看门狗（Watchdog）使Linux设备从异常运行中快速恢复

Published on May 3, 2022 UTC by Wang Ye <https://wangye.org>

看门狗（Watchdog）应该是计算机世界中较为形象的一个动物化的特性，就好像光猫和鼠标一样，在那个混沌初开的计算机时代，技术专家和极客们总会给冷冰冰的机器赋予身边活物的灵性。看门狗的功能简要的来说就是侦测到设备异常并恢复设备状态，这里的恢复一般是重启，就好像公司IT部门在接到设备报障电话总是会问一句："您尝试重启设备了吗？"，确实，重启能解决绝大部分故障，重启后的设备也能工作在较好的状态，随着时间的推移，待故障再次发生的时候怎么办？再重启一下呗。如果无法确认故障的原因或者无法解决故障，那么重启确实很好，但是这仍然需要有人值守，手动重启。看门狗的出现就是为了解决这个痛点，看门狗每间隔一段时间（Interval）会检测设备状态，检测的方式就是一系列测试（Test），如果测试不通过则自动重启。

第一次接触到看门狗（Watchdog）功能还是在树莓派（Raspberry Pi），当时硬件清单说明上指出树莓派支持硬件看门狗，在系统卡死无响应的时候会自动重启，以为这个功能是内置的，所以一直没有管他，直到有一天某个配置较低的ARM卡片机设备，由于我的Docker容器配置错误导致容器疯狂重启进程，消耗大量系统资源，导致我无法登入SSH并重启，我才又想起这只狗。

1 使用场景

可能有读者会有疑问，我们究竟需要不需要看门狗呢，我认为分情况讨论：

有人值守的设备可以不需要，试想一下你在用电脑执行某项操作，可能峰值期间占用了大量CPU，这很正常，比如视频渲染等操作。如果你的看门狗配置不当，它发现CPU被大量占用，触发重启，那你的工作也就白做了。
无人值守的设备就有必要考虑了，试想一下没有人愿意大半夜被叫醒重启设备，尤其是现在大量部署的物联网IoT设备，这些设备数量大、范围广，如果个别或者几个设备出现问题，人工操作成本必然较高。

说个题外话，之前看到某小哥调试一部物联网设备，其中用Python写了一个外挂，外挂的功能就是喂狗，这里的狗当然是看门狗，因为此物联网设备的看门狗监督物联网某个进程，一旦发现进程没有响应或者终止了就自动重启，这给调试这个进程带来一定麻烦，所以需要不停的喂狗来欺骗看门狗。

另外看门狗也被大量应用于高端的路由器等常见网络设备中，其可以在温度过高或者网络断开的时候自动重启。

如无例外说明，以下操作均需要root权限，请使用root账户或者sudo提权，请记住，看门狗的配置存在风险，建议先在开发机器上调试通过后再部署到生产机器上，该文章仅供参考，本人不对该文章描述不当或者阁下操作不当造成的损失负责。

2 安装部署

2.1 通过发行版软件源安装

通过发行版软件源安装应该是最为方便的一种形式，避免了从源代码安装所需要安装的依赖和编译的等待时间，而且后期也能方便的升级或者卸载，例如下面介绍几种常见发行版的软件源安装方式：

```
# Redhat/Fedora/Centos
dnf install watchdog

# Ubuntu/Debian
apt-get install watchdog

# Arch
pacman -Sy watchdog
```

安装完成后，请立即使用下面的命令禁用（disable）它：

```
systemctl disable watchdog
```

这一步很重要，因为使用未经过测试的配置策略，并使看门狗处于活动状态是一件很危险的事情，比如接下来错误的配置可能导致活动的看门狗立即触发设备重启，而且重启后错误的配置又会继续导致看门狗重启设备，一次又一次，很遗憾你的设备再也不能通过正常的途径启动了，出现这种情况会使得系统恢复变得棘手，尤其是远程设备出现这种问题，所以务必先禁止看门狗服务，待我们调试正确后再小心的启动它。

2.2 内核装载软件狗设备（softdog）[可选]

之前提到过树莓派内置了硬件看门狗，如果驱动正常的话可以通过下面的命令找到它：

```
# 查看内核模块
lsmod | grep _wdt
# 或者 lsmod | grep wdog
# 查看看门狗设备路径是否正常
ls -l /dev/watchdog
```

如果一切正常的话你将会看到系统有所输出，这里不单单适用于树莓派，你可以检查需要安装看门狗的设备是否已经具备了相关的驱动模块，如果没有，那么我们可以使用系统内置的软件狗（softdog）作为替代。

Tips：常见的看门狗模块有 `iTCO_wdt`、`bcm2708_wdog`、`bcm2835_wdt`等等，可以在`lsmod`时候灵活选择，一般`/dev/watchdog`这个路径存在的话就说明看门狗驱动模块是正常加载的。

软件狗设备模块需要先装载进内核中：

```
modprobe softdog
# 使用下面命令检查是否装载成功
lsmod | grep softdog
ls -l /dev/watchdog
```

确定`/dev/watchdog`路径存在后，我们内核中就有了看门狗设备，接下来我们

需要配置刚刚安装的看门狗服务以支持此设备。

首先修改配置文件/etc/watchdog.conf，增加以下内容：

```
watchdog-device = /dev/watchdog
```

然后修改watchdog服务的启动配置/etc/default/watchdog或者/etc/sysconfig/watchdog（不同发行版文件位置不同），并增加以下内容：

```
watchdog_module="softdog"
```

至此这个阶段算是完成了，我们接着下面的章节。

3 配置看门狗

如果不告诉看门狗如何"看门"，那么这只狗也没有存在的意义，所以接下来我们需要进行简单的配置并设定一系列测试规则，对于系统当前的状态进行测试（Test），一旦测试失败则重启设备。

3.1 通用配置

修改配置文件/etc/watchdog.conf，修改或增加以下内容：

```
realtime = yes
priority = 1
interval = 3
```

其中realtime表示watchdog作为实时服务，priority表示进程级别，1表示最高，这样即使系统无响应或者进程抢占资源导致繁忙，系统也会优先执行看门狗服务进程任务，避免看门狗抢占不到执行资源而饿死。interval表示检查并测试系统状态的时间间隔，默认是1，表示1秒，可以适当延长，但不可以过长，过长的时间间隔容易让看门狗不能及时发现故障，从而延长了系统恢复的几率和时间。

3.2 测试规则

看门狗提供了一系列内置的测试规则供实际使用，通过合理正确的配置可以响应并应对绝大多数系统故障，这里举一些例子供实际参考：

① CPU负载

设想一下，如果你的设备中的程序意外出现了降低性能的故障，或者有不怀好意者故意捣乱，运行了逻辑炸弹，导致系统资源被大量消耗，这时候可能你连SSH都不能正常登录上去，那么远程排除故障就不现实，这时候我们需要的是系统能够自动重启，不要再执行无所谓的计算消耗。

为实现这一目标，我们需要告诉看门狗，当系统负载达到多少的时候直接重启设备，一般这个负载可以通过`uptime`命令查询，比如下面的执行结果：

```
# uptime
  17:08:02 up  1:48,  0 users,  load average: 0.05, 0.04, 0.01
```

其中`load average`后面的三组数字分别表示当前系统在过去的1分钟、5分钟和15分钟的负载，正常系统空闲情况下应该低于1，如果系统长时间运行高强度计算，那么这个负载会变得很高，再加上设备配置不行，这时候系统基本卡死无法操作了，我们需要让看门狗能检查这个指标。

修改配置文件`/etc/watchdog.conf`，增加或者修改参数`max-load-1`（1分钟）、`max-load-5`（5分钟）和`max-load-15`（15分钟）的值，比如对于入门配置的VPS或者树莓派这样的低配置主机，推荐值如下：

```
max-load-1  = 150
max-load-5  = 100
max-load-15  = 50
```

当然具体还要看实际情况，对于多核高配机器，这里的数值也应当相应增加，以避免看门狗误判。

② 内存消耗

过高的内存消耗除了会导致频繁的swap交换还会影响程序执行，部分程序可能因为OOM（Out-of-memory）错误而异常，这样也容易造成业务中断，还有一种情况就是第三方程序设计缺陷导致内存泄漏，又无法进行修复的，可以让看门狗检测到过多的内存占用时立即重启设备。

同样，配置项位于`/etc/watchdog.conf`，具体如下：

```
min-memory  = 1
allocatable-memory = 1
```

其中`min-memory`指的是最小的内存页数，通常情况下在x86硬件，每页是4kB。

`allocatable-memory`类似于`min-memory`，指的是可分配的内存页数，同样在x86硬件，每页是4kB。

③ 网络情况

对于一些特殊的需求，比如曾经有位网友反馈，他上网通过PPPoE拨号随机分配IP，但是其中偶然会分配到一个IP无法访问某个资源，这时候就需要重新拨号，当然拨号是由路由器负责，一般粗暴的重启路由器也能实现重新拨号的功能，所以这项工作也完全可以交给看门狗，让其检测网络情况，发现问题立即重启设备。

同样，配置项位于`/etc/watchdog.conf`，具体如下：

```
ping   = 114.114.114.114
```

使用ping的方式检测网络连通性，一般适用于ping网关，或者网络上某个你需要与之连接的IP地址，比如上面设置114.114.114.114为测试对象，目前来看是可以ping通的，如果测试通过说明两个结论：一是设备已经连接互联网；二是能够正常用114.114.114.114提供的服务。

这种方式还是存在问题，最简单来说，万一哪一天114.114.114.114通过防火墙关闭了ping的功能，那么你的设备就会陷入无限重启中，所以这个方式还是适用于可信目标服务器的ping，也就是你想要ping的目标服务器在你的掌控范围内。

通过检测网卡流量来判断网络连通性，比如`eth0`是我们通常上网的网卡，如果网卡时间段内没有足够流量流入（incoming traffic (RX)），那么看门狗就会重启设备。

```
interface  = eth0
```

这个配置项仍然需要谨慎，设想一下对于一些较为冷门的设备，除非人为通过设置程序保持心跳，否则很有可能出现一段时间没有流入流量，这会让看门狗误判断。

另外我还要多说一句，不建议在云主机或者VPS上配置网络检测项目，这些主机一般虚拟化运行在母机或者ISP网络中，母机和网络并不能确保100% uptime，很有可能会在某段时间内出现没有网络的情况，比如受到DDoS攻击、施工队挖断光缆等等，谁也不希望自己的设备在看门狗的指挥下疯狂重启吧。

④ 文件变化

假设有这么一个程序，这个程序每隔一段时间会向某个文件记录数据，比如写入日志等，如果规定时间内文件没有变化，那么我们判断程序应该崩溃或者失效了，这时候由看门狗触发重启。

同样，配置项位于`/etc/watchdog.conf`，具体例子如下：

```
file   = /var/log/messages
change   = 1407
```

这里file指示的是需要监测的文件路径，change表示间隔时间（分）。这个测试对于我个人来说用的较少。

⑤ 外部程序执行结果

上面介绍了几种常用的测试规则，看门狗的配置文件/etc/watchdog.conf描述也较为详细，如果有想进一步详细了解的朋友可以参考watchdog配置项手册介绍。

这里再补充一个适用于复杂检测测试规则，比如我们需要检测的对象是否成功规则比较复杂，可能需要额外的脚本或者程序进行判断，这时候可以让watchdog运行我们自定义的外部程序，并且由外部程序告知watchdog测试是否通过，配置项目如下：

```
test-binary  = /path/to/your/check_script
test-timeout = 超时的秒数
```

其中/path/to/your/check_script是watchdog运行检测的外部程序，该程序运行完毕后需要进程返回0表示成功，非0表示失败，一般通过exit调用。

3.3 尝试重启前修复

读到这儿，很多朋友会有一个疑惑，难道我们非到了不得不重启设备的地步吗？能否在重启之前再尽力挽救一下系统呢？这里就不得不提到看门狗的另外一项功能，也让大家知道这不是一只只会重启的狗。

watchdog提供了修复程序选项，也就是说该选项配置后，看门狗在测试系统发现问题时不会急于重启，而是先运行外部的修复程序，选项配置如下：

```
repair-binary  = /path/to/your/repair_script
# 最长允许修复程序执行的时间（单位：秒）
repair-timeout = 30
# 在重启前修复程序最多执行的次数
repair-maximum = 3
```

修复程序可以包含一系列操作，比如重启网卡、重启指定服务等等，需要注意的是修复成功后需要进程返回0告知watchdog，这一般通过exit调用实现。

4 测试并启用看门狗

4.1 测试看门狗是否生效

还记得2.1节我们禁用（disable）的看门狗服务吗？先不要急着启用（enable），我们先演习一般测试一下看门狗能否具备正常工作的条件，修改看门狗服务的启动配置文件`/etc/default/watchdog`或者`/etc/sysconfig/watchdog`（不同发行版文件位置不同），修改或者增加下面的配置项：

```
watchdog_options="-v --no-action"
```

这里给看门狗服务程序调用时附加了两个参数，`-v`（Verbose）表示显示详细操作，`--no-action`表示只告警而不执行重启等对应操作，这对于调试很有用。

好了，现在可以安全的启动看门狗服务了，使用下面的命令：

```
systemctl start watchdog

# 或者
service watchdog start
```

服务启动OK后，使用命令`journalctl -f`或者查看`/var/log/syslog`就可以看到看门狗的输出，这时候你可以进行相应的压力测试，看能否触发看门狗的行为。

4.2 正式启用看门狗

看上去运行的很好，Good Job，现在让我们正式让狗持证上岗吧，还记得4.1节的配置`watchdog_options="-v --no-action"`吗，现在需要删除或者注释掉这行，然后重启启动看门狗服务，并且激活（enable）先前被禁用的看门狗服务。

```
systemctl restart watchdog
systemctl enable watchdog
```

5 总结

到这里看门狗技术我就简单的介绍完了，对于IoT物联网设备来说，无人值守是一种常态，设备需要具备一定的自我恢复的能力以保障业务的延续性，另外看门狗技术也是一种运维技巧，合理的运用可以大为减轻运维负担。

参考文献：《Using Watchdog to Always Keep a Machine Running》

External References (Links)

watchdog配置项手册
https://www.crawford-space.co.uk/old_psc/watchdog/watchdog-configure.html
《Using Watchdog to Always Keep a Machine Running》
https://www.supertechcrew.com/watchdog-keeping-system-always-running/