

AllWatch

面向 Linux 的微型进程监控器：目标进程在前台运行，以内核和 TCP 判断健康，按目标进程设置重启策略

手册

版本	0.1.0
提交	`ae9cb8f`
提交日期	2026 年 11 月 24 日
许可证	MIT

离线手册，内容与所给版本和提交的仓库指南一致。以仓库中的文本为准。

目录

概览 **1**
- 在 Debian 和 Ubuntu 上安装 . . . 1
- 在 Fedora、openSUSE Leap 和 Arch Linux 上安装 . . . 1
- 从源代码仓库运行 . . . 2
- 工作原理 . . . 2
- 配置 . . . 2
- 目标进程必须遵守的约定 . . . 3
- 限制 . . . 3
- 许可证 . . . 3
- 致谢 . . . 3

安装并运行 **Debian** 服务 **4**
- 软件包安装的内容 . . . 4
- 安装 . . . 4
- 路径和文件 . . . 4
- 声明一个目标进程 . . . 4
- 无响应的目标进程：重启或告警 . . . 5
- 服务命令 . . . 5
- 在不启动目标进程的情况下检查配置 . . . 5
- 在不停止服务的情况下停止一个目标进程 . . . 5
- 以另一个用户身份运行 . . . 5
- 升级 . . . 6
- 移除和清除 . . . 6
- 其他发行版系列：Fedora、openSUSE Leap 和 Arch Linux . . . 6
- 软件包的已知限制 . . . 6

故障排查 **7**
- 先读取服务状态 . . . 7
- 服务没有启动 . . . 7
- 目标进程没有被重启 . . . 7
- 目标进程被报告为无响应，但它仍在运行 . . . 7
- `--diagnose` 返回 0，但目标进程不健康 . . . 8
- 测试替身不响应 . . . 8
- 日志无限增长 . . . 8
- 监控器重启后出现重复的目标进程 . . . 8
- 日志中没有任何内容 . . . 8

概览

程序按系统语言输出 *--help* 摘要：默认英语，已翻译 *fr*、*de*、*es* 和 *zh*。

AllWatch 是面向 Linux 的微型进程监控器，由一个 Bash 脚本写成（约一百七十行有效代码）。它让每个目标进程在前台启动，记住其 PID，只依据内核和网络给出的事实判断其健康：进程是否仍在 /proc 中，其 TCP 端口是否仍接受连接，以及它结束时留下什么退出码。它按每个目标进程的策略重启目标，并拒绝在一个刚启动就死亡的进程上无限循环。

目标进程是一个黑盒。它不向 AllWatch 注册，不向它发送任何内容，也不需要知道它的存在。

在 **Debian** 和 **Ubuntu** 上安装

Debian 软件包把监控器安装为 /usr/bin/allwatch，同时安装 allwatch.service systemd 单元、示例配置和五种语言的手册页。请从下载页面 取得软件包和 SHA256SUMS，然后运行：

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

公布的验证级别：安装与诊断已在容器中验证，覆盖 Debian 12 和 13，以及 Ubuntu 22.04 和 24.04。Debian 软件包还经过真实环境的检验：作为服务安装在运行 Debian 13 的 Raspberry Pi 4 上，使用真实应用，先运行两小时，再运行六小时（95 个事件，95 个通过，无一失败），随后卸载并清除。服务安装后处于启用状态，配置不声明任何目标进程；请把目标加入 /etc/allwatch/allwatch.conf，然后运行 sudo systemctl restart allwatch。

在不启动目标进程的情况下检查已安装的版本和配置：

```
allwatch --version
allwatch --help
allwatch --diagnose /etc/allwatch/allwatch.conf
```

--diagnose 读取配置、状态目录和日志路径。它不启动也不探测目标进程，因此不能证明某个目标进程健康；它证明监控器能够读取配置并使用其路径。

要移除软件包：

```
sudo apt remove allwatch
sudo apt purge allwatch
```

apt remove 停止服务但保留其启用状态；apt purge 还会移除配置和服务状态。

在 **Fedora**、**openSUSE Leap** 和 **Arch Linux** 上安装

同一个标签还会构建原生软件包：Fedora 的 .rpm、openSUSE Leap 的 .rpm 和 Arch Linux 的 .pkg.tar.zst。从下载页面下载您所用发行版系列的文件和 SHA256SUMS，然后运行对应的命令行（依次为 Fedora、openSUSE Leap、Arch Linux）：

```
sudo dnf install ./allwatch-0.1.0-1.fc44.noarch.rpm
sudo zypper install --allow-unsigned-rpm ./allwatch-0.1.0-1.leap160.noarch.rpm
sudo pacman -U ./allwatch-0.1.0-1-any.pkg.tar.zst
```

公布的验证级别：安装和诊断已在容器中验证，涵盖 Fedora 43、44 和 45、openSUSE Leap 16.0 以及 Arch Linux。这些软件包不作为已在物理服务器上测试过的软件包来介绍。服务已安装但未启动：请先把您的目标添加到 /etc/allwatch/allwatch.conf，再运行 sudo systemctl enable --now allwatch。allwatch --version、--help 和 --diagnose 与上文所示相同。

移除软件包：

```
sudo dnf remove allwatch
sudo zypper remove allwatch
sudo pacman -R allwatch
```

移除后会保留系统用户 allwatch 和 /var/lib/allwatch，以及您修改过的配置（保存为 allwatch.conf.rpmsave 或 allwatch.conf.pacsave）。若要一并清除：

```
sudo userdel allwatch
sudo rm -rf /etc/allwatch /var/lib/allwatch
```

从源代码仓库运行

要求：Linux（监控器读取 /proc）、Bash 5 或更高版本，以及常用核心工具中的 timeout 和 awk。仅测试替身 dummy-target.sh 还需要 setpriv（util-linux）和 OpenBSD nc。

```
git clone https://github.com/RonanDavalan/AllWatch.git
cd AllWatch
./allwatch.sh --version
./allwatch.sh --help
./allwatch.sh allwatch.conf
```

示例配置以标识 demo 在端口 9001 上启动 dummy-target.sh。在第二个终端中查看目标进程：

```
cat /tmp/allwatch-runtime/demo.pid
: > /dev/tcp/127.0.0.1/9001 && echo "port 9001 answers"
```

强制杀死目标进程，观察它以新的 PID 回来（轮询间隔为 5 秒）：

```
kill -9 "$(cat /tmp/allwatch-runtime/demo.pid)"
sleep 6
cat /tmp/allwatch-runtime/demo.pid
```

让目标进程无响应（它仍然存活，但关闭其端口）。AllWatch 会发现、停止并重启它，大约需要十五秒：

```
kill -USR1 "$(cat /tmp/allwatch-runtime/demo.pid)"
sleep 20
grep ' demo ' ~/.local/state/allwatch/allwatch.log | tail -n 6
```

用 Ctrl+C（或 SIGTERM）停止监控器；它会停止其目标进程并清空状态目录：

```
ls -A /tmp/allwatch-runtime
```

工作原理

AllWatch 是父进程，目标进程是子进程。它每隔几秒查看每个目标进程，并把它归入四种状态之一：

- 运行中：进程存在且其端口接受连接。
- 无响应：进程存在但其端口不响应。使用 ON_FREEZE=restart（默认值）时，AllWatch 发送 SIGTERM，等待目标进程停止（5 秒），必要时发送 SIGKILL，然后把目标进程视为失败。使用 ON_FREEZE=alert 时，它不发送信号，只记录一次 ALERTE GEL，继续监视目标进程，并在端口重新响应时记录 RETABLI。
- 崩溃：进程在没有主动停止的情况下消失。AllWatch 读取退出码，并按目标进程的策略决定是否重启。
- 已退出：目标进程结束且其策略要求不再处理，或已达到重启上限。

配置

配置在启动时逐行读取一次。空行和以 # 开头的行会被忽略。引擎参数，每行一个 KEY=value：

- POLL_SECONDS=5：两次检查之间的秒数。
- PROBE_TIMEOUT=3：端口必须响应探测的秒数。
- MAX_RESTARTS=5：在下面窗口内允许的重启次数。
- RESTART_WINDOW=60：该窗口的长度，以秒计。
- ON_FREEZE=restart：无响应的目标进程会被中和并重启；alert 保持目标进程不动，并记录 ALERTE GEL。其他任何值都会在启动时以退出码 1 拒绝。

目标进程，每行一个：

```
id port policy command...
```

- id：短名称，用于状态文件和日志。
- port：目标进程监听的 TCP 端口（在 127.0.0.1 上探测）。

- policy：always、on-failure、transient 之一。
- command...：该行其余部分，按空格拆分（不支持引号），从启动 allwatch.sh 的目录以前台方式运行。

策略：

- always：目标进程每次结束都重启，即使退出码为 0，除非状态目录中存在 <id>.stop 文件。
- on-failure：崩溃或无响应后重启，但在退出码为 0 后保持停止。
- transient：从不重启。

如果目标进程在 RESTART_WINDOW 秒内达到 MAX_RESTARTS 次重启，AllWatch 就停止重启它，记录 ALERTE VERROU_CRITIQUE，并继续处理其他目标进程。

状态目录是 /tmp/allwatch-runtime（可用 ALLWATCH_RUNTIME 覆盖）；日志是一个平面文件，每个事件一行（date id event），位于 ${XDG_STATE_HOME:-$HOME/.local/state}/allwatch/allwatch.log。

目标进程必须遵守的约定

AllWatch 不做任何迁就；违反约定的目标进程才是过错方。

- 前台。目标进程在 AllWatch 启动的进程的前台中运行。它不得转为后台守护进程或二次 fork：否则 AllWatch 会看到第一个进程结束，然后启动一个重复实例。
- 退出码。干净的、被请求的停止以退出码 0 结束。任何失败都以非零退出码结束。
- 一个 **TCP** 端口。每个目标进程拥有一个 TCP 端口，在初始化后立即打开并接受连接。一个存活但在探测超时内不接受连接的目标进程被视为无响应。
- **SIGTERM**。目标进程捕获 SIGTERM，释放其资源，并在 5 秒内以退出码 0 结束。超过这个时间，AllWatch 发送 SIGKILL。

窗口标题、显示服务器和目标进程自身的输出都不参与健康检查。

限制

- 配置不支持热重载：sudo systemctl restart allwatch。
- 不依据窗口标题、X11 或 Wayland 进行检查。
- 不解析应用程序日志来决定是否重启。
- 不监视转为后台守护进程或二次 fork 的目标进程。
- --diagnose 是静态的：它读取配置和路径，不探测目标进程，也不能证明某个目标进程健康。
- --version、--help 和 --diagnose 只在第一个参数的位置被识别。
- 命令不经过 shell 运行：不支持引号、管道、重定向或变量展开。
- 数值配置值和策略名称不做校验。非数值的 POLL_SECONDS 会让监控器以退出码 1 停止；拼错的策略按 always 处理。ON_FREEZE 会校验：未知值会让监控器以退出码 1 停止。
- 平面日志不做轮转。

许可证

MIT。参见 LICENSE。

致谢

- **Ronan Davalan**：架构师和仲裁者。他确定愿景和范围，仲裁并验证规范。
- **Claude Sonnet 5.5**（Anthropic）：代码的唯一作者（监控器和测试替身），于 2026 年 11 月 13 日在三次编码会话中写成。
- **DeepSeek V4.1 Flash 1017**：参考文档、六小时验证测试及测试报告。
- **Gemini 4 Hydrogène**（Google）：关于监控器的设计与思考交流，图标的构想与设计。

参见 CONTRIBUTORS.md。

安装并运行 **Debian** 服务

Debian 软件包把 AllWatch 安装为 systemd 服务。服务在开机时启动，读取一次配置；如果没有声明任何目标进程，它就等待。本章给出软件包的确切路径、文件和命令。

软件包安装的内容

```
/usr/bin/allwatch                          监控器
/usr/bin/allwatch.sh                       兼容符号链接
/usr/lib/systemd/system/allwatch.service   systemd 单元
/etc/allwatch/allwatch.conf                配置
/usr/share/man/man1/allwatch.1.gz          手册页，英语
/usr/share/man/fr/man1/allwatch.1.gz       手册页，法语
/usr/share/man/de/man1/allwatch.1.gz       手册页，德语
/usr/share/man/es/man1/allwatch.1.gz       手册页，西班牙语
/usr/share/man/zh_CN/man1/allwatch.1.gz    手册页，简体中文
/usr/share/doc/allwatch/README.Debian.gz   软件包说明
/usr/share/doc/allwatch/examples/allwatch.conf.example
/usr/share/doc/allwatch/examples/allwatch.conf  源代码仓库的配置文件
/usr/share/doc/allwatch/examples/dummy-target.sh
```

软件包创建系统用户和组 allwatch（主目录 /var/lib/allwatch，shell 为 /usr/sbin/nologin）。服务以该用户身份从 /var/lib/allwatch 运行。

安装

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

软件包安装程序和单元。当 systemd 正在运行时，它会启用并启动 allwatch.service。随附的配置没有活动目标进程，因此服务启动后等待。

检查软件包状态和程序版本：

```
dpkg -s allwatch | grep -E '^(Status|Version):'
allwatch --version
```

allwatch --version 输出 allwatch 0.1.0；软件包版本是 0.1.0-7。

路径和文件

- /etc/allwatch/allwatch.conf 是由 root 拥有的 conffile，服务用户不可写。它是信任边界：目标进程的命令以服务用户的权限运行。
- /var/lib/allwatch 保存状态和日志，权限 0700，所有者 allwatch:allwatch，由 systemd 创建。平面日志是 /var/lib/allwatch/allwatch/allwatch.log。
- /run/allwatch 是运行时目录，权限 0700，在每次服务启动时创建，在服务停止时移除。它保存 <id>.pid、<id>.fails 和 <id>.stop。
- /var/lib/allwatch 和 /run/allwatch 不会在 purge 后保留；/run/allwatch 不会在停止后保留。

声明一个目标进程

以 root 身份编辑 /etc/allwatch/allwatch.conf，为每个目标进程添加一行：

```
id port policy command...
```

一个使用软件包所装测试替身的真实示例：

```
sudo apt install netcat-openbsd
sudo cp /usr/share/doc/allwatch/examples/allwatch.conf.example /etc/allwatch/allwatch.conf
sudo systemctl restart allwatch
```

示例声明：

```
demo 9001 on-failure /usr/share/doc/allwatch/examples/dummy-target.sh 9001
```

检查端口：

```
bash -c ': > /dev/tcp/127.0.0.1/9001' && echo "port 9001 answers"
```

服务用户必须能够执行该命令。请使用绝对路径：命令按空格拆分，不支持引号，也不经过 shell 运行。对于带空格或 shell 语法的参数，请编写一个包装脚本并调用它。

无响应的目标进程：重启或告警

ON_FREEZE 决定当目标进程仍然存活、但在宽限期后其 TCP 端口不再响应时会发生什么。ON_FREEZE=restart（随附的默认值）发送 SIGTERM，等待约五秒，必要时发送 SIGKILL，然后按目标进程的策略重启它。ON_FREEZE=alert 不发送信号：它记录一次 ALERTE GEL，让目标进程继续运行，并在端口重新响应时记录 RETABLI。崩溃在两种取值下都遵循目标进程的策略。其他任何值都会在启动时以退出码 1 拒绝。

服务命令

```
sudo systemctl status allwatch
sudo systemctl start allwatch
sudo systemctl stop allwatch
sudo systemctl restart allwatch
sudo systemctl enable allwatch
sudo systemctl disable allwatch
sudo journalctl -u allwatch -f
sudo tail -f /var/lib/allwatch/allwatch/allwatch.log
```

配置在启动时读取一次；修改后需要运行 sudo systemctl restart allwatch。不支持热重载。

在不启动目标进程的情况下检查配置

以服务用户身份、使用服务环境运行 --diagnose：

```
sudo -u allwatch env ALLWATCH_RUNTIME=/run/allwatch \
  XDG_STATE_HOME=/var/lib/allwatch \
  allwatch --diagnose /etc/allwatch/allwatch.conf
```

它输出版本、配置路径、运行时路径、日志路径、无响应策略，以及每个目标进程的标识、端口和策略。当配置可读、已存在的运行时目录属于当前用户、已存在的日志目录可写时，它以 0 退出。配置不可读、ON_FREEZE 无效、POLL_SECONDS 不是数值、已存在的运行时目录不属于当前用户，或已存在的日志目录不可写时，它以 1 退出。它不启动任何目标进程，也不探测任何端口。

在不停止服务的情况下停止一个目标进程

以服务用户身份创建停止文件：

```
sudo -u allwatch touch /run/allwatch/demo.stop
```

AllWatch 在目标进程结束时移除该文件，并且不再重启它。该文件本身不会停止正在运行的目标进程；请向目标进程发送 SIGTERM，或让它自行结束。

以另一个用户身份运行

服务以 allwatch 身份运行。如果目标进程需要 root 或另一个账户，请创建 drop-in：

```
sudo systemctl edit allwatch
```

添加：

```
[Service]
User=www-data
Group=www-data
```

然后重启：

```
sudo systemctl restart allwatch
```

目标进程的命令继承该身份。配置文件仍由 root 拥有。

升级

```
sudo apt install ./allwatch_0.1.0-7_all.deb
```

/etc/allwatch/allwatch.conf 会被保留；如果软件包中的版本发生变化，dpkg 会询问保留哪个版本。如果服务在升级前处于活动状态，升级后会重启它。/var/lib/allwatch 及其日志会保留。

移除和清除

```
sudo apt remove allwatch
sudo apt purge allwatch
```

apt remove 停止服务，移除程序、单元、手册页和软件包文档。它保留 /etc/allwatch/allwatch.conf、/var/lib/allwatch 以及 allwatch 用户和组。它不会禁用单元；deb-systemd-helper 的启用状态会保留到 purge。

apt purge 还会移除 conffile、/var/lib/allwatch、为空时的 /etc/allwatch、可移除时的 allwatch 用户和组，以及启用状态。如果仍有进程使用该账户，userdel 或 groupdel 可能失败；请用以下命令检查：

```
getent passwd allwatch
getent group allwatch
```

其他发行版系列：**Fedora**、**openSUSE Leap** 和 **Arch Linux**

Fedora 和 openSUSE Leap 的 .rpm 以及 Arch Linux 的 .pkg.tar.zst 会把与 Debian 软件包相同的文件安装到相同的路径：/usr/bin/allwatch、/usr/lib/systemd/system/allwatch.service、/etc/allwatch/allwatch.conf、手册页和 /usr/share/doc/allwatch。安装您所用系列的文件（依次为 Fedora、openSUSE Leap、Arch Linux）；同一行命令也可升级已安装的版本：

```
sudo dnf install ./allwatch-0.1.0-1.fc44.noarch.rpm
sudo zypper install --allow-unsigned-rpm ./allwatch-0.1.0-1.leap160.noarch.rpm
sudo pacman -U ./allwatch-0.1.0-1-any.pkg.tar.zst
```

与 Debian 软件包的区别：服务已安装但未启动，请在声明好目标后运行一次 sudo systemctl enable --now allwatch；若您修改过 /etc/allwatch/allwatch.conf，它会被保留，并在移除软件包时保存为 allwatch.conf.rpmsave（Fedora、openSUSE）或 allwatch.conf.pacsave（Arch）；没有清除（purge）操作，因此移除会保留用户 allwatch 和 /var/lib/allwatch，软件包会显示清除它们的两条命令：

```
sudo userdel allwatch
sudo rm -rf /etc/allwatch /var/lib/allwatch
```

软件包的已知限制

- 容器验证安装软件包，检查 --version、--help 和 --diagnose，然后移除它。它不启动 systemd，因此服务虽已安装并记录，但未在容器中的 systemd 下验证。
- --diagnose 是静态的：它不探测目标进程。
- /run/allwatch 不会在服务停止后保留。监控器崩溃后，目标进程可能在没有 PID 文件的情况下继续存活；重启后的监控器可能启动重复实例。带有 RuntimeDirectoryPreserve=restart 的 drop-in 可以在服务重启期间保留该目录。

故障排查

AllWatch 没有状态命令。目标进程的状态从进程、TCP 端口和平面日志中读取。本章给出常见故障的确切命令。

先读取服务状态

```
sudo systemctl status allwatch
sudo journalctl -u allwatch -n 50 --no-pager
sudo tail -n 50 /var/lib/allwatch/allwatch/allwatch.log
```

日志行是 date id event。事件包括 DEMARRAGE、DEMARRE、ADOPTE、PLANTE、ALERTE GEL、RETABLI、NEUTRALISE SIGTERM、NEUTRALISE SIGKILL、CRASHE code=N、RELANCE、ALERTE VERROU_CRITIQUE、QUITTE 和 ARRET。

服务没有启动

程序按系统语言（LC_ALL、LC_MESSAGES 或 LANG）输出这些消息；默认语言为英语。

- allwatch: 无法读取配置文件：/etc/allwatch/allwatch.conf——文件缺失或服务用户无法读取。请检查 sudo -u allwatch test -r /etc/allwatch/allwatch.conf 和 sudo systemctl cat allwatch。
- allwatch: /run/allwatch 不属于当前用户——运行时目录属于另一个用户。请检查 ls -ld /run/allwatch /var/lib/allwatch；单元以 allwatch 身份创建它们。如果你添加了使用另一个 User= 的 drop-in，请修正所有权或移除该 drop-in。
- status=203/EXEC——程序缺失或不可执行。请重新安装软件包。
- 配置错误导致 status=1/FAILURE——以服务用户身份运行 --diagnose（见下文）。

目标进程没有被重启

- 策略 on-failure：以退出码 0 结束的目标进程会保持停止。这是约定，不是故障。如果它必须在干净退出后重启，请使用 always。
- 策略 transient：从不重启。
- 存在停止文件：ls -l /run/allwatch/demo.stop；如果停止并非本意，请移除它。该文件本身不会停止正在运行的目标进程。
- ALERTE VERROU_CRITIQUE：目标进程在 RESTART_WINDOW 内达到 MAX_RESTARTS。修复原因，然后重启服务；锁定会持续到监控器重启。
- CRASHE code=N：目标进程以退出码 N 结束。请读取它自己的日志；AllWatch 只报告退出码。

目标进程被报告为无响应，但它仍在运行

- 探测命令是 timeout 3 bash -c ': > /dev/tcp/127.0.0.1/9001'。目标进程必须在声明的端口上监听 127.0.0.1（或 0.0.0.0）。只在另一个地址上监听的目标进程会被视为无响应。
- /etc/allwatch/allwatch.conf 中的端口错误；请用 ss -ltnp | grep 9001 检查。
- 目标进程比 PROBE_TIMEOUT=3 慢；请在配置中提高该值，然后重启服务。
- 目标进程存活但其监听器已关闭；这正是无响应状态。使用 ON_FREEZE=restart（默认值）时，AllWatch 发送 SIGTERM，等待 5 秒，然后发送 SIGKILL，并重启它。使用 ON_FREEZE=alert 时，它记录一次 ALERTE GEL，让目标进程继续运行；端口重新响应时记录 RETABLI。

`--diagnose` 返回 `0`，但目标进程不健康

--diagnose 是静态的。它读取配置和路径；它不探测目标进程。返回码 0 只能证明监控器能够读取配置并使用其目录。

测试替身不响应

- 缺少 netcat-openbsd：sudo apt install netcat-openbsd。
- 命令使用了相对路径。服务从 /var/lib/allwatch 运行；请使用绝对路径 /usr/share/doc/allwatch/examples/dummy-target.sh。
- 端口已被占用：ss -ltnp | grep 9001。
- 服务用户无法执行该文件：sudo -u allwatch test -x /usr/share/doc/allwatch/examples/dummy-target.sh。

日志无限增长

AllWatch 不轮转日志。请使用 logrotate 或 systemd 定时器；文件是 /var/lib/allwatch/allwatch/allwatch.log。

监控器重启后出现重复的目标进程

服务停止时会移除 /run/allwatch。目标进程可能在监控器崩溃后继续存活，却没有 PID 文件；新的监控器会启动一个重复实例。请添加 drop-in：

```
sudo systemctl edit allwatch
```

```
[Service]
RuntimeDirectoryPreserve=restart
```

然后重启。运行时目录会在服务重启期间保留，因此新的监控器可以接管仍然存活的 PID。

日志中没有任何内容

服务以 allwatch 身份运行，并写入 /var/lib/allwatch/allwatch/allwatch.log；请用 sudo systemctl show allwatch -p Environment 检查服务环境。如果目录不可写，日志会保持为空；--diagnose 会报告这一点。