

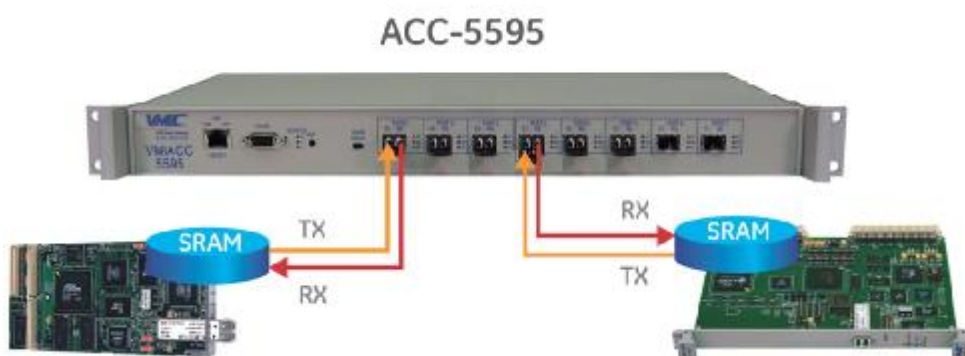
Windows 和 Linux 下反射内存的相关安装和测试

目 录

一、硬件安装与连接.....	1
二、Windows 下的测试.....	1
1、驱动安装与连接测试.....	1
2、单台 PC 读写测试.....	2
3、点对点读写测试	2
二、Linux 下的编译和测试.....	3
1、下载 linux 驱动程序包.....	3
2、安装正确的 linux 内核版本	3
3、修改 build 指向的目录.....	4
4、修改两个文件	5
5、编译	5
6、安装驱动程序	5
7、运行诊断程序	6

一、硬件安装与连接

硬件包括两张 VMIC5565 卡，Hub5595 一台。尾纤两根。厂家设定的板卡 NodeID 都为 0X00，改变 5565 板上跳线的位置，使卡的 NodeID 不重复。网路连接如下图所示。



二、Windows 下的测试

1、驱动安装与连接测试

在两台 Windows 操作系统计算机上正确安装 VMIC5565 卡，连接好光纤；重新启动操作系统，操作系统发现新硬件；将文件夹 162-000447-940 中 PMC5565.sys 驱动文件添加给新发现的硬件，完成驱动安装。

板卡和 Hub 连接和工作时，Status, SIG.DET, OWN.Data 状态灯显示正常。

运行文件夹中提供的 Setup.exe 程序，安装 RFM2g Util 程序，在两台 PC 机的控制台模式下，运行 RFM2g Util 程序，键入“1”，打开 VMIC5565 卡。

获取板级 ID 和节点 ID,通过 checkring, 来检查两张板卡间的通讯是否正常。

2、单台 PC 读写测试

通过运行 `performance` 命令，测试本地读写得到的带宽和每秒读写次数，这个指标与计算机平台的配置和性能有关系。 其中一台测试结果如下

```

#1111 > performance
VMIC RW2g Performance Test (DMA Threshold is 64)

```

Bytes	Read IOPS	Read MBps	Write IOPS	Write MBps
4	90572	0.3	92865	0.4
8	88868	0.6	90815	0.7
12	78855	0.9	85162	1.0
16	72886	1.1	83758	1.3
20	65516	1.2	80620	1.5
24	57618	1.3	77957	1.8
28	59435	1.6	76134	2.0
32	58708	1.8	75306	2.3
64	23660	1.4	24178	1.5
96	23788	2.2	24854	2.2
128	22642	2.9	24800	2.9
256	23168	5.7	23687	5.8
384	22780	8.3	22862	8.4
512	22887	10.7	21739	10.6
640	21625	13.2	20728	12.7
768	21688	15.4	20827	14.7
896	20327	17.4	18577	15.9
1024	17454	17.0	17325	16.9
1152	14123	15.5	15947	14.6
1280	16665	20.3	17073	21.3
1408	10363	24.7	16899	22.7
1536	18440	27.0	17806	24.9
1664	18073	28.7	16825	25.4
1792	12441	21.2	14906	25.5
1920	14637	25.7	8288	15.5
2048	14848	27.4	18169	19.9
2560	15564	38.0	13717	33.5
3072	8841	25.9	11994	35.1
3584	15367	35.4	11871	37.8
4096	9822	38.4	9815	38.1
4608	6620	51.7	5308	41.5
5120	4824	57.2	3678	43.4
5632	2197	34.3	3478	54.3
6144	1788	59.6	1920	60.0
6656	1224	57.4	1255	58.9
7168	744	46.5	727	45.6
8192	54	58.7	654	51.4
9216	588	51.4	614	57.6
10240	468	50.9	509	55.1
11264	375	46.9	418	53.3
12288	257	41.5	247	46.3
13312	173	43.3	184	46.4

所示，其中，IOPS 表示每秒读写次数，MBps 为带宽。

3、点对点读写测试

对两张卡完成点对点读写操作，写数据格式为：

Write 数据 地址 每次写入数据包大小 数据个数

Read 地址 每次读取数据包大小 数据个数

两台机器在特定地址完成读写操作，读写结果一致。

二、Linux 下的编译和测试

1、下载 linux 驱动程序包

目前最新的驱动是直接从官方网站上下载的，文件名是 162-RFM2G-DRV-LNX-R07_03-000。

2、安装正确的 linux 内核版本

可以想见，在这个 linux 驱动程序包释放出来的时候，linux 的最新内核版本应该是在 2.6.18 左右，而目前的 ubuntu9.10 的内核直接是 2.6.31。

因为这个源代码包用到了 linux 内核相关东西，而不同版本的内核的源代码肯定是不一样的，2.6.18 内核的源代码和 2.6.31 内核的源代码肯定就不同，事实也是如此。如果直接用 2.6.31 的内核源码编译的时候，会碰到一个很头痛的问题，2.6.18 的 `proc_entry` 的结构在 2.6.31 中已经改的面目全非，直接编译通不过。

因此，在本次调试的过程中，将 ubuntu 的版本直接降级为 2.6.28（找不到 2.6.18 的 deb 包,图个方便），解决这个 `proc_entry` 的编译问题。

Ubuntu 中降内核版本的方法很简单，直接去下面的网址：

<http://kernel.ubuntu.com/~kernel-ppa/mainline/?C=M;O=D>

下载自己想要的版本，需要说明的是，以 2.6.28.10 版本为例，首先下载二进制的包，对于 32 位机器来讲，是

[linux-image-2.6.28-02062810-generic 2.6.28-02062810_i386.deb](#)

这个包并不会提供相应的 2.6.28 内核对应的头文件，而这些头文件会被驱动程序所用到，所以还需要下载头文件包，另外因为 `generic` 的版本会直接依赖于不带 `generic` 的版本，所以需要下载：

[linux-headers-2.6.28-02062810 2.6.28-02062810 all.deb](#)

[linux-headers-2.6.28-02062810-generic 2.6.28-02062810 i386.deb](#)

两个 deb 包。

然后首先安装头文件包，命令如下，

```
dpkg -i linux-headers-2.6.28-02062810 2.6.28-02062810 all.deb
```

```
dpkg -i linux-headers-2.6.28-02062810-generic 2.6.28-02062810 i386.deb
```

然后安装二进制包，命令如下：

```
dpkg -i linux-image-2.6.28-02062810-generic 2.6.28-02062810 i386.deb
```

最后修改 grub 的启动顺序，如果是 grub1（ubuntu9.10 以前），就直接修改/boot/grub/menu.lst 即可，如果是 grub2（ubuntu9.10 以后），直接修改/boot/grub/grub.cfg

3、修改 build 指向的目录

如果是先安装二进制包，后安装头文件包，在编译的时候铁定会出现一个编译问题，即在编译驱动程序 rfm2g.ko 的时候，找不到 modules 的编译方法，问题的症结在于在/lib/modules/2.6.28-10-generic 下没有一个 build 文件夹，而这个文件夹是一个引用，直接指向 linux 头文件夹的，make modules 的编译方法就在这个头文件夹中。

如果是碰到了这种情况，使用下面的这个命令就可以：

```
ln -s /usr/src/linux-2.6.28-10-generic /lib/modules/2.6.28-10-generic/build
```

4、修改两个文件

同样由于版本的问题，会导致编译的时候出现两个不是很致命的编译问题。

首先修改 rfm2g\api\config.c 文件中的第 73 行，即

```
#include <asm/page.h>
```

这个头文件在 2.6.28 中已经是取消了的，所以将之注释掉即可

```
//#include <asm/page.h>
```

另外一个需要修改的文件是在 rfm2g\driver\rfm2g_int.c 文件中，有两处 &proc_root，这个东西在 2.6.28 中也有变化，直接将之替换成 NULL 即可。

5、编译

直接在 rfm2g 文件夹下使用以下命令：

```
make install
```

这个命令将会生成两个重要的东西：

- 1、在 drivers 目录中，生成驱动程序 rfm2g.ko
- 2、在 diags 目录中，会生成可执行诊断程序 rfmutil

在这两个模块生成之后，make install 会调用 rfm2g_init，会报错，报错的原因是找不到 etc/rc.d/init.d 文件夹，redhat 中有，但是 ubuntu 中没有，但是做了变通，能够实现同样的功能，不过不用管它。

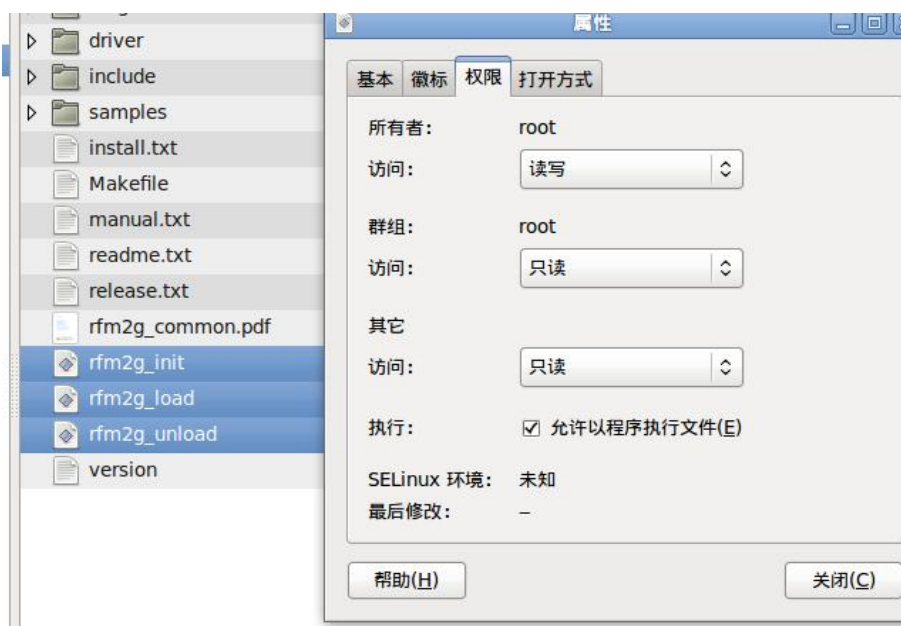
6、安装驱动程序

Linux 中的驱动程序就是一个模块，直接用 insmod 安装，用 rmmod 卸载。

不过在这里推荐使用 rfm2g_load 和 rfm2g_unload 这两个脚本来实现安装和卸载，这两个脚本是在 rfm2g 的文件夹下。原因是 rfm2g_load 在安装驱动程序之后，在 /dev 文件夹下建立了对应的文件 /dev/rfm2g0~4，而这个是后面的诊断程序需要打开和关闭的文件。

如果是直接用 insmod 安装，首先调用 rfm2g_unload 卸载，然后用 rfm2g_load 安装一下即可。

在运行 rfm2g_load 之前，需要把三个执行文件权限设为允许以程序执行文件的方式。如下图所示：



在 /rfm2g 目录下运行执行文件 ./rfm2g_load，如下图所示，完成安装。



7、运行诊断程序

同样在\diags 目录下放开 rfm_util 执行文件权限，运行 ./rfm2g.util，如下图所示：


```
International Copyright Secured. All Rights Reserved.

insmod: error inserting '/lib/modules/2.6.32-24-generic/extra/rfm:
exists
root@ubuntu:~/rfm2g# ls
api      include      manual.txt    rfm2g_common.pdf  rfm2g_unload
diags    install.txt  readme.txt    rfm2g_init         samples
driver   Makefile     release.txt   rfm2g_load         version
root@ubuntu:~/rfm2g# cd diags
root@ubuntu:~/rfm2g/diags# ls
Makefile      rfm2g_util      rfm2g_util.o
Makefile_PPC  rfm2g_util.c    rfm2g_utilospec.c
root@ubuntu:~/rfm2g/diags# ./rfm2g_util

PCI RFM2g Commandline Diagnostic Utility

Available devices:

0. /dev/rfm2g0 (VMIPCI-5565, Node 4)
Please enter device number: 0
```

可以看到 RFM 卡当前的型号，NODE ID 号。

运行 rfm_util，会出现一个选择，让选择使用几号反射内存，如果只有一块，直接选择 0 即可。

然后可以运行相关的诊断程序，例如：boardid ,nodeid,Performance。

```
root@ubuntu:~/rfm2g/diags# ./rfm2g_util

PCI RFM2g Commandline Diagnostic Utility

Available devices:

0. /dev/rfm2g0 (VMIPCI-5565, Node 4)
Please enter device number: 0
UTIL0 > boardid
Invalid command "boardid".
UTIL0 > boardid
Board ID          0x05 (VMIPCI-5565)

UTIL0 > nodeid
Node ID           0x04

UTIL0 > performance

RFM2g Performance Test (DMA Threshold is 4394967295)
-----
Bytes    Read IOps    Read MBps    Write IOps    Write MBps
4        307450          1.2          3642943        6.3
8        1993234         7.7          3548676       27.1
32       240030         2.0          1247757        14.3
16       568778         8.7          3185280       47.4
20       296375         4.0          1868929       20.4
```

8、单卡读写测试

在 linux 下测试单卡读写速度，结果如下：

Auto-Transformer				
Transformer Data Table (kVA, Taps, etc.)				
Rating	Primary	Secondary	Ratio	Notes
100	10000	1000	10:1	
150	15000	1500	10:1	
200	20000	2000	10:1	
250	25000	2500	10:1	
300	30000	3000	10:1	
350	35000	3500	10:1	
400	40000	4000	10:1	
450	45000	4500	10:1	
500	50000	5000	10:1	
550	55000	5500	10:1	
600	60000	6000	10:1	
650	65000	6500	10:1	
700	70000	7000	10:1	
750	75000	7500	10:1	
800	80000	8000	10:1	
850	85000	8500	10:1	
900	90000	9000	10:1	
950	95000	9500	10:1	
1000	100000	10000	10:1	
1100	110000	11000	10:1	
1200	120000	12000	10:1	
1300	130000	13000	10:1	
1400	140000	14000	10:1	
1500	150000	15000	10:1	
1600	160000	16000	10:1	
1700	170000	17000	10:1	
1800	180000	18000	10:1	
1900	190000	19000	10:1	
2000	200000	20000	10:1	
2200	220000	22000	10:1	
2400	240000	24000	10:1	
2600	260000	26000	10:1	
2800	280000	28000	10:1	
3000	300000	30000	10:1	
3200	320000	32000	10:1	
3400	340000	34000	10:1	
3600	360000	36000	10:1	
3800	380000	38000	10:1	
4000	400000	40000	10:1	
4200	420000	42000	10:1	
4400	440000	44000	10:1	
4600	460000	46000	10:1	
4800	480000	48000	10:1	
5000	500000	50000	10:1	
5200	520000	52000	10:1	
5400	540000	54000	10:1	
5600	560000	56000	10:1	
5800	580000	58000	10:1	
6000	600000	60000	10:1	
6200	620000	62000	10:1	
6400	640000	64000	10:1	
6600	660000	66000	10:1	
6800	680000	68000	10:1	
7000	700000	70000	10:1	
7200	720000	72000	10:1	
7400	740000	74000	10:1	
7600	760000	76000	10:1	
7800	780000	78000	10:1	
8000	800000	80000	10:1	
8200	820000	82000	10:1	
8400	840000	84000	10:1	
8600	860000	86000	10:1	
8800	880000	88000	10:1	
9000	900000	90000	10:1	
9200	920000	92000	10:1	
9400	940000	94000	10:1	
9600	960000	96000	10:1	
9800	980000	98000	10:1	
10000	1000000	100000	10:1	