守护进程（Daemon）是一种运行在后台的一种特殊的进程，它独立于控制终端并且周期性的执行某种任务或等待处理某些发生的事件。

守护进程的作用：

Linux服务器在启动时需要启动很多系统服务（其实Windows也这样），它们向本地或网络用户提供了Linux的系统功能接口，直接面向应用程序和用户。提供这些服务的程序是由运行在后台的守护进程（daemons）来执行的。

它们常常伴随着Linux系统启动时启动，关闭时关闭。linux系统有很多守护进程，大多数服务器都是用守护进程实现的。另外，某些守护进程还协助完成了很多系统任务，比如负责计划任务的atd和crond、负责打印的lqd等。

实际上守护进程也是有分类的，可以按照它的启动和管理方式来区分，分为独立启动的stand alone和xinetd两类。

所谓的stand alone，从字面上的意思来看就是“独立”的含义。这种类型的守护进程有两大特点，一是可以自行启动运行而不需要利用系统其他机制来管理，二是启动之后会一直占用内存与系统资源。因而这种守护进程拥有了一个非常突出的优点：响应最快。stand alone守护进程非常多，常见的apache、mysql等都是。

至于xinetd则是一种比较新型的守护进程。它由一个统一的stand alone守护进程来负责唤起。这个特殊的守护进程还有一个好听的名字——super daemon。之所以会引入这种机制，就是因为stand alone会一直占用内存和资源显得很浪费。所以一些喜欢精打细算的人就提出来按需分配这种概念。也就是说，当没有客户端要求的时候，xinetd类型的守护进程都是未启动，待有客户端要求服务时，super daemon才会去唤醒具体的xinetd守护进程。但是这种按需分配的机制的致命缺点就是不能及时响应。但是优点也非常鲜明。其一，由于super daemon负责唤醒各项服务，因此可以赋予super daemon安全管控的机制，这就类似网络防火墙的功能了；其二，也是它的设计初衷，即客户端的联机结束后就关闭，不会一直占用系统资源。

大多数Linux发行版会将所有stand alone守护进程的启动脚本都放置在/etc/init.d/目录下，这是一个公认的目录。而CentOS实际上是放在了/etc/rc.d/init.d/目录下了，而/etc/init.d只是它的一个符号连接。大家在记忆的时候，只要记住公认目录即可，那些发行版自己耍的小聪明就不要理会了。

1. 创建子进程，父进程退出：

这是编写守护进程的第一步，由于守护进程是脱离终端的，因此完成第一步后就会在shell终端里造成一个程序已经运行完毕的假象。之后的所有工作在子进程中完成，而用户在shell终端里则可以执行其他命令，从而在形式上做到了与控制终端脱离。实现的语句如下：if(pid=fork()){exit(0);}是父进程就结束，然后子进程继续执行。

1. 在子进程中创建新的会话（脱离控制终端）：

这步是创建守护进程中最重要的一步，虽然实现起来很简单，但是它的意义非常重要，在这里使用的是系统函数setsid（）来创建一个新的会话，并且担任该会话组的组长。在这里有两个概念需要解释一下，进程组合会话期。

进程组：是一个或多个进程的集合。进程组有进程组ID来唯一标识。除了进程号（PID）之外，进程组ID也是一个进程的必备属性。每个进程组都有一个组长进程，其组长进程的进程号等于进程组ID。且该进程组ID不会因组长进程的退出而受到影响。

会话周期：会话期是一个或者多个进程的集合。通常一个会话开始于用户的登录，终止于用户的退出，在此期间该用户运行的所有进程都属于这个会话期。

1. 改变当前目录

使用fork创建的子进程继承了父进程的当前的工作目录。由于在进程运行中，当前目录所在的文件系统是不能卸载的，这对以后的使用会造成诸多的麻烦。因此，通常的做法是让根目录”/”作为守护进程的当前工作目录。这样就可以避免上述的问题。如有特殊的需求，也可以把当前工作目录换成其他的路径。改变工作目录的方法是使用chdir函数。

1. 重设文件权限掩码

文件权限掩码：是指屏蔽掉文件权限中的对应位。例如，有个文件权限掩码是050，它就屏蔽了文件组拥有者的可读与可执行权限（对应二进制为，rwx, 101）。由于fork函数创建的子进程继承了父进程的文件权限掩码，这就给子进程使用文件带来了诸多的麻烦。因此，把文件权限掩码设置为0（即，不屏蔽任何权限），可以增强该守护进程的灵活性。设置文件权限掩码的函数是umask。通常的使用方法为umask(0)。

1. 关闭文件描述符

用fork创建的子进程也会从父进程那里继承一些已经打开了的文件。这些被打开的文件可能永远不会被守护进程读写，但它们一样消耗系统资源，而且可能导致所在的文件系统无法卸载。在使用setsid调用之后，守护进程已经与所属的控制终端失去了联系，因此从终端输入的字符不可能达到守护进程，守护进程中用常规方法（如printf）输出的字符也不可能在终端上显示出来。所以，文件描述符为0、1、2（即，标准输入、标准输出、标准错误输出）的三个文件已经失去了存在的价值，也应该关闭。

(6) 守护进程退出处理

当用户需要外部停止守护进程时，通常使用kill命令停止该守护进程。所以，守护进程中需要编码来实现kill发出的signal信号处理，达到进程正常退出。