Building a python daemon process

June 18, 2012 / Gavin Jackson

daemon | init scripts | logging | programming | python

This blog post is a quick introduction to building a python daemon process. This sample builds a true Unix daemon process, with full logging and an optional init script (which has been tested on SLES 11 SP2 but should work on other Redhat based distributions).

A Unix daemon has some fundamental characteristics (it's a lot more than just a background process), these are listed in *PEP 3143 Standard Daemon Process Library* http://www.python.org/dev/peps/pep-3143/#correct-daemon-behaviour.

Cron may be a suitable alternative to this approach, however a daemon allows you to have a single, long lived process that can maintain state, allows you to send process signals to, a guarantee that you only have one instance in execution (which can be a gotcha with poorly written cron scripts that don't implement lock files), by modifying the INIT INFO section of the init script you can also control where in the system boot up the daemon commences execution.

Step 1 - Required Libraries

Install the Python Daemon package from (http://pypi.python.org/pypi/python-daemon/) - note that you will also need an earlier version of the python lockfile (version 0.8) to work (http://pypi.python.org/pypi/lockfile/0.9.1).

Step 2 - Sample Code

```
# To kick off the script, run the following from the python directory:
```

```
# PYTHONPATH=`pwd` python testdaemon.py start
```

```
#standard python libs
import logging
```

```
import time

#third party libs
```

`from daemon import runner`

`class App():`

```

 def __init__(self):
 self.stdin_path = '/dev/null'
 self.stdout_path = '/dev/tty'
 self.stderr_path = '/dev/tty'
```

****`self.pidfile_path = '/var/run/testdaemon/testdaemon.pid'`

```
 self.pidfile_timeout = 5
```

```
 def run(self):

 while True:
 #Main code goes here ...
```

```
 #Note that logger level needs to be set to logging.DEBUG before this shows up in the logs
```

`logger.debug("Debug message")`

```
 logger.info("Info message")
```

```
 logger.warn("Warning message")
```

`logger.error("Error message")`

```
 time.sleep(10)
```

```
app = App()
logger = logging.getLogger("DaemonLog")
logger.setLevel(logging.INFO)
formatter = logging.Formatter("%(asctime)s - %(name)s - %(levelname)s - %(message)s")
```

****`handler = logging.FileHandler("/var/log/testdaemon/testdaemon.log")`

```
handler.setFormatter(formatter)
logger.addHandler(handler)
```

```
daemon_runner = runner.DaemonRunner(app)
```

```
#This ensures that the logger file handle does not get closed during daemonization
daemon_runner.daemon_context.files_preserve=[handler.stream]
daemon_runner.do_action()
```

Save this script under **/usr/share/testdaemon/testdaemon.py**

You need to ensure that the **/var/log/testdaemon** and **/var/run/testdaemon** folders exist and the user running the daemon process has write permissions.

Step 3 - Init Script

You can run the script by running **python testdaemon.py start**

This will spawn the daemon process. You can stop it by running **daemon.py stop**

The following init script will allow you to start and stop the daemon on system startup and shutdown respectively:

```
#! /bin/bash
```

`# Copyright (c) 1996-2012 My Company.`

```
# All rights reserved.
#

# Author: Bob Bobson, 2012
```

`#`

```
# Please send feedback to bob@bob.com
#

# /etc/init.d/testdaemon
#

### BEGIN INIT INFO
# Provides: testdaemon
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Short-Description: Test daemon process
# Description: Runs up the test daemon process
### END INIT INFO
```

```
# Activate the python virtual environment
```

****`. /path_to_virtualenv/activate`

```
case "$1" in
 start)
 echo "Starting server"

 # Start the daemon
```

`python``/usr/share/testdaemon/testdaemon.py``start`

```
 ;;
 stop)
 echo "Stopping server"

 # Stop the daemon
```

`python``/usr/share/testdaemon/testdaemon.py``stop`

`;;`

```
 restart)
 echo "Restarting server"
```

```
 python /usr/share/testdaemon/testdaemon.py restart
```

`;;`

```
 *)

 # Refuse to do other stuff
 echo "Usage: /etc/init.d/testdaemon.sh {start|stop|restart}"
 exit 1
 ;;
esac
```

```
exit 0
```

Note that this is using python virtualenv to use an isolated python environment (please refer to my previous blog entry here).

Save file in **/etc/init.d/testdaemon**

chmod u+x **/etc/init.d/testdaemon**

Enable using **chkconfig testdaemon on**

Check that it is enabled **chkconfig --list**

…

testdaemon 0:off 1:off 2:off **3:on** 4:off **5:on** 6:off

…

Kick it off by running **service testdaemon start**

You will now see it running in the background:

```
ps -aux|grep testdaemon
root 15225 0.0 0.0 64304 5948 ? S 13:10 0:00 python /usr/share/testdaemon/testdaemon.py start
```

Step 4 - Example Usage

```
(pyenv-zenkai)rama:/usr/share/testdaemon # python ./testdaemon.py start
(pyenv-zenkai)rama:/usr/share/testdaemon # started with pid 14565
```

```
(pyenv-zenkai)rama:/usr/share/testdaemon # cat /var/log/testdaemon/testdaemon.log
2012-06-18 12:57:22,512 - DaemonLog - INFO - Info message
2012-06-18 12:57:22,512 - DaemonLog - WARNING - Warning message
2012-06-18 12:57:22,512 - DaemonLog - ERROR - Error message
2012-06-18 12:58:19,770 - DaemonLog - INFO - Info message
2012-06-18 12:58:19,771 - DaemonLog - WARNING - Warning message
2012-06-18 12:58:19,771 - DaemonLog - ERROR - Error message
2012-06-18 12:58:29,779 - DaemonLog - INFO - Info message
2012-06-18 12:58:29,779 - DaemonLog - WARNING - Warning message
2012-06-18 12:58:29,780 - DaemonLog - ERROR - Error message
```

...

```
(pyenv-zenkai)rama:/usr/share/testdaemon # python ./testdaemon.py stop
Terminating on signal 15
(pyenv-zenkai)rama:/usr/share/testdaemon #
```