Linux 守护进程简介

概述：本文主要介绍 Linux 的守护进程。文章可以分为四部分，第一部分简单介绍守护进程的基本概念；第二部分介绍守护进程的工作方式；第三部分介绍一下 Redhat Linux 中的守护进程管理工具；第四部分介绍如何使用 nohup 命令创建简单的守护进程。

本文以 CentOS5.4 Linux 为测试系统，所有实例均来自该测试系统。文中所述内容适用于 Redhat Linux，但并不适用于所有版本的 Linux。

1. 守护进程的基本概念

Linux 或者 Unix 操作系统中，在系统引导的时候会开启很多服务，这些服务就叫做守护进程。为了增加灵活性，root 用户可以选择系统开启的模式，这些模式叫做运行级，每一种运行级以一定的方式配置系统。

守护进程是脱离于终端并且在后台运行的进程。守护进程脱离终端是为了避免进程在执行过程中产生的信息在任何终端上显示，另外进程也不会被任何终端所产生的信息所打断。

守护进程，也就是通常说的 Daemon 进程，是 Linux 中的后台服务进程。它是一个生存期较长的进程，通常独立于控制终端并且周期性地执行某种任务或等待处理某些发生的事件。守护进程常常在系统引导装入时启动，在系统关闭时终止。Linux 系统有很多守护进程，大多数服务都是通过守护进程实现的。同时，守护进程还能完成许多系统任务。例如，计划任务进程 crond、http 进程 httpd 等(这里的结尾字母 d 就是 Daemon 的意思)。

在 Linux 中，每一个系统与用户进行交流的界面称为终端，每一个从此终端开始运行的进程都会依附于这个终端，这个终端就称为这些进程的控制终端，当控制终端被关闭时，相应的进程都会自动关闭。但是守护进程却能够突破这种限制，它从被执行时开始运转，直到整个系统被关闭时才退出。如果想让某个进程不因为终端或其它的变化而受到影响，那么必须把这个进程变成一个守护进程。

2. 守护进程的工作方式

守护进程的工作方式有两种，分别为独立运行（stand-alone）的工作方式和 xinetd 工作方式。下面分别介绍这两种工作方式。

2.1. 独立运行的守护进程

独立运行的守护进程由 init 脚本负责管理，所有独立运行的守护进程的 init 脚本存放在/etc/rc.d/init.d 目录下。系统服务都是独立运行的守护进程，如 syslogd 和 cron 等。独立运行的守护进程工作方式被称为：stand-alone，它是 Unix 传统的 C/S 模式的访问模式。服务器（即，进程）监听（Listen）在一个特定的端口上等待客户端的联机。如果客户端产生一个连接请求，守护进程就创建（Fork）一个子服务器（或，子进程）响应这个连接，而主服务器继续监听，并维持含有多个子服务器的子服务器池来等待下一个客户端请求。stand-alone 模式的工作原理如图 1 所示。

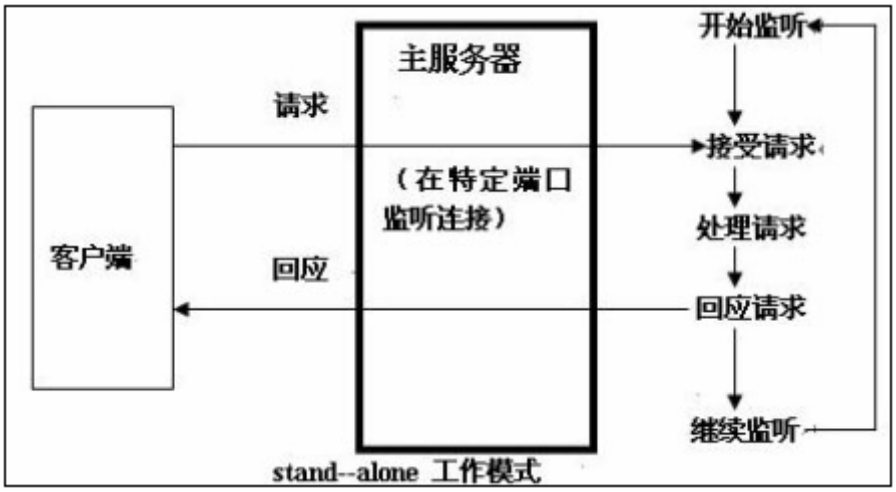

图 1 stand-alone 工作模式

工作在 stand-alone 模式下的网络服务有 route、gated。另外还有大家最熟悉的 Web 服务器 Apache 和邮件服务器 Sendmail、域名服务器 Bind。在这些负载很大服务器上，都预先创建了子服务器，这样可以提高服务速度。在 Linux 系统中处于 stand-alone 工作模式的服务由/etc/rc.d/ rcN.d（N 是数字，表示系统的运行级）目录中的符号链接文件启动。

2.2. xinetd 模式

从守护进程的概念可以看出，对于系统所要提供的每一种服务，都必须运行一个监听某个端口的守护进程，这通常意味着资源浪费。为了解决这个问题，Linux 引入了“网络守护进程服务”的概念。Redhat Linux 使用的网络守护进程是 xinetd（eXtended Internet Daemon）。和 stand-alone 模式相比 xinetd 模式也称 Internet Super-Server（超级服务器）。xinetd 进程能够同时监听多个指定的端口，在接受用户请求时，它能够根据用户请求端口的情况，启动不同的网络服务进程来处理这些用户请求。我们可以把 xinetd 进程看成一个管理启动服务的管理服务器，它决定把一个客户请求交给哪个程序处理，然后启动相应的守护进程，当处

理完客户请求后，再关闭守护进程。xinetd 模式工作原理如图 2 所示。

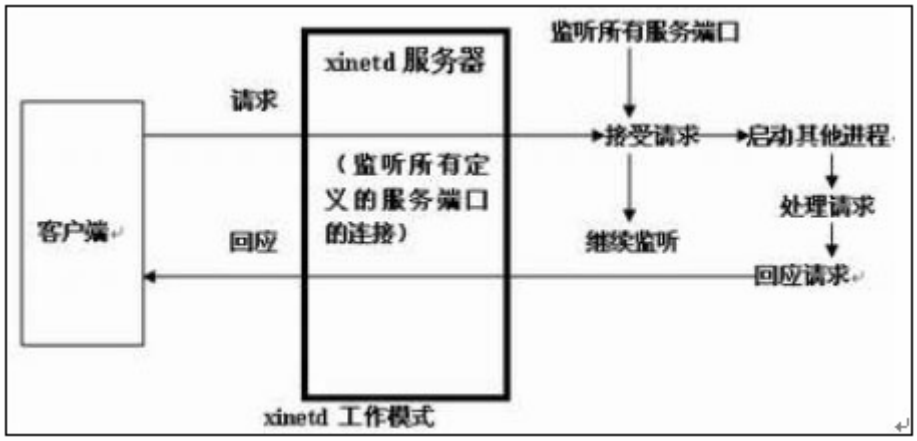

图 2 xinetd 工作模式

stand-alone 工作模式中，系统必使每一个网络服务器进程都监听其服务器端口。相比之下，运行单个 xinetd 进程就可以同时监听所有服务端口，这样就降低了系统开销，节省了系统资源。但是对于访问量大、经常出现并发访问，xinetd 进程需要频繁启动对应的网络服务进程，反而会导致系统性能下降。

一般来说系统一些负载高的服务如 sendmail、httpd 服务是单独启动的。xinetd 守护进程也采取 stand-alone 工作模式。而其它服务类型都可以使用 xinetd 超级服务器管理。查看系统当前运行的守护进程可以使用命令 pstree。

3. Redhat Linux 的守护进程管理工具

Redhat Linux 提供了三种不同的守护进程管理工具：system-config-services、ntsysv、chkconfig。我们可以根据具体需要灵活运用。

3.1. system-config-services 工具

system-config-services（系统服务配置工具）是一个图形化应用程序，它显示了每项服务的描述，每项服务是否在系统引导时启动（运行级 3、4、5），以及允许我们启动、停止或重新启动/etc/rc.d/init.d 目录中的哪些系统服务，允许在需要时启动哪些 xinetd 服务。要从桌面启动服务配置工具，可以通过单击面板上的“系统”-“管理”-“服务器设置”-“服务”，即可打开图形化的管理界面。在终端 shell 命令行下输入命令“system-config-services”也可打开图形化的管理界面。如图 3 和图 4 所示，分别显示了“服务配置”管理界面中“后台服务(S)”标签和“按需服务(O)”标签的内容。通过图 3 和图 4 中关于“后台服务(S)”和“按需服务(O)”的描述，可以让我们对守护进程的 stand-alone 模式和 xinetd 模式有更深层次的理解。

图 3 服务配置界面—后台服务(S)标签

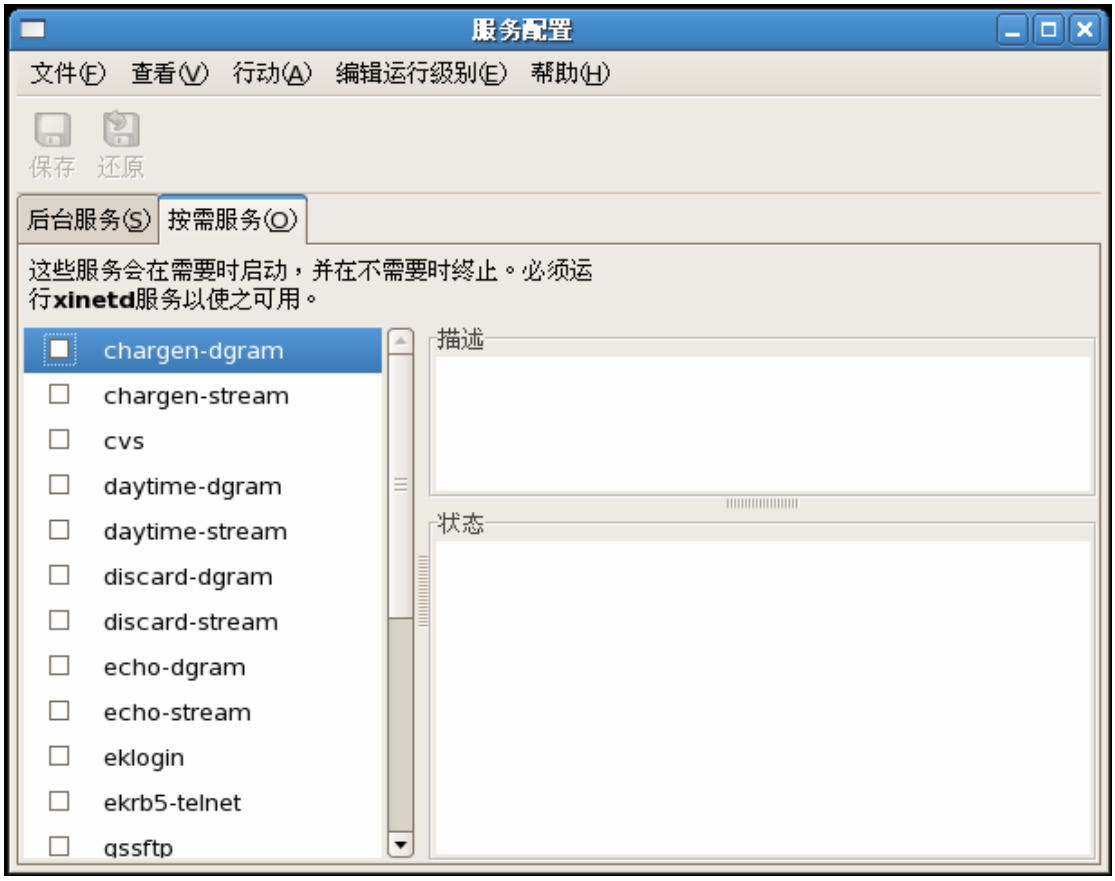

图 4 服务配置界面—按需服务(0)标签

system-config-services 列出了/etc/rc.d/init.d 目录下的服务和由 xinetd 控制的服务。由于图形化控制界面比较直观，我们就不再介绍如何使用这个工具了。

3.2. ntsysv 工具

ntsysv 工具为管理系统服务项在指定的运行级初始化时是否自动启动提供了简单的界面。我们也可以使用 ntsysv 来设置由 xinetd 管理的服务是否在需要时能够启动。单独的 ntsysv 命令只会修改服务在系统当前的运行级初始化时的启动状况。使用上下箭头来上下查看列表。使用空格键来选择或取消选择服务。要在服务列表和“确定”、“取消”按钮间切换，使用 Tab 键。“*”标明某项服务被设为该运行级下自动启动。按 F1 键会弹出每项服务的简短描述。如图 5 所示，这是单独运行命令“ntsysv”后终端中显示的界面。

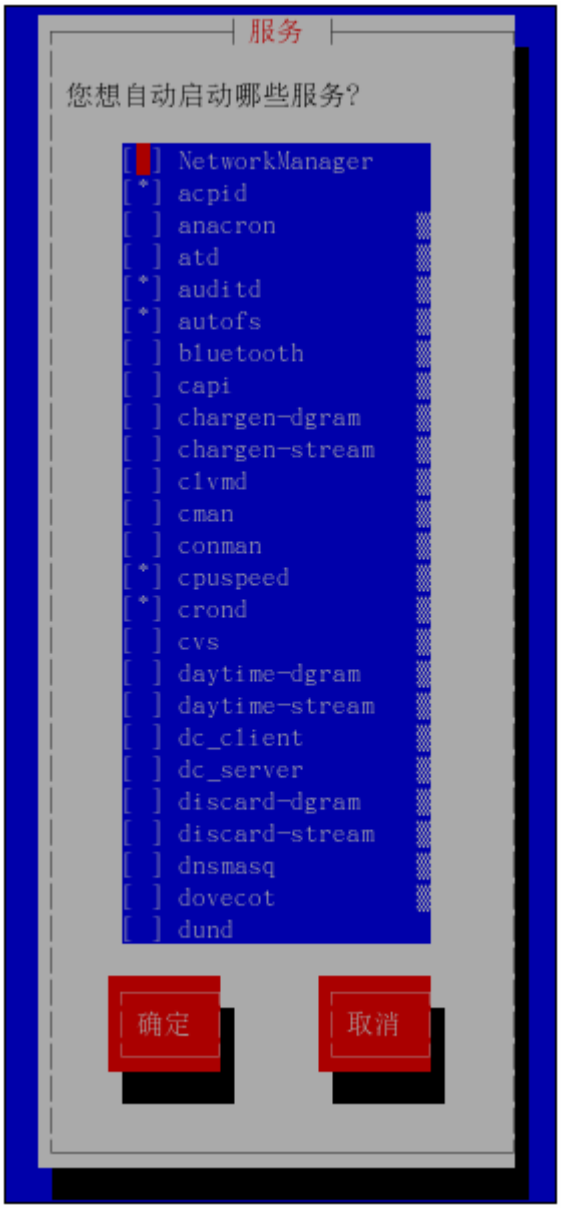

图 5 ntsysv 提供的配置界面

要配置不同的运行级，我们可以使用“--level”选项来指定一个或多个运行级。例如，命令“ntsysv --level 345”配置运行级 3、4 和 5，显示的配置界面与图 5 相同，不同的是这里对服务所做的配置都将同时应用到运行级 3、4 和 5 中。

3.3. chkconfig 工具

关于如何使用 chkconfig 工具，请参考文档《Redhat Linux 中用 chkconfig 管理系统服务项》，这里就不再叙述。

4. 用 nohup 命令创建守护进程

在 Unix/Linux 中，普通进程可以用“&”符号放到后台运行，如果启动该进程的控制终端退出了，则该进程随即终止，因为普通进程不同于守护进程。要实现守护进程，一种方法是按守护进程的规则去编程，比较麻烦；另一种方法是仍然用普通方法编程，然后用 nohup 命令去启动程序。用 nonup 启动的程序所创建的进程，在控制终端退出后，进程仍在继续运行，起到了守护进程的作用（虽然它不是严格意义上的守护进程）。

nohup 命令的语法：nohup program [> file] &

使用 nohup 命令，如果没有为程序的标准输出设置重定向，程序的标准输出被自动重定向到当前目录的 nohup.out 文件中，起到了 log 的作用。如果当前目录的 nohup.out 文件不可写，输出重定向到$HOME/nohup.out 文件中。如果没有文件能创建或打开以用于追加，那么命令不可调用。

示例：用 nohup 运行 checkpublish.sh 脚本，如图 6 所示。

```
[root@localhost ~]# nohup /usr/local/turbocms/bin/checkpublish.sh &
[1] 9028
[root@localhost ~]# nohup: appending output to `nohup.out'

[root@localhost ~]# ls
anaconda-ks.cfg  install.log.link_soft  test1          testDir
Desktop          install.log.syslog     test1SoftLink  testDirSL
install.log      nohup.out              test2          testdu
```

图 6 用 nohup 命令运行 checkpublish.sh 脚本

实例：用 nohup 运行 checkpublish.sh 脚本并设置输出重定向，如图 7 所示。

```
[root@localhost ~]# nohup /usr/local/turbocms/bin/checkpublish.sh > /tmp/log/20100304.log &
[1] 9233
[root@localhost ~]# ls
anaconda-ks.cfg  Desktop  install.log  install.log.link_soft  install.log.syslog
[root@localhost ~]# ls /tmp/log/
20100304.log
```

图 7 用 nohup 命令运行 checkpublish.sh 脚本并设置输出重定向

编者注：写了这么多东西，不知道质量如何，希望和大家多多交流共同进步，我还是一个 Linux 的初学者，难免很多谬误，希望高手赐教指正，以期不断进步。