

nfc-tools 参考手册

使用 nfc-tools

nfc-tools 提供了多种 PN532 的功能，这些工具是免费、开源的，适用于多种 Linux 发行版。树莓派也可以便捷地获取和编译该工具的源代码，本手册以 Raspbian 为例。

安装依赖

编译和安装 nfc-tools 之前，需要先安装依赖，否则编译可能会失败：

```
sudo apt-get install libusb-dev libpcsclite-dev libtool
sudo apt-get install automake autoconf
```

获取 libnfc 和 mfoc

可以通过 git 命令获取工具的源代码。

```
git clone https://github.com/nfc-tools/libnfc.git
git clone https://github.com/nfc-tools/mfoc.git
```

编译 libnfc

通过以下命令编译 libnfc 工具：

```
cd libnfc
autoreconf -vis
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
```

注意：make install 必须使用 sudo 权限。

编译 mfoc

通过以下命令编译 mfoc 工具：

```
cd mfoc
autoreconf -vis
./configure --prefix=/usr --sysconfdir=/etc
```

`make`

`sudo make install`

注意：`make install` 必须使用 `sudo` 权限。

nfc-tools 串口配置

1. 通过跳线帽，把 I0 设置为 L，I1 设置为 L。
2. 拨码开关设置为

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

即：

- NSS/SCL/RX → 树莓派 TX
 - MOSI/SDA/TX → 树莓派 RX
3. 创建 `/etc/nfc/libnfc.conf`
`sudo mkdir /etc/nfc/`
`sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf`
 4. 编辑 `/etc/nfc/libnfc.conf`
`sudo nano /etc/nfc/libnfc.conf`
删除 `device.connstring = "pn532_uart:/dev/ttyUSB0"` 这一行的 “#”。并把它改成 `device.connstring = "pn532_uart:/dev/ttyS0"`

注意：

- a) 对于树莓派而言，还需要通过 `sudo raspi-config` 树莓派启用串口。默认情况下，树莓派的串口用于登陆 Shell 终端，而如果要使用串口和模块进行通讯的话，需要执行 `sudo raspi-config` 配置串口：
 - Interfacing Options → Serial
 - Would you like a login shell to be accessible over serial? → No
 - Would you like the serial port hardware to be enabled? → Yes
 - b) 对于其他的 Linux 发行版而言，则需要用一个 USB 串口模块，用于把 PN532 NFC HAT 连接到电脑。
5. 如果执行 `nfc-list`，提示 `NFC device: opened`，则接口正常工作。

nfc-tools I2C 配置

1. 通过跳线帽，把 I0 设置为 H，I1 设置为 L
2. 拨码开关设置为

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
-----	------	------	-----	-----	-----	----	----

OFF	OFF	OFF	OFF	ON	ON	OFF	OFF
-----	-----	-----	-----	----	----	-----	-----

即：

- NSS/SCL/RX → 树莓派 SCL
 - MOSI/SDA/TX → 树莓派 SDA
3. 执行 `sudo raspi-config` 启用 I2C 接口：
 - Interfacing Options → I2C
 - Would you like the ARM I2C interface to be enabled? → Yes
 4. 创建 `/etc/nfc/libnfc.conf`

```
sudo mkdir /etc/nfc/
```

```
sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf
```
 5. 编辑 `/etc/nfc/libnfc.conf`

```
sudo nano /etc/nfc/libnfc.conf
```

删除 `device.connstring = "pn532_uart:/dev/ttyUSB0"` 这一行的 “#”。并把它改成 `device.connstring = "pn532_i2c:/dev/i2c-1"`
 6. 如果执行 `nfc-list`，提示 `NFC device: opened`，则接口正常工作。

nfc-tools SPI 配置

为了可以让树莓派挂载更多 SPI 设备，PN532 NFC HAT 并不使用树莓派的 CE0 作为片选管脚，而是使用 D4（BCM）作为片选管脚。而 nfc-tools 的 SPI 功能使用 CE0 作为片选。因此，如果要让 nfc-tools 使用 SPI 接口和模块进行通讯的话，还需要连接 D4 和 CE0。不过且慢！连接 D4 和 CE0 之前，还需要进行必要的工作，以便让树莓派接口安全地工作。

1. 执行 `gpio readall` 查看 D4 管脚的状态。如下图红框所示。

-Pi 3-											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALT0	1	3	4		5v			
3	9	SCL.1	ALT0	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	1	ALT5	TxD	15	14
		0v			9	10	1	ALT5	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	ALT0	1	19	20		0v			
9	13	MISO	ALT0	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	ALT0	0	23	24	1	OUT	CE0	10	8
		0v			25	26	1	OUT	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	1	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	1	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
-Pi 3-											

2. **Mode** 必须为 **IN**, 不能是 **OUT**! 请务必注意, 以免损坏树莓派。只有当 **Mode** 设置为 **IN** 的时候, **D4** 管脚才会工作于高阻态。如果 **Mode** 是 **OUT** 的话, 请依次执行:

```
python3
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)
exit()
```

3. 连接 **D4** 和 **CE0**。
4. 通过跳线帽, 把 **I0** 设置为 **L**, **I1** 设置为 **H**。
5. 用跳线帽连接 **RSTPDN** → **D20**。
6. 拨码开关设置为

SCK	MISO	MOSI	NSS	SCL	SDA	RX	TX
ON	ON	ON	ON	OFF	OFF	OFF	OFF

即:

- SCK → 树莓派 SCK
- MISO → 树莓派 MISO
- MOSI/SDA/TX → 树莓派 MOSI
- NSS/SCL/RX → 树莓派 D4

7. 执行 `sudo raspi-config` 启用 **SPI** 接口:

- Interfacing Options → SPI
 - Would you like the SPI interface to be enabled? → Yes
8. 创建 `/etc/nfc/libnfc.conf`
`sudo mkdir /etc/nfc/`
`sudo cp libnfc/libnfc.conf.sample /etc/nfc/libnfc.conf`
 9. 编辑 `/etc/nfc/libnfc.conf`
`sudo nano /etc/nfc/libnfc.conf`
 删除 `device.connstring = "pn532_uart:/dev/ttyUSB0"` 这一行的 “#”。并把它改成 `device.connstring = "pn532_spi:/dev/spidev0.0"`
 10. 如果执行 `nfc-list`, 提示 `NFC device: opened`, 则接口正常工作。

nfc-tools 例子

1. 列出卡。
`nfc-list`
 首先把卡贴近 PN532 NFC HAT, 然后再执行此命令, 可读取卡 ID、识别卡类型。
2. 导出卡。
`mfoc -O output.mfd`
 用于把卡导出, 该命令的 `output.mfd` 为保存的文件名, 可根据需要修改。如果卡的某些块加密了, 则该命令会穷举常用的密码, 这种情况下不保证可以顺利读取卡的内容。
3. 写入整张卡。
`nfc-mfclassic w a f input.mfd`
 用于把 `dump` 出的卡内容写入另外一张卡片中。该命令的 `input.mfd` 为导入的卡内容, 根据实际情况修改。
4. 设置 UID。
`nfc-mfsetuid 01234567`
 用于修改魔法卡的 UID (第 0 块), `01234567` 为四字节的 UID, 用户应根据需要进行设置。

注意:

 - a) 只能用于魔法卡, 且该卡必须支持后门命令: `HALT, 0x40, 0x43`。

修改成功的话, 终端提示:

```
Sent bits:    50 00 57 cd
Sent bits:    40 (7 bits)
Received bits: a (4 bits)
Sent bits:    43
Received bits: 0a
Card unlocked
Sent bits:    a0 00 5f b1
Received bits: 0a
```

```
Sent bits:    01 23 45 67 00 08 04 00 46 59 25 58 49 10
23 02 23 eb
Received bits: 0a
```

否则终端提示

```
Sent bits:    50 00 57 cd
Sent bits:    40 (7 bits)
Warning: Unlock command [1/2]: failed / not acknowledged.
Sent bits:    a0 00 5f b1
Sent bits:    01 23 45 67 00 08 04 00 46 59 25 58 49 10
23 02 23 eb
```

b) 魔法卡可能会被某些读卡器识别，而不对其进行响应。虽然，这不会损坏魔法卡。

5. 格式化卡：

```
nfc-mfclassic f A u dummy.mfd dummy.mfd f
```

dummy.mfd 是空卡的数据文件。

6. 解锁。在一些情况下，用户可会把卡锁死（例如写入了错误的访问控制位）。这个时候可以用这个语句强制格式化：

```
nfc-mfclassic W A u dummy.mfd dummy.mfd f
```

仅对魔法卡有效。

其他的工具的用法请查阅：<http://nfc-tools.org/index.php?title=Libnfc>

工具的名称可以在 libnfc/utils 目录下找到，例如：nfc-barcode、nfc-emulate-forum-tag4、nfc-list、nfc-mfclassic 等等。

另外，nfc-tools 还提供了一些示例程序，位于 libnfc/examples 目录。工具的用法请参见：

<http://nfc-tools.org/index.php/Category:Examples>