

浙江大学实验报告

课程名称：嵌入式系统

指导老师：翁凯

姓名：张腾

实验名称：看门狗

实验类型：嵌入式开发

学号：3120101111

一、实验目的和要求

配置内核中的硬件看门狗，使得一定时间内不喂狗就重启 Acadia 或 RPi 或 WRTnode，写一个程序或脚本保持一定频率的喂狗，当关闭这个程序或脚本时形成重启。实验报告要记录和表现出重启。

二、实验内容和原理

1. 掌握看门狗的概念；
2. 掌握 Acadia 或 RPi 或 WRTnode 上编写看门狗程序的方法。

三、主要仪器设备

硬件

Acadia 或 RPi 或 WRTnode 板一块；

5V/1A 电源一个；

microUSB 线一根；

USB-TTL 串口线一根（FT232RL 芯片或 PL2303 芯片）。

以下为自备（可选）器材：

PC（Windows/Mac OS/Linux）一台；

以太网线一根（可能还需要路由器等）。

软件

PC 上的 USB-TTL 串口线配套的驱动程序；

PC 上的串口终端软件，如 minicom、picocom、putty 等；

PC 上的 SSH 软件，如 putty 等。

实验名称: 看门狗 姓名: 张腾 学号: 3120101111

四、操作方法和实验步骤

1. 掌握看门狗的概念;

Linux 自带了一个 watchdog 的实现, 用于监视系统的运行, 包括一个内核 watchdog module 和一个用户空间的 watchdog 程序。内核 watchdog 模块通过 /dev/watchdog 这个字符设备与用户空间通信。用户空间程序一旦打开 /dev/watchdog 设备 (俗称“开门放狗”), 就会导致在内核中启动一个 1 分钟的定时器 (系统默认时间), 此后, 用户空间程序需要保证在 1 分钟之内向这个设备写入数据 (俗称“定期喂狗”), 每次写操作会导致重新设定定时器。如果用户空间程序在 1 分钟之内没有写操作, 定时器到期会导致一次系统 reboot 操作。通过这种机制, 我们可以保证系统核心进程大部分时间都处于运行状态, 即使特定情形下进程崩溃, 因无法正常定时“喂狗”, Linux 系统在看门狗作用下重新启动 (reboot), 核心进程又运行起来了。多用于嵌入式系统。

2. 掌握 Acadia 或 RPi 或 WRTnode 上编写看门狗程序的方法。

首先加载看门狗模块, 编辑 /etc/modules, 添加“bcm2708_wdog”

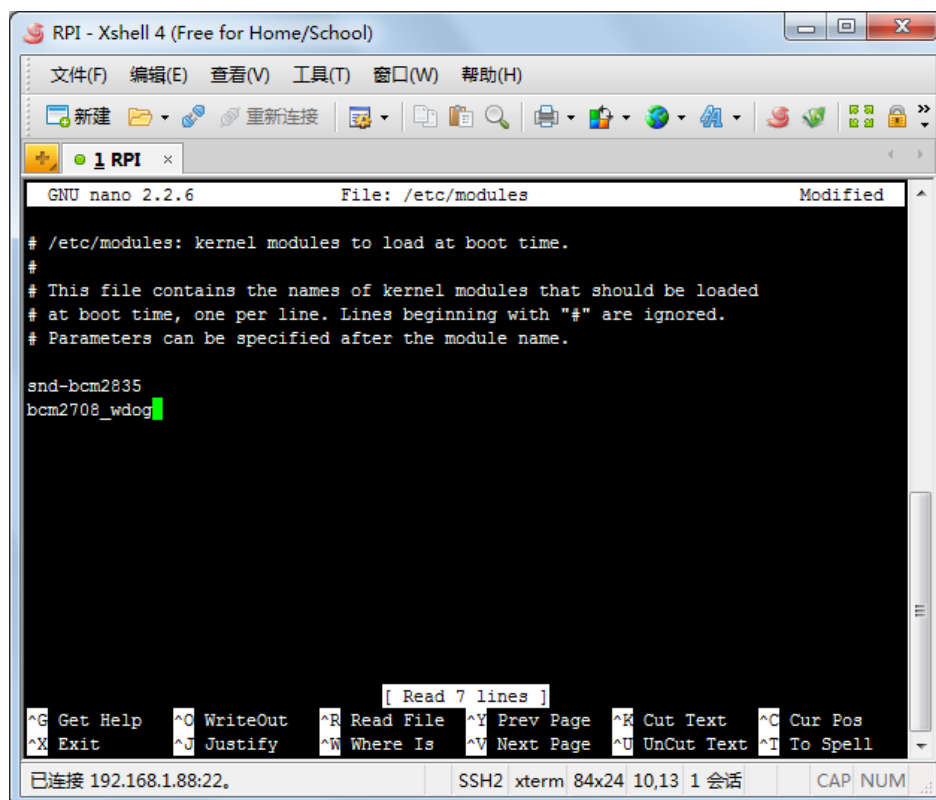
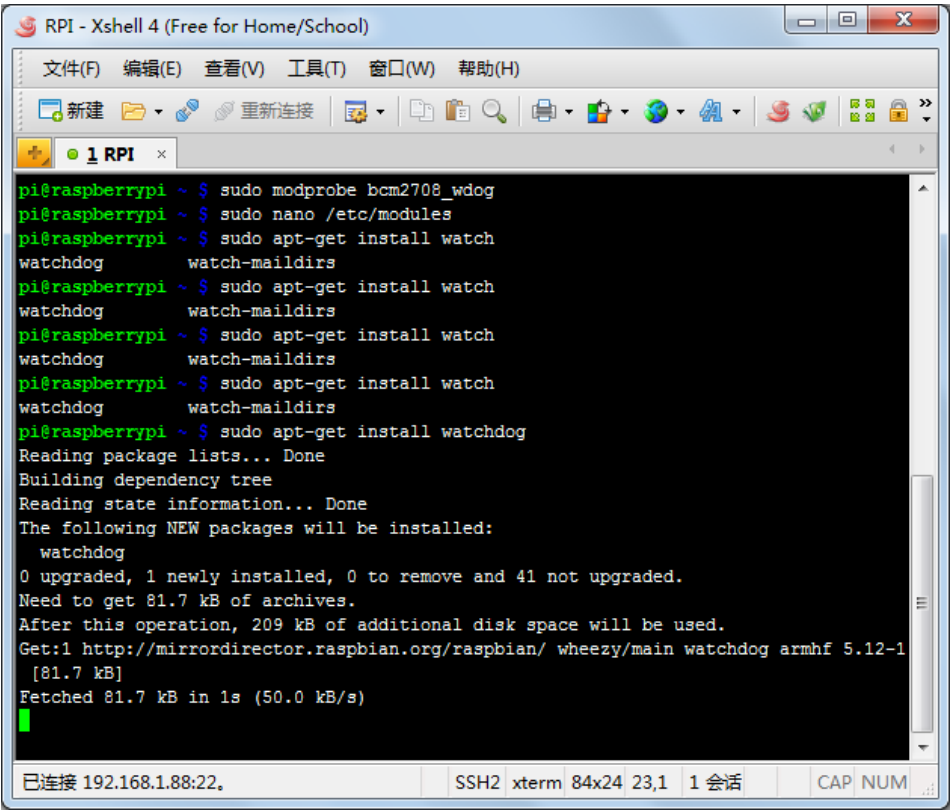


Figure 1

实验名称：看门狗 姓名：张腾 学号：3120101111

使用 apt-get 下载 watchdog



```
RPI - Xshell 4 (Free for Home/School)
文件(F) 编辑(E) 查看(V) 工具(T) 窗口(W) 帮助(H)
新建 重新连接
1 RPI
pi@raspberrypi ~ $ sudo modprobe bcm2708_wdog
pi@raspberrypi ~ $ sudo nano /etc/modules
pi@raspberrypi ~ $ sudo apt-get install watch
watchdog watch-malldirs
pi@raspberrypi ~ $ sudo apt-get install watch
watchdog watch-malldirs
pi@raspberrypi ~ $ sudo apt-get install watch
watchdog watch-malldirs
pi@raspberrypi ~ $ sudo apt-get install watchdog
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  watchdog
0 upgraded, 1 newly installed, 0 to remove and 41 not upgraded.
Need to get 81.7 kB of archives.
After this operation, 209 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main watchdog armhf 5.12-1
[81.7 kB]
Fetched 81.7 kB in 1s (50.0 kB/s)
已连接 192.168.1.88:22. SSH2 xterm 84x24 23,1 1 会话 CAP NUM
```

Figure 2

接下来需要编写定时喂狗程序，在上文已经提到，看门狗程序是通过/dev/watchdog 这个字符设备与用户空间进行通信的，所以，我们需要做的就是，设置饥饿时间（即多长时间不喂狗就会 reboot），另外利用一个循环按照一定频率向/dev/watchdog 设备写字符，当我们关掉这个程序时，就会造成 reboot。

喂狗程序如下（C 语言）：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>
#include <time.h>
#include <getopt.h>
#include <sys/signal.h>
```

实验名称: 看门狗 姓名: 张腾 学号: 3120101111

```
#include <termios.h>

struct watchdog_info{
    unsigned int options; //options the card/driver supports 19
    unsigned int firmware_version; //firmcard version of the card
    unsigned char identity[32]; //identity of the board 21
};

#define WATCHDOG_IOCTL_BASE 'W'
#define WDIOC_GETSUPPORT _IOR(WATCHDOG_IOCTL_BASE, 0, struct watchdog_info)
#define WDIOC_SETTIMEOUT _IOW(WATCHDOG_IOCTL_BASE, 6, int)
#define WDIOC_GETTIMEOUT _IOR(WATCHDOG_IOCTL_BASE, 7, int)
#define WDIOS_DISABLECARD 0x0001 /* Turn off the watchdog timer */
#define WDIOS_ENABLECARD 0x0002 /* Turn on the watchdog timer */
#define WDIOC_SETOPTIONS _IOR(WATCHDOG_IOCTL_BASE, 4, int)
#define WDIOC_KEEPAIVE _IOR(WATCHDOG_IOCTL_BASE, 5, int)

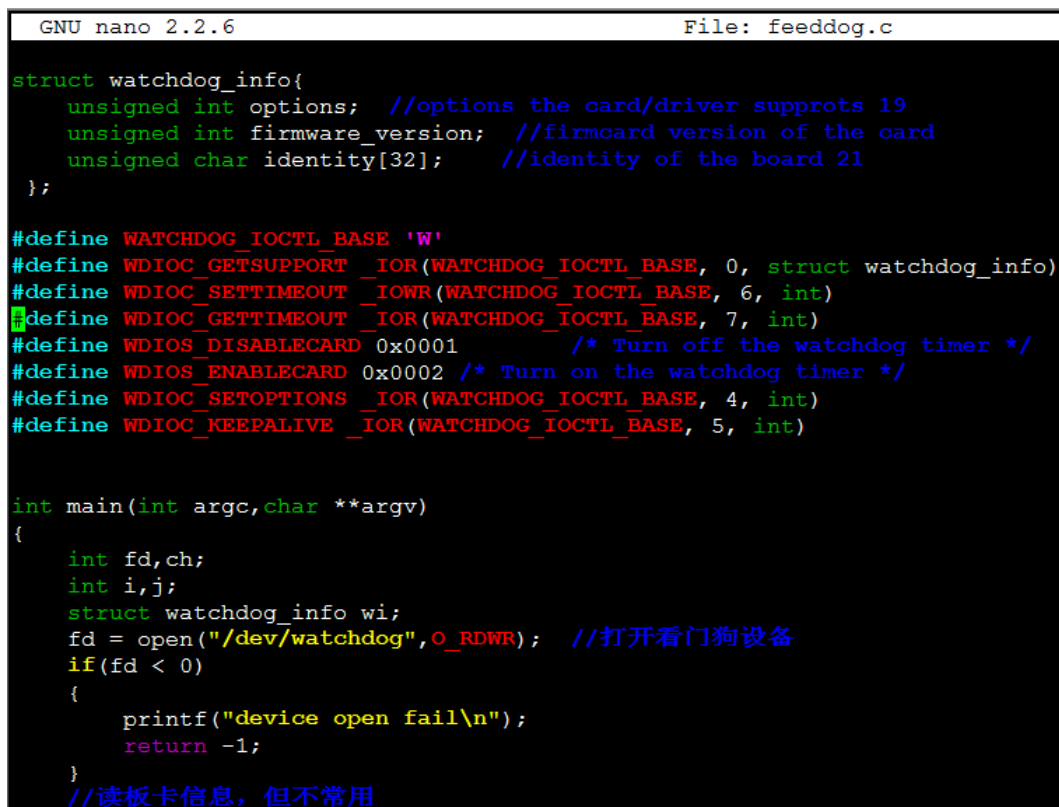
int main(int argc,char **argv)
{
    int fd,ch;
    int i,j;
    struct watchdog_info wi;
    fd = open("/dev/watchdog",O_RDWR); //打开看门狗设备
    if(fd < 0)
    {
        printf("device open fail\n");
        return -1;
    }
    //读板卡信息, 但不常用
    ioctl(fd,WDIOC_GETSUPPORT,&wi);
    printf("%d,%s\n",wi.options,wi.identity);
    //重新设置时间为 10s
    i=10;
    printf("%d\n",ioctl(fd,WDIOC_SETTIMEOUT,&i));
    //读新的设置时间
    printf("%d\n",ioctl(fd,WDIOC_GETTIMEOUT,&i));
    printf("%d\n",i);
    //看门狗开始和停止工作, 打开和关闭设备具有同样的功能
    //关闭
    i=WDIOS_DISABLECARD;
```

实验名称: 看门狗 姓名: 张腾 学号: 3120101111

```
printf("%d\n",ioctl(fd,WDIIOC_SETOPTIONS,&i));
//打开
i=WDIOS_ENABLECARD;
printf("%d\n",ioctl(fd,WDIIOC_SETOPTIONS,&i));

while(1)
{
    sleep(0.1);
    ioctl(fd,WDIIOC_KEEPAIVE,NULL);
}

close(fd); //关闭设备
return 0;
}
```



```
GNU nano 2.2.6                                File: feddog.c

struct watchdog_info{
    unsigned int options; //options the card/driver supports 19
    unsigned int firmware_version; //firmcard version of the card
    unsigned char identity[32]; //identity of the board 21
};

#define WATCHDOG_IOCTL_BASE 'W'
#define WDIIOC_GETSUPPORT _IOR(WATCHDOG_IOCTL_BASE, 0, struct watchdog_info)
#define WDIIOC_SETTIMEOUT _IOWR(WATCHDOG_IOCTL_BASE, 6, int)
#define WDIIOC_GETTIMEOUT _IOR(WATCHDOG_IOCTL_BASE, 7, int)
#define WDIOS_DISABLECARD 0x0001 /* Turn off the watchdog timer */
#define WDIOS_ENABLECARD 0x0002 /* Turn on the watchdog timer */
#define WDIIOC_SETOPTIONS _IOR(WATCHDOG_IOCTL_BASE, 4, int)
#define WDIIOC_KEEPAIVE _IOR(WATCHDOG_IOCTL_BASE, 5, int)

int main(int argc, char **argv)
{
    int fd, ch;
    int i, j;
    struct watchdog_info wi;
    fd = open("/dev/watchdog", O_RDWR); //打开看门狗设备
    if (fd < 0)
    {
        printf("device open fail\n");
        return -1;
    }
    //读板卡信息, 但不常用
}
```

Figure 3

实验名称: 看门狗 姓名: 张腾 学号: 3120101111

gcc -o feeddog feeddog.c 生成可执行文件

```
[Mon Mar 30 20:35:55 2015] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0x4DE1
[Mon Mar 30 20:36:08 2015] Adding 102396k swap on /var/swap. Priority:-1 extents:2 across:2134012k SSFS
pi@raspberrypi ~ $ clear
pi@raspberrypi ~ $ sudo nano feeddog.c
pi@raspberrypi ~ $ gcc -o feeddog feeddog.c
pi@raspberrypi ~ $ sudo nano feeddog.c
pi@raspberrypi ~ $ sudo nano feeddog.c
pi@raspberrypi ~ $ gcc -o feeddog feeddog.c
pi@raspberrypi ~ $
```

Figure 4

在后台执行程序，由于持续喂狗，不断将看门狗的计数寄存器清零，所以树莓派没有 reboot

```
pi@raspberrypi ~ $ gcc -o feeddog feeddog.c
pi@raspberrypi ~ $ sudo ./feeddog &
[1] 3048
pi@raspberrypi ~ $ 33152,BCM2708
0
0
10
0
0
pi@raspberrypi ~ $
```

Figure 5

使用 kill -9 结束该进程

```
pi@raspberrypi ~ $ sudo ./feeddog &
[1] 3048
pi@raspberrypi ~ $ 33152,BCM2708
0
0
10
0
0
pi@raspberrypi ~ $ sudo kill -9 3048
[1]+  Killed                  sudo ./feeddog
```

Figure 6

可以看到 SSH 连接断开，树莓派重启

```
pi@raspberrypi ~ $
Connection closed by foreign host.

Ty
```

未连接。

Figure 7

五、实验数据记录和处理

暂无实验数据

六、实验结果与分析

完成全部实验要求

七、讨论、心得

通过本次试验，了解了看门狗的概念，并能够在树莓派上实现定时喂狗，在以后编程时应该可以利用看门狗的特性。