

LinuxAPI 使用手册

1. 前言	2
2. 系统要求	2
3. API 列表	2
4. API 详细说明	1
4.1. 设备类 API	1
4.1.1. ICC_Reader_Open	1
4.1.2. ICC_Reader_Close	1
4.1.3. ICC_PosBeep	1
4.2. 读卡信息类 API	2
4.2.1. readTSi	2
4.2.2. readSFZWeb	2
4.2.3. ICC_ScanCode	3
4.2.4. ReadBank	3
4.2.5. ReadMKWeb	4
4.2.6. RCKWeb	4
4.2.7. iReadCardBas	4
4.2.8. iVerifyPIN	5
4.2.9. iChangePIN	5
4.2.10. iReadIdentityCard	5
4.2.11. iReadCardBas_HSM_Step1	6
4.2.12. iReadCardBas_HSM_Step2	6
4.2.13. Read15693KWeb	7
4.3. 15693 卡操作类 API	7
4.3.1. PICC_Reader_Inventory	7
4.3.2. PICC_Reader_15693_Read	7
4.3.3. PICC_Reader_15693_Write	8
4.3.4. PICC_Reader_API	8
4.3.5. PICC_Reader_DSFID	8
4.3.6. PICC_Reader_LockDataBlock	9
4.3.7. PICC_Reader_SystemInfor	9
4.4. 其他卡操作类 API	9
4.4.1. ICC_Reader_pre_PowerOn	9
4.4.2. ICC_Reader_hot_PowerOn	10
4.4.3. ICC_Reader_PowerOn	10
4.4.4. ICC_Reader_PowerOff	10
4.4.5. ICC_Reader_GetStatus	10
4.4.6. ICC_Reader_Application	11
4.4.7. PICC_Reader_SetTypeA	11
4.4.8. PICC_Reader_Request	11
4.4.9. PICC_Reader_anticoll	12
4.4.10. PICC_Reader_Select	12
4.4.11. PICC_Reader_PowerOnTypeA	12

4.4.12. <i>PICC_Reader_Application</i>	12
4.4.13. <i>PICC_Reader_Authentication_Pass</i>	13
4.4.14. <i>PICC_Reader_Read</i>	13
4.4.15. <i>Rcard1</i>	13
4.4.16. <i>PICC_Reader_ID_Request</i>	14
5. API 调用说明	14
5.1. 例子程序(以 C 语言为例,文件名 T.C).....	14
6. 卡片操作要点	19
6.1. 接触 CPU 卡	19
6.2. TYPEA CPU 卡	19
6.3. M1 卡	19

1. 前言

本应用程序接口(API)用于读卡器应用系统的开发。

2. 系统要求

使用本 API 的系统，必须满足下列条件：

- Linux
- 至少 32 兆内存（32M RAM or Larger）
- 至少 10 兆空闲硬盘空间（10M Free Hard Disk Space or Larger）
- 至少一个空闲普通串口或 USB 口（视用户需求而定）。

3. API 列表

API 分为下列几类，在下面各表中列出。

设备指令类API		
序号	函数名	功能描述
1.	ICC_Reader_Open	初始化设备
2.	ICC_Reader_Close	关闭设备

读卡信息类 API		
序号	函数名	功能描述
3.	readTSi	读社保卡信息
4.	readSFZWeb	读身份证信息，json 格式
5.	scanWeb	扫二维码信息，json 格式
6.	ReadYHKWeb	读银行卡信息，json 格式
7.	ReadMKWeb	读 M1 卡信息，json 格式
8.	ReadCKWeb	读磁条卡信息，json 格式
9.	iReadCardBas	读基本信息
10.	iVerifyPIN	校验 PIN
11.	iChangePIN	修改 PIN
12.	iReadIdentityCard	读身份证信息

4. API 详细说明

4.1. 设备类API

4.1.1.ICC_Reader_Open

根据指定端口初始化设备通讯。

```
int ICC_Reader_Open(char* dev);
```

参数说明：

dev [in] “USB1”。

返回值：

>0	成功
<=0	打开错误

4.1.2.ICC_Reader_Close

关闭设备通讯。

```
int ICC_Reader_Close (long ReaderHandle);
```

参数说明：

ReaderHandle [in]设备句柄。

返回值：

无。

4.1.3.ICC_PosBeep

打开蜂鸣

```
long ICC_PosBeep(long ReaderHandle, unsigned char time);
```

参数说明：

ReaderHandle [in]设备句柄。

time [in] 蜂鸣时间

返回值:

0 成功
其他 失败

4.2. 读卡信息类API

4.2.1.readTSi

读社保卡信息。

```
int readTSi(long ReaderHandle, char *pOutInfo, char *pAreaCode, char *pSsn, char *pCardno, char *pIcno, char *pName, char *pResetinfo, char *pVer, char *pDate, char *pTerm, char *pPsamno, char *pJsno);
```

参数说明:

ReaderHandle [in] 设备句柄

pOutInfo [out] char 型数组, 100 用于保存返回的社保卡错误信息

pAreaCode [out] char 型数组, 100 用于保存返回的发卡地区行政区划代码

pSsn [out] char 型数组, 100 用于保存返回的社会保障号码

pCardno [out] char 型数组, 100 用于保存返回的社保卡卡号

pIcno [out] char 型数组, 100 用于保存返回的卡识别码

pName [out] char 型数组, 100 用于保存返回的姓名

pResetinfo [out] char 型数组, 100 用于保存返回的卡复位信息

pVer [out] char 型数组, 100 用于保存返回的规范版本

pDate [out] char 型数组, 100 用于保存返回的发卡日期

pTerm [out] char 型数组, 100 用于保存返回的卡有效期

pPsamno [out] char 型数组, 100 用于保存返回的 PSAM 卡号

pJsno [out] char 型数组, 100 用于保存返回的省内人员编码

返回值:

0 成功
其他 失败

4.2.2.readSFZWeb

读身份证信息。

```
int readSFZWeb(long ReaderHandle, char* pCertType, char* pName, char* pSex, char* pNation, char* pBirthday, char* pAddress, char* pCardno, char* pAgency, char* pSigndate, char* pVaildterm, char* pENfullname, char* pNationality, char* pCertVersion, char* pPassNu, char* pSignCount);
```

参数说明:

ReaderHandle [in] 设备句柄

pCertType [out] char 型数组, 100, 当返回为”I”时, 读取外国人永久居留身份证; 返回为”J”时, 读取港澳台居住证, 其他则读取二代身份证

pName [out] char 型数组, 100, 用于保存返回的姓名

pSex [out] char 型数组, 100, 用于保存返回的性别

pNation [out] char 型数组, 100, 用于保存返回的民族(二代证 港澳台居住证有效)

pBirthday[out] char 型数组, 100, 用于保存返回的出生日期

pAddress [out] char 型数组, 100, 用于保存返回的住址(二代证 港澳台居住证有效)

pCardno [out] char 型数组, 100, 用于保存返回的身份证号

pAgency [out] char 型数组, 100, 用于保存返回的签发机关

pSigndate [out] char 型数组, 100, 用于保存返回的有效期限起始日期

pVaildterm [out] char 型数组, 100, 用于保存返回的有效期限结束日期

pENfullname [out] char 型数组, 100, 用于保存返回的英文名称(外国人永久居留身份证有效)

pNationality [out] char 型数组, 100, 用于保存返回的国籍(外国人永久居留身份证有效)

pCertVersion [out] char 型数组, 100, 用于保存返回的证件版本号(外国人永久居留身份证有效)

pPassNu [out] char 型数组, 100, 用于保存返回的通行证号(港澳台居住证有效)

pSignCount [out] char 型数组, 100, 用于保存返回的签发次数(港澳台居住证有效)

返回值:

0	成功
其他	失败

4.2.3.ICC_ScanCode

扫码信息

```
int ICC_ScanCode(long ReaderHandle, unsigned char* buf, int* iLen);
```

参数说明:

ReaderHandle [in] 设备句柄

buf [out] char 型数组, 1024 用于保存返回的扫码信息

iLen [out] int 型数组, 100 用于保存返回扫码的长度

返回值:

4	成功
其他	失败

4.2.4.ReadBank

读银行卡信息

```
int ReadBank (long ReaderHandle, char*carddata);
```

参数说明:

ReaderHandle [in] 设备句柄

carddata [out] char 型数组, 1024 用于保存返回的银行卡信息

返回值:

0	成功
其他	失败

4.2.5. ReadMKWeb

测试读 M1 卡信息

```
int ReadMKWeb(long ReaderHandle, char* carddata);
```

参数说明:

ReaderHandle [in] 设备句柄

carddata [out] char 型数组, 1024 用于保存返回的 M1 卡信息

返回值:

0	成功
其他	失败

4.2.6. RCKWeb

测试读磁条卡信息

```
int RCKWeb(long ReaderHandle, char* carddata);
```

参数说明:

ReaderHandle [in] 设备句柄

carddata [out] char 型数组, 1024 用于保存返回的磁条卡信息

返回值:

0	成功
其他	失败

4.2.7. iReadCardBas

读基本信息

```
long iReadCardBas(int iType, char* pOutInfo);
```

参数说明:

iType[in] 操作卡类型 4

1-对接触卡操作; 2-对非接触卡操作; 3-自动寻卡, 接触卡优先; 4-自动寻卡, 非接触卡优先。

pOutInfo [out] char 型数组, 1024 用于读出数据或返回错误信息

返回值:

0	成功
其他	失败

4.2.8. iVerifyPIN

PIN 校验

```
long iVerifyPIN(int iType, char* pOutInfo);
```

参数说明:

iType[in] 操作卡类型 4

1-对接触卡操作; 2-对非接触卡操作; 3-自动寻卡, 接触卡优先; 4-自动寻卡, 非接触卡优先。

pOutInfo [out] char 型数组, 512 当成功时返回空字符串, 当失败时返回错误信息

返回值:

0	成功
其他	失败

4.2.9. iChangePIN

PIN 修改

```
long iChangePIN(int iType, char* pOutInfo);
```

参数说明:

iType[in] 操作卡类型 4

1-对接触卡操作; 2-对非接触卡操作; 3-自动寻卡, 接触卡优先; 4-自动寻卡, 非接触卡优先。

pOutInfo [out] char 型数组, 512 当成功时返回空字符串, 当失败时返回错误信息

返回值:

0	成功
其他	失败

4.2.10. iReadIdentityCard

读取身份证信息

```
long iReadIdentityCard(char *pOutInfo, char *pErrMsg);
```

参数说明:

pOutInfo [out] char 型数组, 2048 用于读取身份证信息

参数包含内容为:

姓名|性别|民族|出生日期|地址|身份证号码|签发机关|签发日期|有效期限|

pErrMsg [out] char 型数组, 200 用于读取错误信息

返回值：

0 成功
其他 失败

4.2.11. iReadCardBas_HSM_Step1

基于加密机的读基本信息（步骤一）

```
long iReadCardBas_HSM_Step1 (int iType, char* pOutInfo);
```

参数说明：

iType [int] 表示执行本函数时操作卡的类型，定义如下： 1-对接触卡操作； 2-对非接触卡操作； 3-自动寻卡，接触卡优先； 4-自动寻卡，非接触卡优先。

pOutInfo [out] char 型数组，1024 用于读取数据或返回错误信息

当函数执行成功时，该输出参数为读出的社保卡内部认证和外部认证（3.0 卡读个人基本信息需要 RKsse 密钥验证）的计算数据，依次为：发卡地区行政区划代码（卡识别码前 6 位）|卡复位信息（历史字节）|算法标识|卡识别码|内部认证过程因子(Random2)|内部认证鉴别所需的原始信息(Random1)|外部认证过程因子(random4)|外部认证鉴别所需的原始信息(random3)|，其中外部认证相关数据项全部不为空或全部为空。

返回值：

0 成功
其他 失败

4.2.12. iReadCardBas_HSM_Step2

基于加密机的读基本信息（步骤二）

```
long iReadCardBas_HSM_Step2 (char* pKey, char* pOutInfo);
```

参数说明：

pKey [int] char 型数组，256，输入的数据为加密机返回的内部认证和外部认证结果（1） 输入参数 pKey 16+16+|+16+16+|

加密机返回的内部认证和外部认证结果，即内部认证鉴别数据（16 位，内部认证过程因子分散后加密内部认证原始信息的密文）+内部认证鉴别所需的原始信息（16 位）|外部认证鉴别数据（16 位，外部认证过程因子分散后加密外部认证原始信息的密文）+外部认证鉴别所需的原始信息（16 位）|。

注：如果没有外部认证，则后两个参数都为||，即连续两个空字符串。

pOutInfo [out] char 型数组，1024 用于读取数据或返回错误信息

当函数执行成功时，该输出参数为读出的社保卡基本信息各数据项，依次为：发卡地区行政区划代码（卡识别码前 6 位）、社会保障号码、社保卡卡号、卡识别码、姓名、卡复位信息（历史字节）、规范版本、发卡日期、卡有效期、终端机编号、终端设备号。各数据项之间以“|”分割，且最后一个数据项之后也必须 有 “|”。例如：

639900|111111198101011110|X00000019|639900D15600000500BF7C7A48FB4966|
 张三|00814E43238697159900BF7C7A |1.00|20151001|20251001|410100813475|
 终端设备号|。

当函数执行失败时，该输出参数为错误信息描述。

返回值：

0	成功
其他	失败

4.2.13. Read15693KWeb

测试读取 15693 卡信息

```
int Read15693KWeb(long ReaderHandle, char* webJson);
```

参数说明：

ReaderHandle [int] 设备句柄

webJson [out] char 型数组，128 用于读取 15693 卡信息或获取错误信息

返回值：

0	成功
其他	失败

4.3. 15693卡操作类API

4.3.1. PICC_Reader_Inventory

通过防冲突用于得到读卡区域内所有卡片的序列号

```
long PICC_Reader_Inventory(long ReaderHandle, unsigned char* Response_APDU);
```

参数说明：

ReaderHandle [in] 设备句柄

Response_APDU [out] char 型数组，1024，返回读卡区域内所有卡片的序列号信息

返回值：

0	成功
其他	失败

4.3.2. PICC_Reader_15693_Read

读取 1 个扇区的值

```
long PICC_Reader_15693_Read(long ReaderHandle, unsigned char blk_add, unsigned char* Response_APDU)
```

参数说明：

ReaderHandle [in] 设备句柄

blk_add [int] unsigned char 型, 10, 输入要读取的扇区号

Response_APDU [out] unsigned char 型数组, 256 返回读取的扇区的数据

返回值:

0 成功

其他 失败

4.3.3. PICC_Reader_15693_Write

对一个块进行写操作（每次只能写一个块）

```
long PICC_Reader_15693_Write(long ReaderHandle,unsigned char blk_add,unsigned char* data,
                             unsigned char* Response_APDU);
```

参数说明:

ReaderHandle [in] 设备句柄

blk_add [in] unsigned char 型, 10, 输入要写入的块号

data [in] unsigned char 型数组, 8, 输入要写入的数据

Response_APDU [out] unsigned char 型数组, 128, 返回写入的数据

返回值:

0 成功

其他 失败

4.3.4. PICC_Reader_API

Data[0] 0 代表写 API, 1 代表锁 API

```
long PICC_Reader_API(long ReaderHandle,unsigned char* data,unsigned char* Response_APDU);
```

参数说明:

ReaderHandle [in] 设备句柄

data [int] unsigned char 型数组, 10, Data[0] 0 代表写 API, 1 代表锁 API

Response_APDU [out] unsigned char 型数组, 128

返回值:

0 成功

其他 失败

4.3.5 PICC_Reader_DSFID

Data[0] 0 代表写 DSFID, 1 代表锁 DSFID

```
long PICC_Reader_DSFID(long ReaderHandle,unsigned char* data,unsigned char* Response_APDU);
```

参数说明:

ReaderHandle [in] 设备句柄

data [int] unsigned char 型数组, 10, Data[0] 0 代表写 DSFID, 1 代表锁 DSFID

Response_APDU [out] unsigned char 型数组，128

返回值：

0 成功
其他 失败

4.3.6. PICC_Reader_LockDataBlock

锁定块内容。注意：此过程不可逆（不能解锁）块锁定后内容不能在修改

long PICC_Reader_LockDataBlock(long ReaderHandle,unsigned char blk_add,unsigned char* Response_APDU);

参数说明：

ReaderHandle [in] 设备句柄
blk_add [in] unsigned char 型，10 ，要锁定的快号
Response_APDU [out] unsigned char 型数组，128

返回值：

0 成功
其他 失败

4.3.7. PICC_Reader_SystemInfor

读取卡的详细信息

long PICC_Reader_SystemInfor(long ReaderHandle,unsigned char *Response_APDU);

参数说明：

ReaderHandle [in] 设备句柄
Response_APDU [out] unsigned char 型数组，1024 返回卡的详细信息

返回值：

0 成功
其他 失败

4.4. 其他卡操作类API

4.4.1. ICC_Reader_pre_PowerOn

接触 CPU 卡冷上电复位

long ICC_Reader_pre_PowerOn(long ReaderHandle,unsigned char ICC_Slot_No,unsigned char* Response);参数说明：

ReaderHandle [in] 设备句柄
ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座
Response [out] unsigned char 型数组，256 返回卡片上电的数据

返回值：

大于 0 成功 返回冷复位数据长度
其他 失败

4.4.2. ICC_Reader_hot_PowerOn

接触 CPU 卡热上电复位

long ICC_Reader_hot_PowerOn(long ReaderHandle,unsigned char ICC_Slot_No,unsigned char* Response);参数说明:

ReaderHandle [in] 设备句柄

ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座

Response [out] unsigned char 型数组, 256 返回卡片上电的数据

返回值:

大于 0 成功 返回热复位数据长度
其他 失败

4.4.3. ICC_Reader_PowerOn

接触 CPU 卡上电复位（冷+热）

long ICC_Reader_PowerOn(long ReaderHandle,unsigned char ICC_Slot_No,unsigned char* Response);参数说明:

ReaderHandle [in] 设备句柄

ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座

Response [out] unsigned char 型数组, 256 返回卡片上电的数据

返回值:

大于 0 成功 返回卡片数据长度
其他 失败

4.4.4. ICC_Reader_PowerOff

接触 CPU 卡下电

long ICC_Reader_PowerOff(long ReaderHandle,unsigned char ICC_Slot_No);

参数说明:

ReaderHandle [in] 设备句柄

ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座

返回值:

0 成功
其他 失败

4.4.5. ICC_Reader_GetStatus

获取 CPU 卡座状态

```
long ICC_Reader_GetStatus (long ReaderHandle,unsigned char ICC_Slot_No);
```

参数说明：

ReaderHandle [in] 设备句柄

ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座

返回值：

0 表示有卡上电, -3 表示有卡未上, -2 表示无卡。

其他 失败

4.4.6. ICC_Reader_Application

执行 APDU 命令

```
long ICC_Reader_Application(long ReaderHandle,unsigned char ICC_Slot_No,long Lenth_of_Command_APDU,unsigned char* Command_APDU,unsigned char* Response_APDU);
```

参数说明：

ReaderHandle [in] 设备句柄

ICC_Slot_No [in] 卡座号 0x01:大卡座, 0x02:副卡, 0x11:SAM1 卡座, 0x12:SAM2 卡座, 0x13:SAM3 卡座, 0x14:SAM4 卡座

Lenth_of_Command_APDU [in] 下发 APDU 命令长度

Command_APDU [in] 下发 APDU 命令

Response_APDU [out] 执行 APDU 响应数据

返回值：

大于 0 成功

其他 失败

4.4.7. PICC_Reader_SetTypeA

设置读卡器读 TypeA 卡

```
int PICC_Reader_SetTypeA(long ReaderHandle);
```

参数说明：

ReaderHandle [in] 设备句柄

返回值：

0 成功

其他 失败

4.4.8. PICC_Reader_Request

TypeA、M1 请求卡片

```
int PICC_Reader_Request(long ReaderHandle);
```

参数说明：

ReaderHandle [in] 设备句柄

返回值：

0 成功

其他 失败

4.4.9. PICC_Reader_anticoll

读卡器防碰撞

```
int PICC_Reader_anticoll(long ReaderHandle, unsigned char *uid);
```

参数说明：

ReaderHandle [in] 设备句柄

uid [out] unsigned char 型数组，10 卡片序列号

返回值：

0 成功

其他 失败

4.4.10. PICC_Reader_Select

读卡器选择卡片（TypeA 及 M1 卡）

```
int PICC_Reader_Select(long ReaderHandle);
```

参数说明：

ReaderHandle [in] 设备句柄

返回值：

0 成功

其他 失败

4.4.11. PICC_Reader_PowerOnTypeA

非接 TypeA CPU 卡上电复位

```
int PICC_Reader_PowerOnTypeA(long ReaderHandle, unsigned char* Response);
```

参数说明：

ReaderHandle [in] 设备句柄

Response [out] unsigned char 型数组，128 返回卡片的复位信息

返回值：

大于 0 成功 返回参数 Response 的长度

其他 失败

4.4.12. PICC_Reader_Application

执行 APDU 命令

```
int PICC_Reader_Application(long ReaderHandle, long Lenth_of_Command_APDU, unsigned char* Command_APDU, unsigned char* Response_APDU);
```

参数说明：

ReaderHandle [in] 设备句柄

Lenth_of_Command_APDU [in] 下发 APDU 命令长度

Command_APDU [in] 下发 APDU 命令

Response_APDU [out] 执行 APDU 响应数据

返回值:

大于 0 成功
其他 失败

4.4.13. PICC_Reader_Authentication_Pass

M1 卡认证密钥，该函数将使用参数 PassWord 传入的 Key 进行认证

```
int PICC_Reader_Authentication_Pass(long ReaderHandle,unsigned char Mode, unsigned char  
SecNr,unsigned char *PassWord);
```

参数说明:

ReaderHandle [in] 设备句柄

Mode [in] unsigned char 型, 认证模式 =0x60 认证 KeyA =0x61 认证 KeyB

SecNr [in] unsigned char 型, 扇区号 (0~15)

PassWord [in] unsigned char 型数组, 10 密钥

例子: unsigned char PassWord [10]={0xff,0xff,0xff,0xff,0xff,0xff};

返回值:

0 成功
其他 失败

4.4.14. PICC_Reader_Read

读取 M1 卡块数据

```
int PICC_Reader_Read(long ReaderHandle,unsigned char Addr,unsigned char *OutReport);
```

参数说明:

ReaderHandle [in] 设备句柄

Addr [in] unsigned char 型 块号 S50 该值的范围为 0~63, S70 该值范围为 0~255

OutReport [out] unsigned char 型数组, 128 读取数据

返回值:

0 成功
其他 失败

4.4.15. Rcard1

读磁条卡

```
int Rcard1(long ReaderHandle,unsigned char ctime,int track,unsigned char *t_rlen,unsigned char  
*getdata);
```

参数说明:

ReaderHandle [in] 设备句柄

ctime [in] unsigned char 型 超时时间

track [in] int 型 磁道 =0x01 一磁道 =0x02 二磁道 =0x03 三磁道 =0x04 四磁道

rlen [out] unsigned char 型数组, 10 返回数据长度

getdata [out] unsigned char 型数组, 1024 返回磁条轨道数据

返回值:

0 成功
其他 失败

4.4.16. PICC_Reader_ID_Request

身份证寻卡

long PICC_Reader_ID_Request(long ReaderHandle);

参数说明:

ReaderHandle [in] 设备句柄

返回值:

0 成功
其他 失败

5. API 调用说明

5.1. 例子程序(以C语言为例,文件名t.c)

```
#include<stdio.h>
#include <dlfcn.h>
#include <stdlib.h>
#include "icreader.h"
#include <time.h>
#include <string.h>

void callback(){
    int re = 0;
    int i = 0;
    long ReaderHandle = 0;

    unsigned char buf[100] = {0};
    unsigned char uid[100] = { 0 };
    char pName[100] = {0};
    char pSex[100] = { 0 };
    char pNation[100] = { 0 };
    char pBirth[100] = { 0 };
    char pAddress[100] = { 0 };
    char pCertNo[100] = { 0 };
    char pDepartment[100] = { 0 };

    char pExpire[100] = { 0 };
    unsigned char Response[1025] = { 0 };
    char pErrMsg[100] = { 0 };
    char Data[1024] = {0};
```

```
int iArr[1] = {0};
int iType=0;
char pAreaCode[100]={0};
char pSsn[100]={0};
char pCardno[100]={0};
char pIcno[100]={0};
char pName1[100]={0};
char pResetinfo[100]={0};
char pVer[100]={0};
char pDate[100]={0};
char pTerm[100]={0};
char pPsamno[100]={0};
char pJsno[100]={0};

char pCertType[10]={0};
char pSigndate[100]={0};
char pVaildterm[100]={0};
char pENfullname[100]={0};
char pNationality[100]={0};
char pCertVersion[100]={0};
char pPassNu[100]={0};
char pSignCount[100]={0};

unsigned char wltData[1025] = { 0 };

ReaderHandle=ICC_Reader_Open("USB1");

//printf("ReaderHandle=%d\n", ReaderHandle);

char carddata[3000] = {0};

while(1)
{
    int a=0;
    int b=0;
    printf("Please Import:: 1.ReadSFZ__2.ReadSBK__3.Scan__4.
ReadYHK__5. ReadM1__6.ReadCTK__7. Exit   \r\n");
    scanf("%d",&a);
    switch(a){
    case 1:
```

```
/*
re=readSFZWeb( ReaderHandle,carddata);
if(re==0)
{
    printf("SFZ:   %s\n", carddata);

}
*/
```

```
re=readSFZWeb( ReaderHandle,pCertType,pName,pSex,pNation,pBirth,pAddress,pCer
tNo,pDepartment,pSigndate,pVaildterm,pENfullname,pNationality,pCertVersion,pPassNu,
pSignCount);
```

```
//printf("CertType: %s\n",pCertType);
if(re==0)
{
    if(!strcmp(pCertType,"I")){

        printf("ForeignSFZ:  \n");
        printf("EnName:   %s\n", pENfullname);
        printf("Sex:    %s\n", pSex);
        printf("Number:   %s\n", pCertNo);
        printf("Nation:   %s\n", pNationality);
        printf("Signdate:   %s\n", pSigndate);
        printf("Vaildterm:   %s\n", pVaildterm);
        printf("Birth:    %s\n", pBirth);
        printf("CertVersion:   %s\n", pCertVersion);
        printf("Department:   %s\n", pDepartment);

    }
    else if(!strcmp(pCertType,"J")){
        printf("GanaoTaJuzhuzheng:  \n");
        printf("Name:    %s\n", pName);
        printf("Sex:    %s\n", pSex);
        printf("Birth:   %s\n", pBirth);
        printf("Address:  %s\n", pAddress);
        printf("Certno:   %s\n", pCertNo);
        printf("Department:   %s\n", pDepartment);
        printf("Signdate:   %s\n", pSigndate);
        printf("Vaildterm:   %s\n", pVaildterm);
        printf("PassNu:    %s\n", pPassNu);
        printf("SignCount:   %s\n", pSignCount);

    }
    else{
```

```
        printf("TwoDaiSFZ:  %s\n", carddata);
        printf("Name:  %s\n", pName);
        printf("Sex:  %s\n", pSex);
        printf("Nation:  %s\n", pNation);
        printf("Birth:  %s\n", pBirth);
        printf("Address:  %s\n", pAddress);
        printf("Certno:  %s\n", pCertNo);
        printf("Department:  %s\n", pDepartment);
        printf("Signdate:  %s\n", pSigndate);
        printf("Vaildterm:  %s\n", pVaildterm);

    }
}
break;
case 2:
re =
readTSi( ReaderHandle,carddata,pAreaCode,pSsn,pCardno,pIcno,pName1,pResetinfo,pVer
,pDate,pTerm,pPsamno,pJsno);
if(re==0)
{
    printf("AreaCode:  %s\n", pAreaCode);
    printf("Ssn:  %s\n", pSsn);
    printf("Cardno:  %s\n", pCardno);
    printf("Icno:  %s\n", pIcno);
    printf("Name:  %s\n", pName1);
    printf("Resetinfo:  %s\n", pResetinfo);
    printf("Ver:  %s\n", pVer);
    printf("Date:  %s\n", pDate);
    printf("Term:  %s\n", pTerm);
    printf("Psamno:  %s\n", pPsamno);
    printf("Jsno:  %s\n", pJsno);
}
else
{
    printf("SiCard:  %s\n", carddata);
}

break;
case 3:
re=scanWeb(ReaderHandle,Data);
printf("%s\n",Data);
break;
```

```
        case 4:
            re=ReadYHKWeb(ReaderHandle,Data);
            printf("%s\n",Data);
            break;
        case 5:
            re=ReadMKWeb(ReaderHandle,Data);
            if(re==0)
            {
                printf("%s\n",Data);
            }
            else
            {
                printf("%ld\n",re);
            }
            break;
        case 6:
            re=RCKWeb(ReaderHandle,Data);

            printf("%s\n",Data);

            break;
        case 7:
            re=Read15693KWeb(ReaderHandle,Data);

            printf("%s\n",Data);

            break;
        case 8:
            ICC_Reader_Close(ReaderHandle);
            return;
    }
    ICC_Reader_Close(ReaderHandle);
    return NULL;
}
int main() {

    callback();
    return 0;    // exit(0);
}
```

6. 卡片操作要点

6.1. 接触CPU卡

- 1、 上电
- 2、 APDU 命令

6.2. TypeA CPU卡

- 1、 设置为 TypeA 卡片
- 2、 请求卡片
- 3、 防碰撞
- 4、 选择卡片
- 5、 上电
- 6、 APDU 命令

6.3. M1卡

- 1、 请求卡片
- 2、 防碰撞
- 3、 选择卡片
- 4、 认证密钥
- 5、 读或写

```
// gcc  t.c  -o st -Wl,-rpath-link=$$PWD/  -L$PWD/  -ldl  -licreader
```