一个启动的进程， 一般为 shell， 它的进程 ID（PID） 被用作该会话的会话 ID（SID） 。

一个会话可以包含多个进程组， 其中每个进程组都由同一个进程领头， 包含多个相互关联的进程。 进程组中的进程通常会共享同一个终端设备。 进程组 ID（PGID） 与会话 ID（SID） 可以相同， 也可以不同。



会话中，只有领头进程能够直接与终端设备交互，其他进程需要通过与领头进程通信，从而间接与终端设备交互。

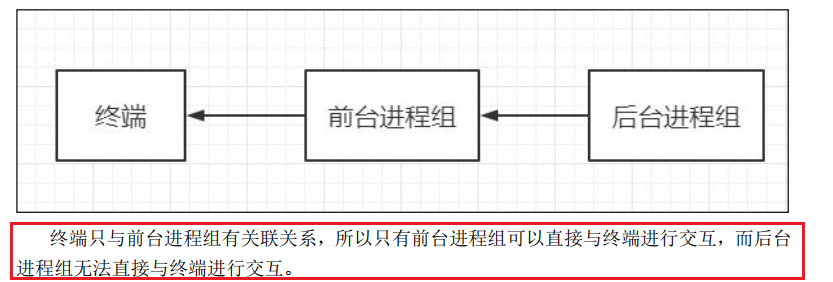


# 7.3 前台进程和后台进程

进程组可以被分为前台进程组（foreground process group）和后台进程组（background process group）

前台进程组是指当前正在与用户交互的进程组， 通常是通过终端输入命令启动的进程组

后台进程组是指在终端不需要与用户进行交互的进程组， 通常是通过在命令行末尾加上"&"符号启动的进程组



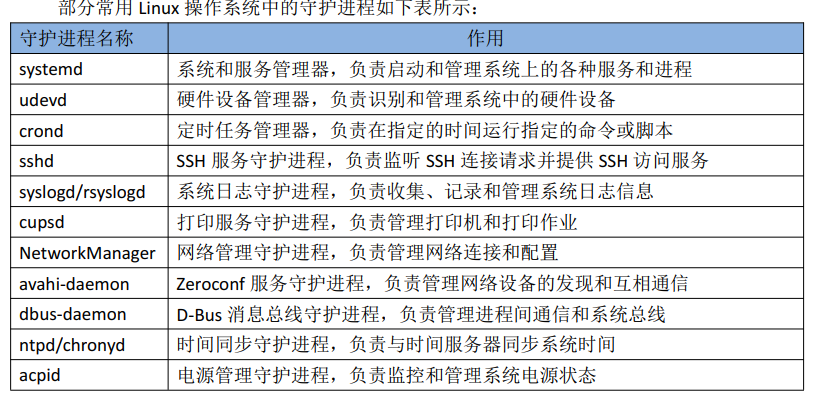
前台进城组->后台进城组 ctrl+z 将其暂停，再使用bg进程号码

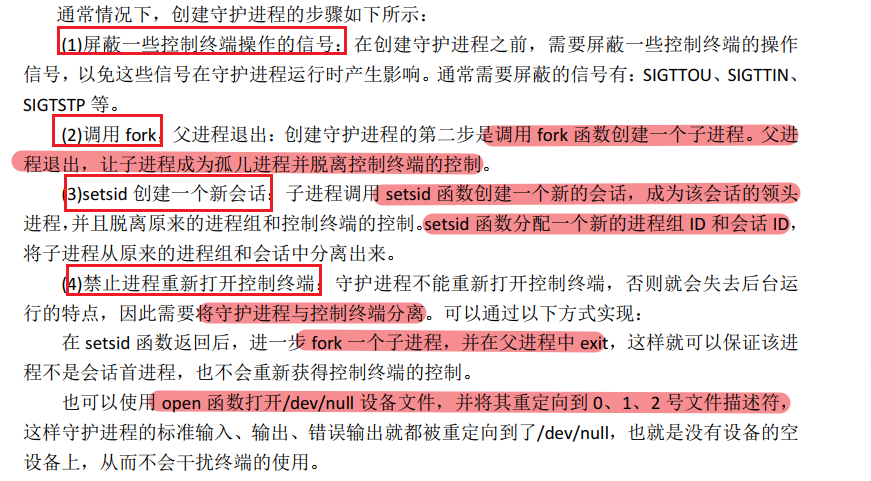
后台进城组->前台进城组 fg

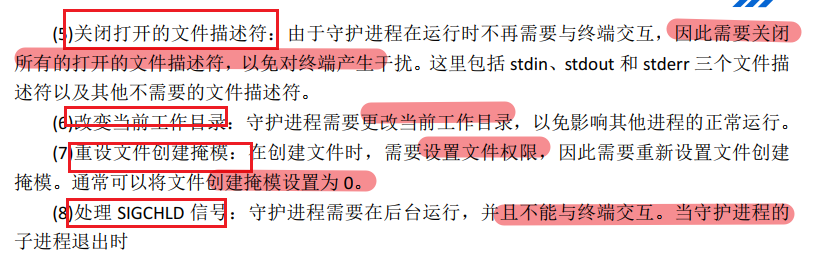
# 7.4 守护进程（daemon）

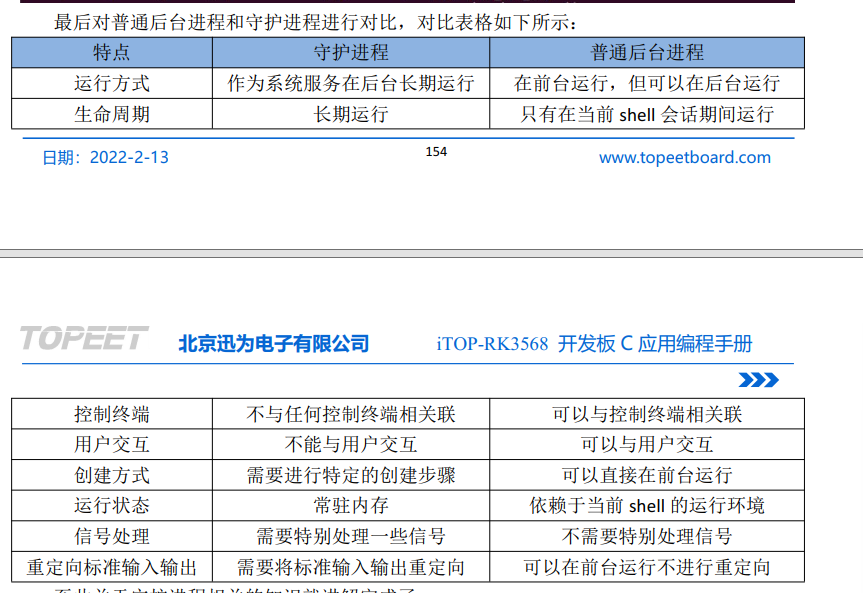
后台进程的一种，作用如其名，不与用户交互不受用户登陆和注销的影响，以root用户身份执行，执行一些系统级别的任务。

特点：后台运行，无人值守，执行系统级别任务、记录日志、系统管理（检测cpu和内存使用情况）



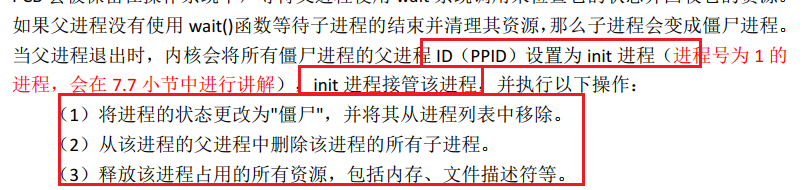






# 7.5 僵尸进程

僵尸进程（zombie） 是指已经执行完毕， 但是父进程没有及时回收其资源的进程



僵尸进程会占用系统的资源， 尤其是进程表中的表项和 PCB 等系统资源。 如果僵尸进程过多， 就会导致进程表耗尽， 从而导致系统崩溃。 因此， 及时清除僵尸进程是非常重要的。

# 7.6 孤儿进程

孤儿进程是指其父进程先于它结束， 从而没有父进程来对它进行资源回收和管理的进程。当进程变为孤儿进程之后， 会被进程 init 进程领养。

僵尸进程则是子进程已经结束了，但父进程却没有回收资源和管理的进程，就是它已经死了但没死透。

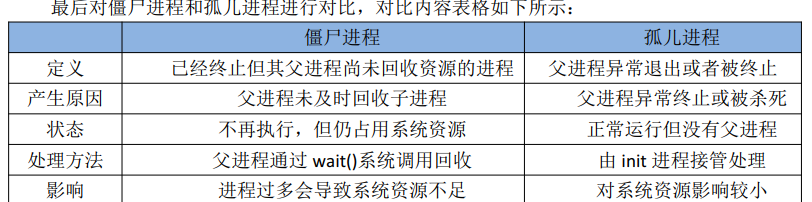
几种情况会导致孤儿进程出现：

父进程终止、父进程崩溃、子进程快于父进程

孤儿进程对系统资源也是有影响的

处理孤儿进程的方法：

* 父进程等待子进程
* 孤儿进程成为进程1（祖先进程）的子进程， 系统自动执行。



# 7.7 1号进程

守护进程创建成功之后其父进程的 PID 为 1， 僵尸进程产生之后会变成 init 的子进程， 从而被清理， 而孤儿进程在产生之后会被 init 进程领养， 同样会成为 init 的子进程。

init 进程是在 Linux 系统中启动所有进程的第一个进程， 它是所有其他进程的祖先进程。所有进程都是由其他进程（即父进程）创建的，只有1号进程除外，它是有内核直接创建的，它的进程号始终为1，是所有其他进程的祖先进程。

init 进程的主要责任是启动系统中的各种进程和服务。 在 Linux 系统中， 系统服务和进程通常由 init 脚本控制。



init 进程实现方式的发展进行简单的概述:

**SysV init- >** **Upstart -> systemd**