EAP功能思考

作者：滑国青

历史：2023.08.02

# 前言

# 软狗

## soft-wdt

[GitHub - sunmingbao/soft-wdt: A software watchdog timer for linux. Support multiple watchdogs.](https://github.com/sunmingbao/soft-wdt)

[sunmingbao (Mingbao Sun) · GitHub](https://github.com/sunmingbao)

[无忧老猪\_Linux应用开发技术,Linux内核学习笔记,Linux内核、驱动开发移植-CSDN博客](https://blog.csdn.net/crazycoder8848/)

soft\_wdt软件看门狗简介

======================

soft\_wdt(以下简称本软件)是一个软件实现的Linux看门狗，

他和/drivers/watchdog/softdog.c实现的软件看门狗几乎一样。

主要的不同点是，前者支持一个看门狗，本软件则支持大量的看门狗。

soft\_wdt代码编译后，生成一个内核模块soft\_wdt.ko。

模块加载后，将创建一个设备文件/dev/soft\_wdt

用户态程序，通过系统调用open每打开一次/dev/soft\_wdt，就得到一个新的看门狗，

此看门狗的使用方法就和普通的看门狗一样。

例如:

1) 向fd写入任何数据，就等于是喂狗。

2) 用户可以通过ioctl对看门狗进行各种操作。

3) 如果模拟加载时，模块参数nowayout的值为0，

那么当用户向fd写入一次含有字符V(注意，是大写)的数据时，

就将此看门狗设置成了可关闭的。

下面具体说说使用方法：

(一)编译看门狗

见Build.txt

(二)加载看门狗

本软件提供的模块参数如下。用户可根据需要进行指定。

nowayout - 一旦启动看门狗，不可以停止 (0，no；1，yes。default=0)

timeout - 看门狗超时时间，单位：秒。 (0 ~ 65536, default=5)

no\_reboot - 看门狗超时，不重启系统 。(0，no; 1，yes default=0)

core\_dump\_ill\_task - 看门狗超时时，core dump异常任务，(0，no; 1，yes default=1)

注意，core dump是通过向异常线程发送SIGABRT信号实现的。

因此，如果使用看门狗的程序，想自己记录异常信息，可以通过捕获SIGABRT信号来实现。

下面是加载命令的示例。

1. 使用默认参数加载(默认值如上面所列)

insmod soft\_wdt.ko

2. 指定参数加载(12秒超时，看门狗可关闭，超时不重启机器)

insmod soft\_wdt.ko timeout=12 nowayout=0 no\_reboot=1

(三)用户态程序使用看门狗

//以下是示例代码

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <linux/watchdog.h>

#define SOFT\_WDT\_DEV "/dev/soft\_wdt"

int feed\_dog\_cnt;

int feed\_dog\_gap;

int main(int argc, char \*argv[])

{

int i;

int timeout;

struct watchdog\_info ident;

int fd;

if (argc<3)

{

printf("usage:\n %s <feed\_gap(in seconds)> <feed\_cnt>\n", argv[0]);

return 0;

}

fd=open("/dev/soft\_wdt", O\_WRONLY);

if (fd < 0)

{

printf("open %s failed\n", SOFT\_WDT\_DEV);

exit(1);

}

printf("open %s succeed\n", SOFT\_WDT\_DEV);

timeout = 7;

printf("set timeout to %d\n", timeout);

ioctl(fd, WDIOC\_SETTIMEOUT, &timeout);

timeout = 0;

ioctl(fd, WDIOC\_GETTIMEOUT, &timeout);

printf("get timeout returns %d\n", timeout);

ioctl(fd, WDIOC\_GETSUPPORT, &ident);

printf("dog name is %s\n", ident.identity);

printf("make the dog closable\n");

write(fd, "V", 1);

feed\_dog\_gap = atoi(argv[1]);

feed\_dog\_cnt = atoi(argv[2]);

for (i=0; i<feed\_dog\_cnt; i++)

{

printf("feed dog\n");

write(fd, "1234", 4);

usleep(feed\_dog\_gap\*1000000);

}

printf("stop feeding dog\n");

while (1)

{

usleep(1000000);

}

return 0;

}

# 结束