# Supervisor进程管理工具

2024年8月28日　　18:29

# Supervisor

## 一、简介

supervisor 是一个用 python 语言编写的进程管理工具，它可以很方便的监听、启动、停止、重启一个或多个进程。当一个进程意外被杀死,supervisor 监听到进程死后，可以很方便的让进程自动恢复，不再需要程序员或系统管理员自己编写代码来控制。

主要原理就是它有一个主进程，要管理的脚本都是它的子进程。通过这个主进程来管理这些脚本

**1、supervisord**

- supervisor 的服务端:运行 supervisor 时会启动一个进程 supervisord
- 负责启动所管理的进程，并将所管理的进程作为自己的子进程来启动
- 可以在所管理的进程出现崩溃时自动启动

**2、supervisorctl**

- 用于控制supervisor的一个工具
- supervisor的客户端:supervisorctl是命令行管理工具，可以用命令来进行子进程的管理

**3、echo supervisord conf**

- echo supervisord conf是supervisor的一个内置命令，用于生成supervisor 配置文件的模版
- 使用

  echo_supervisord_conf > supervisord.conf

  这行命令会将 supervisor 的模版配置文件输出到名为 supervisord.conf的文件中

## 二、安装

**更新软件包列表:**

```
sudo apt update
```

**安装 supervisor:**

```
sudo apt install supervisor
```

**查看配置文件**

```
cd /etc/supervisor
```

**启动 supervisor**

```
sudo systemctl start supervisor
```

**验证启动**

```
sudo systemctl status supervisor
```

## 三、配置详解

Supervisor 安装后会在系统中生成两个配置文件

主进程配置文件: /etc/supervisor/supervisord.conf
子进程配置文件: /etc/supervisor/conf.d 子进程可能有多个配置文件

### 1、默认主配置文件

```
; supervisor config file

[unix_http_server]
file=/var/run/supervisor.sock   ; (the path to the socket file)
chmod=0700                       ; sockef file mode (default 0700)

[supervisord]
logfile=/var/log/supervisor/supervisord.log ; (main log file;default $CWD/supervisord.log)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
childlogdir=/var/log/supervisor            ; ('AUTO' child log dir, default $TEMP)

; the below section must remain in the config file for RPC
; (supervisorctl/web interface) to work, additional interfaces may be
; added by defining them in separate rpcinterface: sections
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///var/run/supervisor.sock ; use a unix:// URL  for a unix socket

; The [include] section can just contain the "files" setting.  This
; setting can list multiple files (separated by whitespace or
; newlines).  It can also contain wildcards.  The filenames are
; interpreted as relative to this file.  Included files *cannot*
; include files themselves.

[include]
files = /etc/supervisor/conf.d/*.conf
```

**[unix_http_server]（与客户端交流的方式）**

- 这部分设置 http 服务器监听的 UNIX_domain_socket（在同一台计算机上用于通信socket，避免使用网络进行通信，更快）
- file：指向 UNIX domain socket 所在的位置
- chmod:启动时更改 supervisor.sock 的权限

**[supervisord]:**

与 supervisord 有关的全局配置需要在这部分设

- logfile: 指向记录 supervisord 进程的 log 文件**（记录服务器日志文件的地方）**
- pidfile:pidfile 保存子进程的路径**（记录子进程id）**
- childlogdir:子进程 log 目录设为 AUTO 的 log 目录**（记录子进程的日志文件）**

**[supervisorctl]:**

- serverurl:进入supervisord 的URL,对于 UNIX domain sockets应设为 unix:///absolute/path/to/file.sock

**[include]:（导入脚本的配置）**

- 如果配置文件包含该部分，则该部分必须包含一个 files 键
- files:包含一个或多个文件，这里包含了/etc/supervisor/conf.d/目录下所有的 .conf 文件，可以在该目录下增加我们自己的配置文件在该配置文件中增加[program:x]部分，用来运行我们自己的程序

## 2、子进程配置文件

在/etc/supervisor/conf.d 路径下配置子进程

**supervisor_monitor.conf**

```
[program:ActionMonitor]
command=python -u ActionMonitor.py
user=root
autostart=true
autorestart=true
stderr_logfile=/脚本错误日志输出路径/action_monitor.err.log
stdout_logfile=/脚本信息日志输出路径/action_monitor.out.log
directory=/脚本所在路径
```

[program:x]:
（1）配置文件必须包括至少一个 program，x是program 名称，必须写上，不能为空
（2）command:包含一个命令，当这个 program 启动时执行
（3）directory:执行子进程时 supervisord 暂时切换到该目录
（4）user:用户名
（5）autostart: 如果设置为 true，当 Supervisor启动时，这个程序也会自动启动。
（6）autorestart: 如果设置为 true，当这个程序退出时，Supervisor 会自动重启它。
可以设置为 unexpected，这样只有在程序异常退出(退出状态码非 0)时才会重启。
（7）redirect stderr:如果是 true，则进程的 stderr 输出被发送回其stdout 文件描述符上的 supervisord
（9）stdout logfile:将进程 stdout 输出到指定文件
（10）stderr_logfile:将进程 stderr 输出到指定文件
（11）startsecs:程序被认为是成功启动所需的秒数。也就是进程从STARING 状态转换到 RUNNING 状态 program 所需要保持运行的时间(单位:秒)
（12）startretries:启动失败后的最大重试次数
（13）priority:程序启动的优先级，默认为 999，数值越小优先级越高
（14）numprocs:指定启动多少个进程实例

## 四、基本命令

**通过配置文件启动 supervisor**

```
supervisord -c supervisor.conf
```

**查看状态**

```
supervisorctl status
```

**控制 Supervisor 管理的进程**

```
#查看状态
supervisorctl status

#启动进程
#启动所有进程:
supervisorctl start all

#启动指定进程(例如:myprogram):
supervisorctl start myprogram

#停止进程
#停止所有进程:
supervisorctl stop all

#停止指定进程(例如:myprogram):
supervisorctl stop myprogram

#重启进程
#重启所有进程:
supervisorctl restart all

#重启指定进程(例如:myprogram):
supervisorctl restart myprogram
```

**重新加载配置**

```
#当更改了 supervisor 的配置文件后，需要重新加载配置才能生效supervisorctlreread
#更新进程组应用更新
supervisorctl update
```

**停止 Supervisor**

```
supervisorctl shutdown
```