Linux 后台进程

作者：Anileh
原文链接：https://ld246.com/article/1664091851453
来源网站：链滴

Linux守护进程

在Linux服务器实际应用中，经常会有需要长时间执行的任务。

这类任务若在前台运行，用户无法进行其他操作或者断开与服务器的连接，否则任务将被中止，便需守护进程的工具。

本文简要讲解 nohup, tumx， screen ， supervisor 四种方式

nohup

nohup 命令的作用就是让后台工作在离开操作终端时，也能够正确地在后台执行

命令

nohup Command [Arg …] [&]

参数说明

- Command：要执行的命令。
- Arg：一些参数，可以指定输出文件。
- &：让命令在后台执行，终端退出后命令仍旧执行。

实例：

以下命令在后台执行 /usr/local/目录下的 demo.sh 脚本：

nohup /root/demo.sh &

demo.sh 的内容如下：

pwd

在终端如果看到以下输出说明运行成功：

/usr/local

如果要停止运行，你需要使用以下命令查找到 nohup 运行脚本到 PID，然后使用 kill 命令来删除：

ps -aux | grep "demo.sh"

参数说明：

- **a** : 显示所有程序
- **u** : 以用户为主的格式来显示
- **x** : 显示所有程序，不区分终端机

另外也可以使用如下方式：

```
ps -def | grep "demo.sh   # 查找此文件的进程
kill -9  进程号PID        # kill PID 来删除。
```

tumx

nohup虽然可以在保持在后台一直运行，但是不太便于管理，尤其是对于耗时极长的大数据项目。

为大家介绍一位新朋友——tmux，“Terminal MultipleXer”，意思是“终端复用器”，可以让人通过一个窗口操作多个会话。

检查安装

CentOS/RedHat/Fedora

sudo yum -y install tmux

tmux 依赖于 libevent 库和 ncurses 库，此过程会自动安装上。

Ubuntu/Debian

sudo apt-get install tmux

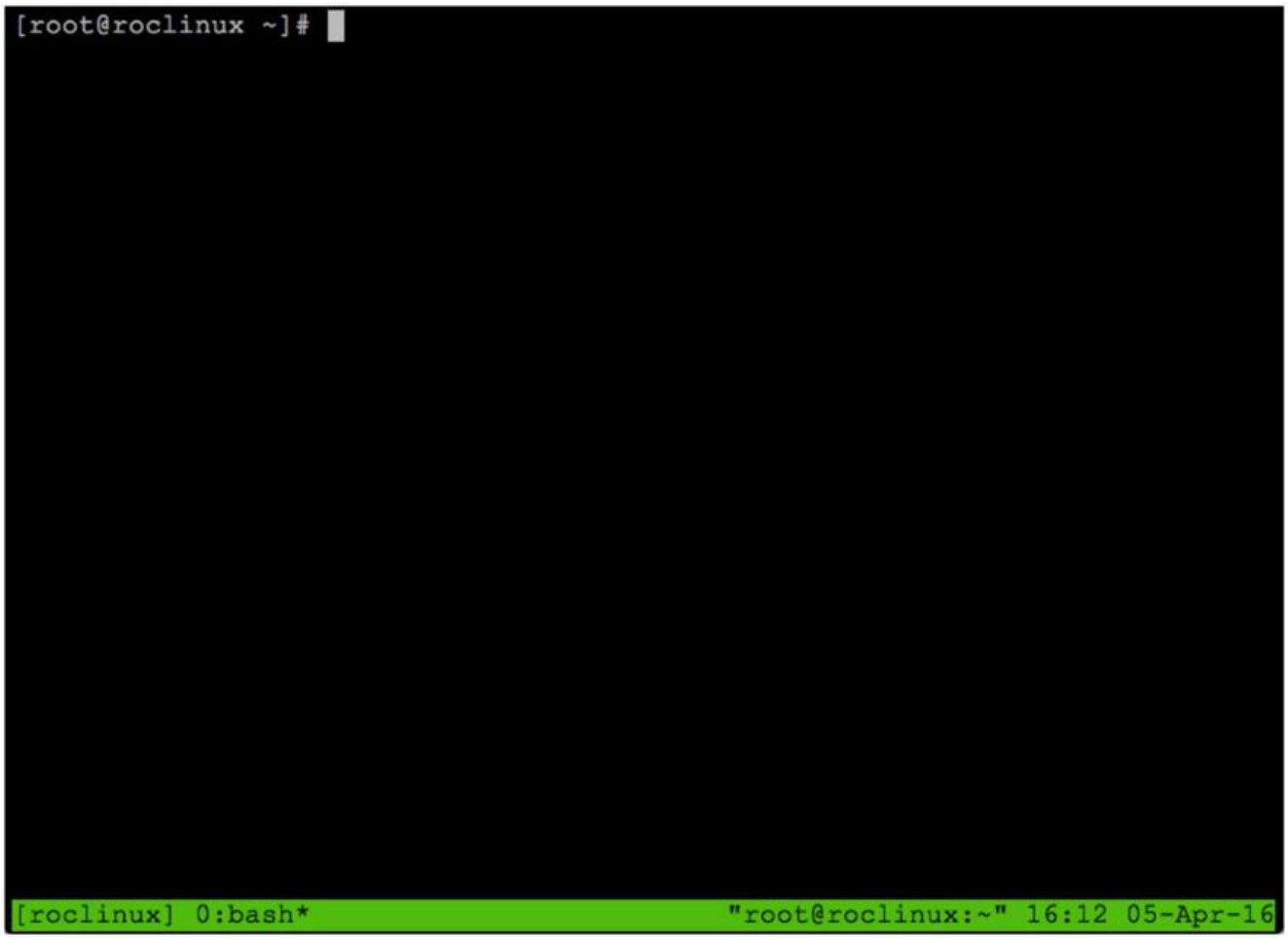
基本命令

新建会话

sudo tmux new -s name

-s： session 的缩写，顾名思义，启动了一个全新的 tmux 会话（tmux session），并且把这个会话名叫作 name。

在这里起名为 roclinux，tmux 会在会话中创建一个窗口（tmux window），就是图中的 0：bash，* 表示当前的活跃窗口。



原文链接：Linux 后台进程

多窗口

命令：按 Ctrl+B 组合键，再单独按一下 C 键

最底部出现了一个 1：bash，如图，我们在 tmux session 中又创建了一个窗口。

多窗口切换

命令：在按下 Ctrl+B 组合键后，按相应数字键，就可以切换到相应的窗口了

假如我们要切换到 0：bash 这个窗口：按 Ctrl+B 组合键，按数字 0 键。

挂起会话并保持活跃状态

命令：

sudo watch -n second free

意义：每隔 second 秒更新一次内存使用状态，如果不输入 Ctrl+C，则永远不会退出。

退出当前 session

这个操作并不会导致如上的 watch 命令终止

命令：输入组合键 Ctrl+B，然后输入字母d

结果：tmux 环境消失，只有一行提示 [detached]：表明已经切断了办公电脑和刚才那个 tmux 之间桥梁

[root@roclinux ~]# tmux new -s roclinux

[detached]

重新连接会话

```
sudo tmux a -t name
```

name：要连接的 session 的名字

如果有多个 session，可以列出它们：

```
sudo tmux ls
# 结果
roclinux: 2 windows (created Fri Jan 22 16:30:13 2016) [130x36]
```

screen

screen 可以提供从单个 SSH 会话中使用多个 shell 窗口（会话）。
当会话被分离或网络中断时，screen 会话中启动的进程仍将运行，你可以随时重新连接到 screen 会
。

检查安装

screen 在一些流行的发行版上已经预安装了，检查是否已经安装。

```
sudo screen -v
# 输出：Screen version 4.00.03 (FAU)
```

如果在 Linux 中还没有 screen，可以使用系统提供的包管理器安装。

CentOS/RedHat/Fedora

```
sudo yum -y install screen
```

Ubuntu/Debian

```
sudo apt-get -y install screen
```

基本命令

启动 screen 会话

```
sudo screen -S name
```

name ： 这个 session 的名字，替换为对你会话有意义的名字

分离screen 会话

要从当前的 screen 会话中分离，你可以按下 Ctrl-A 和 d。
所有的 screen 会话仍将是活跃的，你之后可以随时重新连接。

重新连接到 screen 会话

```
sudo screen -r args
```

如果你有多个 screen 会话，你可以用 ls 参数列出它们。

```
sudo screen -ls`

There are screens on:
7880.session    (Detached)
7934.session2   (Detached)
7907.session1   (Detached)
3 Sockets in /var/run/screen/S-root
```

如上可见有三个活跃的 screen 会话，如果想要还原 “session2” 会话，可以执行：

```
sudo screen -r 7934
# 或
sudo screen -r session2
```

中止 screen 会话

有几种方法来中止 screen 会话。你可以按下 Ctrl+d，或者在命令行中使用 exit 命令。

要查看 screen 命令所有有用的功能，你可以查看 screen 的 man 手册。

```
sudo man screen

NAME
screen - screen manager with VT100/ANSI terminal emulation

SYNOPSIS
screen [ -options ] [ cmd [ args ] ]
screen -r [[pid.]tty[.host]]
screen -r sessionowner/[[pid.]tty[.host]]
```

supervisor

Supervisor 是用 Python 开发的一套通用的进程管理程序，能将一个普通的命令行进程变为后台daemon，并监控进程状态，异常退出时能自动重启。

原理: 通过fork/exec的方式把被管理的进程当作 supervisor 的子进程来启动，这样只要在superviso的配置文件中，把要管理的进程的可执行文件的路径写进去即可。

当子进程挂掉时，父进程可以准确获取子进程挂掉的信息，可以选择是否自己启动和报警；还可以为 supervisord 或者每个子进程，设置一个非root的user，这个user就可以管理它对应的进程。

检查安装

提醒： Supervisor 官方版目前只能运行在 Python 2.4 以上版本，但是还无法运行在 Python 3 上，不已经有一个 Python 3 的移植版 supervisor-py3k

CentOS/RedHat/Fedora

```
sudo yum -y install supervisor
```

Ubuntu/Debian

sudo apt-get -y install supervisor

pip安装

pip install supervisor

配置文件

supervisor 配置文件：/etc/supervisor/supervisord.conf

注：supervisor的配置文件默认是不全的，不过在大部分默认的情况下，上面说的基本功能已经满足。

子进程配置文件路径：/etc/supervisord.d/

注：默认子进程配置文件为ini格式，可在supervisor主配置文件中修改。

本人服务器的文件目录: /etc/supervisord.conf

supervisor.conf 配置文件说明：

```
[unix_http_server]
file=/tmp/supervisor.sock   ;UNIX socket 文件，supervisorctl 会使用
;chmod=0700                 ;socket文件的mode，默认是0700
;chown=nobody:nogroup       ;socket文件的owner，格式：uid:gid

;[inet_http_server]         ;HTTP服务器，提供web管理界面
;port=127.0.0.1:9001        ;Web管理后台运行的IP和端口，如果开放到公网，需要注意安全性
;username=user              ;登录管理后台的用户名
;password=123               ;登录管理后台的密码

[supervisord]
logfile=/tmp/supervisord.log ;日志文件，默认是 $CWD/supervisord.log
logfile_maxbytes=50MB       ;日志文件大小，超出会rotate，默认 50MB，如果设成0，表示不限大小
logfile_backups=10          ;日志文件保留备份数量默认10，设为0表示不备份
loglevel=info               ;日志级别，默认info，其它: debug,warn,trace
pidfile=/tmp/supervisord.pid ;pid 文件
nodaemon=false              ;是否在前台启动，默认是false，即以 daemon 的方式启动
minfds=1024                 ;可以打开的文件描述符的最小值，默认 1024
minprocs=200                ;可以打开的进程数的最小值，默认 200

[supervisorctl]
serverurl=unix:///tmp/supervisor.sock ;通过UNIX socket连接supervisord，路径与unix_http_server部分的file一致
;serverurl=http://127.0.0.1:9001 ; 通过HTTP的方式连接supervisord

; [program:xx]是被管理的进程配置参数，xx是进程的名称
[program:xx]
command=/opt/apache-tomcat-8.0.35/bin/catalina.sh run  ; 程序启动命令
autostart=true       ; 在supervisord启动的时候也自动启动
startsecs=10         ; 启动10秒后没有异常退出，就表示进程正常启动了，默认为1秒
autorestart=true     ; 程序退出后自动重启,可选值：[unexpected,true,false]，默认为unexpected表示进程意外杀死后才重启
startretries=3       ; 启动失败自动重试次数，默认是3
user=tomcat          ; 用哪个用户启动进程，默认是root
```

```
priority=999        ; 进程启动优先级，默认999，值小的优先启动
redirect_stderr=true ; 把stderr重定向到stdout，默认false
stdout_logfile_maxbytes=20MB  ; stdout 日志文件大小，默认50MB
stdout_logfile_backups = 20   ; stdout 日志文件备份数，默认是10
; stdout 日志文件，需要注意当指定目录不存在时无法正常启动，所以需要手动创建目录（supervisod 会自动创建日志文件）
stdout_logfile=/opt/apache-tomcat-8.0.35/logs/catalina.out
stopasgroup=false     ;默认为false,进程被杀死时，是否向这个进程组发送stop信号，包括子进程
killasgroup=false     ;默认为false，向进程组发送kill信号，包括子进程

;包含其它配置文件
[include]
files = relative/directory/*.ini    ;可以指定一个或多个以.ini结束的配置文件
```

子进程配置文件说明：

给需要管理的子进程(程序)编写一个配置文件，放在/etc/supervisor.d/目录下，以.ini结尾，此处新建log.ini

参数见上面的 **supervisor.conf 配置文件说明**

```
#redis数据定时插入数据库
[program:dbtimer]
directory=/home/www/blog
command=php index.php swoole/cron/index
autostart=true
autorestart=true
stderr_logfile=/home/www/blog/timerin.log
stdout_logfile=/home/www/blog/timerout.log

#swoole启动
[program:swoolerun]
directory=/home/www/blog
command=php swooleRun.php
autostart=true
autorestart=true
stderr_logfile=/home/www/blog/swoolein.log
stdout_logfile=/home/www/blog/swooleout.log
```

基本使用

supervisor 启动

```
// 启动supervisor并加载默认配置文件
sudo systemctl start supervisord.service

//将supervisor加入开机启动项
sudo systemctl enable supervisord.service
```

常用命令

```
supervisorctl status        // 查看所有进程的状态
supervisorctl update        // 配置文件修改后使用该命令加载新的配置
supervisorctl reload        // 重新启动配置中的所有程序
supervisorctl restart <application name>  // 重启指定应用
```

```
supervisorctl stop <application name>     // 停止指定应用
supervisorctl start <application name>    // 启动指定应用
```

注： 把 <application name> 换成 all 可以管理配置中的所有进程。直接输入supervisorctl 进入supervisorctl 的shell交互界面，此时上面的命令不带supervisorctl 可直接使用