ZKFinger SDK For Linux

接口开发手册

熵基科技股份有限公司
http://www.zkteco.com

版权声明

1. 目录

2. 前言

本文档会为您提供 SDK 基本的开发指南和技术背景介绍,帮助您更好地使用我们的技术服务。

衷心感谢您对我们技术与产品的信任和支持!

3. ZKFinger10.0 介绍

熵基科技一直专注于指纹识别算法的研究和产业化推广，已将指纹识别系统应用到各个行业中。随着指纹识别系统越来越广泛的应用，市场对指纹识别算法的精确性,适用性和运算速度等多方面提出更高的要求。为满足这些需求，我们从低质量指纹图像的增强,指纹的特征提取、指纹图像的分类与检索及压缩技术、指纹图像匹配算法等多方面进行优化，推出ZKFinger10.0版高速算法。该算法在大规模的数据库上进行了严格的测试，误识率（False Accept Rate, FAR）、拒识率（False Reject Rate, FRR）、拒登率（Error Registration Rate, ERR）等性能都大大提高，对过干、太湿、伤疤、脱皮等低质量的指纹图像处理效果也明显增强，算法比对速度提升了10倍以上。该算法的指纹模板也同时进行了优化存储，与以前的算法版本的指纹模板不兼容。您选择ZKFinger10.0版高速算法后，必须重新登记用户指纹模板。

ZKFinger10.0 算法具有以下特点：

- ZKFinger SDK软件开发包能够快速集成到客户系统中，通过开放图像处理接口，可以支持任何扫描设备和指纹Senor(图像DPI>=300DPI)。
- ZKFinger10.0算法通过自适应的、适合匹配的滤镜和恰当的阈值，减弱图片噪声，增强脊和谷的对比度，甚至能够从质量很差的指纹（脏、刀伤、疤、痕、干燥、湿润或撕破）中获取前档的全局和局部特征点。
- ZKFinger10.0算法支持180°旋转比对。
- ZKFinger10.0算法不需要指纹必须有全局特征点（核心点、三角点等），通过局部特征点就可以完成识别。
- ZKFinger10.0算法通过分类算法（指纹被分成五大类型：拱形、左环类、右环类、尖拱类、漩涡类“斗”），预先使用全局特征排序，从而大大的加速指纹匹配过程。

3. 技术规格

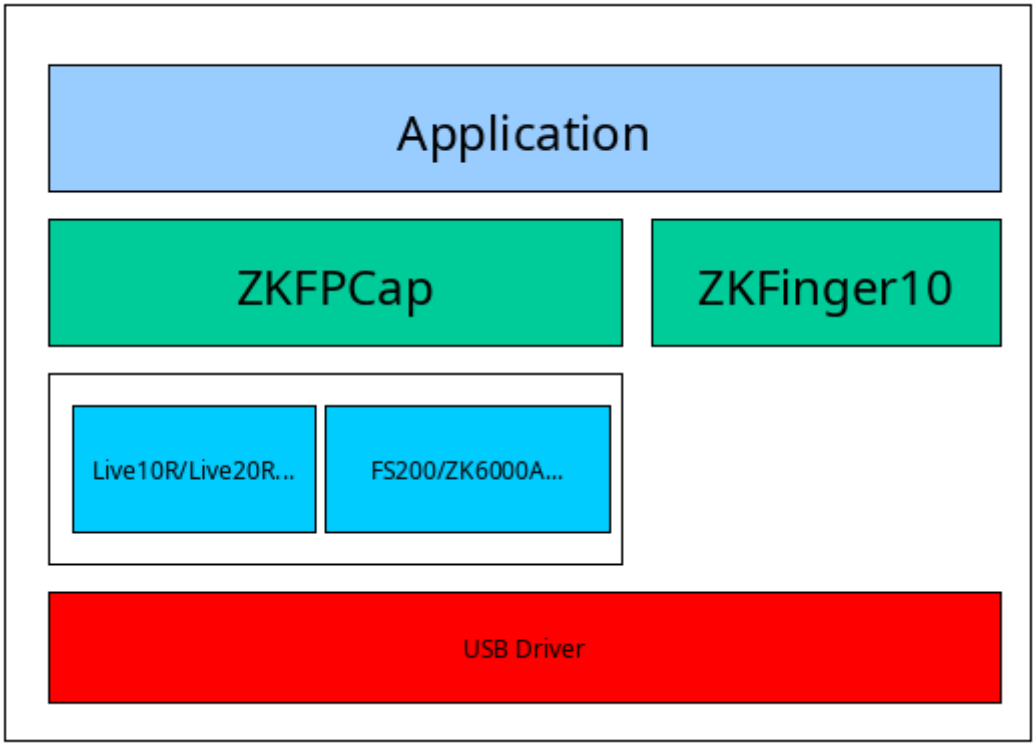
3.1. 开发语言

支持各种主流桌面开发语言(如C、C++、Java(jna)等)

3.2. 平台支持

CPU平台	系统版本
X86_64（Intel/AMD/海光/兆芯等）	Ubuntu12.04+ Deepin OS V15.11+ UOS 20 银河麒麟V4/V10
ARM64(飞腾、鲲鹏等)	UOS 20 麒麟系统V4/V10
MIPS64(龙芯)	UOS 20 麒麟系统V4//V10
其他Linux 系统/CPU等	根据不同系统、不同CPU定制评估适配(大部分客户可以拿以上版本自行验证；有不满足再走定制流程)

3.3. SDK 架构



3.4. 技术参数

参数名	描述
模板大小	<=1664字节
旋转	0~360度
FAR	<=0.001%
FRR	<=1%
比对速度（1:1)	<=50ms
识别速度（1:50,000)	<=500ms
图像DPI	500DPI

3.5. 快速集成

3.5.1. SDK文件列表

文件名	描述	其他备注
libzksensorcore.so	采集器通讯核心库	
libidfprcap.so	免驱指纹采集器采集动态库	
libslkidcap.so	Live10R/Live20R 采集器采集动态库	
libzkfpcap.so	采集器接口动态库	对应文档设备API
libzkfp.so	算法接口动态库	对应文档算法API
libzkalg12.so	ZKFinger10.0算法核心库	
libusb-0.1.so.4	算法核心依赖	仅X86_64需要
libusb-1.0.so.0.1.0	算法核心依赖	仅X86_64需要

3.5.2. 支持设备列表

设备名	VID	PID
Live10R	0x1B55	0x0124
Live20R	0x1B55	0x0120

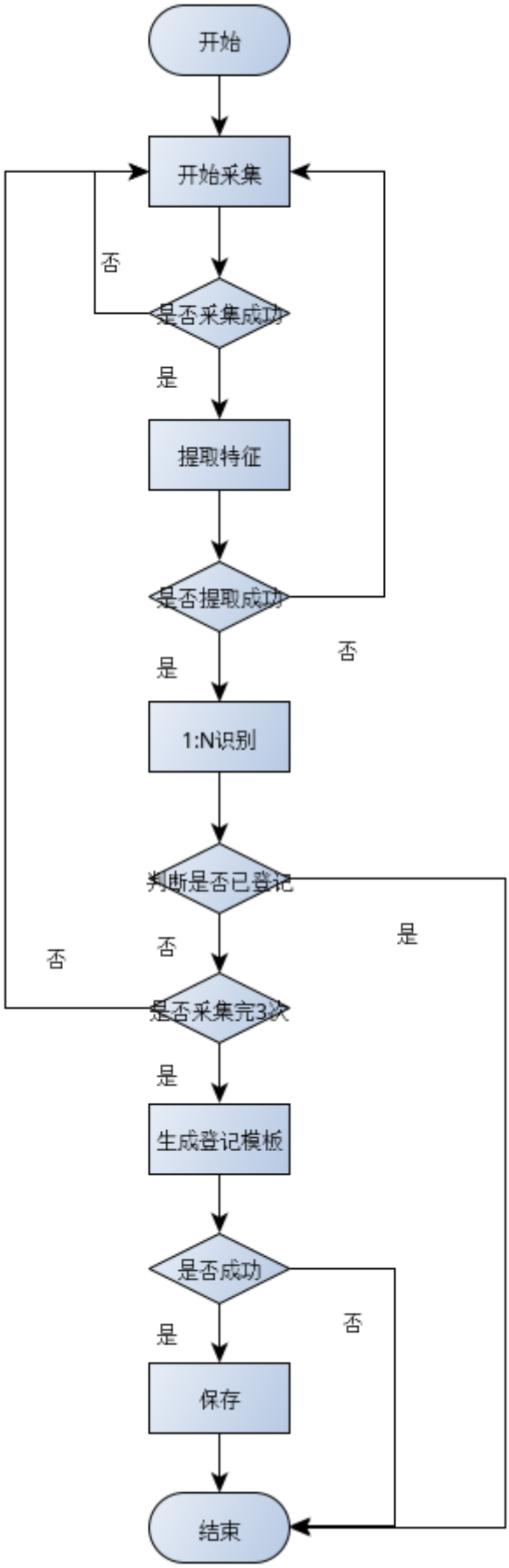
设备名	VID	PID
FS200(-R)	0x1B55	0x0304
FS300(-R)	0x1B55	0x0306
ZK6000A(-R)	0x1B55	0x0308
ZK7000A(-R)	0x1B55	0x0302

3.6. 编程引导

ZKFinger SDK 单独提供采集SDK和算法SDK,其中采集SDK只是3.5.2 设备列表列举的设备；算法SDK版本为ZKFinger10.0，由于算法绑定了采集器，因此使用时需要在调用采集SDK连接设备成功后才可以初始化算法。其中FS200/FS300/ZK6000A/ZK7000A 内置加密芯片需烧写ZKFinger10.0许可才可以使用本算法。

以下将对登记比对流程进行简单介绍。

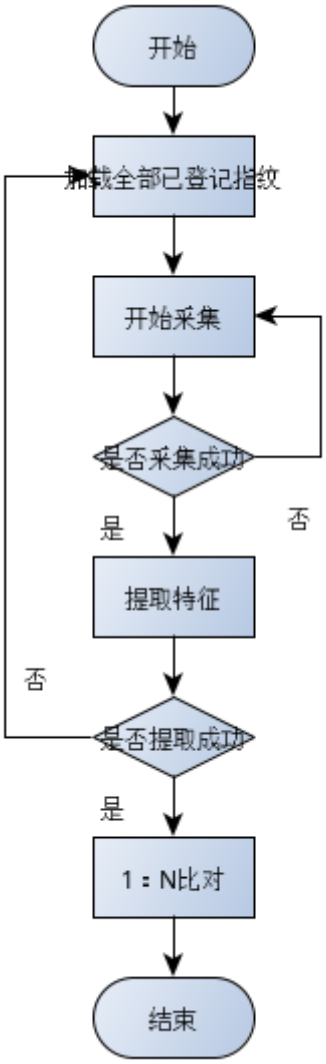
3.6.1. 登记流程



- 调用sensorCapture采集图像
- 当采集图像成功后调用提取模板；失败则继续采集
- 提取模板失败时继续采集图像；提取成功时调用identify判断是否指纹已登记
- 如果手指已登记结束登记过程并提示用户；采集3次指纹模板，不足三次继续采集图像
- 生成登记模板，当失败时结束登记过程并提示用户；成功时保存数据库

- 完成登记过程

3.6.2. 比对流程



- 1:N识别首先需要将已登记的模板都加载到内存中；可在算法初始化成功之后加载
- 调用采集SDK采集图像
- 当采集失败时继续采集；当采集成功时调用算法SDK提取模板
- 当提取模板失败时继续采集图像；当提取模板成功时调用identify比对指纹，并反馈结果给用户
- 完成比对过程

4. SDK接口说明

4.1. 设备接口说明

4.1.1. 功能列表

函数	功能描述
sensorEnumDevices	枚举设备，搜索全部本SDK支持的设备列表
sensorOpen	打开设备
sensorClose	关闭设备
sensorRebootEx	重启设备(打开->重启->关闭)
sensorCapture	采集图像

函数	功能描述
sensorGetParameter(Ex)	获取参数
sensorSetParameter(Ex)	设置参数
sensorReboot	重启设备(设备已打开状态)
sensorStatus	获取设备状态

4.1.2. sensorEnumDevices

```
[函数]
    int sensorEnumDevices(TXUSBDevice deviceList[], int nMaxCount);
[功能]
    枚举设备
[参数]
    deviceList[out]
        返回设备数组
    maxCount[in]
        数组大小
[返回值]
    获取支持的指纹设备数
```

4.1.3. sensorOpen

```
[函数]
    void* sensorOpen(PXUSBDevice device);
[功能]
    打开指纹仪
[参数]
    device[in]
        设备结构体指针
[返回值]
    设备句柄
```

4.1.4. sensorClose

```
[函数]
    int sensorClose(void* handle);
[功能]
    关闭设备
[参数]
    handle[in]
        设备句柄(见sensorOpen)
[返回值]
    0 表示成功
    其他失败
```

4.1.5. sensorRebootEx

[函数]
    int sensorRebootEx(int index);
[功能]
    重启设备(不需要先调用sensorOpen)
[参数]
    index[in]
        设备索引(云桌面版固定传0)
[返回值]
    0 表示成功
    其他失败

4.1.6. sensorCapture

[函数]
    int sensorCapture(void *handle, unsigned char *imageBuffer, int imageBufferSize);
[功能]
    采集指纹图像
[参数]
    handle[in]
        设备句柄(见sensorOpen)
    imageBuffer[out]
        返回raw图像数据(由调用者申请内存，不小于width*height字节)
    imageBufferSize[in]
        imageBuffer 内存大小
[返回值]
    >0 表示取像成功并返回数据长度
    其他失败

4.1.7. sensorGetParameter

[函数]
    int sensorGetParameter(void *handle, int paramCode);
[功能]
    获取简单参数
[参数]
    handle[in]
        设备句柄(见sensorOpen)
    parmaCode[in]
        参数代码
            1：指纹图像宽
            2：指纹图像高
[返回值]
    参数值

4.1.8. sensorSetParameter

[函数]
    int sensorSetParameter(void *handle, int paramCode, int paramValue);
[功能]
    设置参数
[参数]
    handle[in]
        设备句柄(见sensorOpen)
    parmaCode[in]
        参数代码
            5：取像模式

```
    paramValue[in]
        参数值,当paramCode=5, paramValue=0(默认值) 表示探测模式, paramValue=1表示流模式
[返回值]
    0 表示成功
    其他失败
```

4.1.9. sensorGetParameterEx

```
[函数]
    int sensorGetParameterEx(void *handle, int paramCode, char *paramValue, int *paramLen);
[功能]
    获取参数
[参数]
    handle[in]
        设备句柄(见sensorOpen)
    parmaCode[in]
        参数代码
            1: 指纹图像宽(int)
            2: 指纹图像高(int)
            1103: 设备序列号(String)
    paramValue[out]
        返回参数值(由调用者分配内存)
    paramLen[in/out]
        in: paramValue内存大小
        out: 实际返回参数值数据大小
[返回值]
    0 表示成功
    其他失败
[示例]
    //获取指纹图像宽
    int width = 0;
    int retSize = 4;
    int ret = sensorGetParameterEx(handle, 1, (unsigned char*)&width, &retSize);
    //获取设备序列号
    char szSerialNumber[64];
    int retSize = 64;
    int ret = sensorGetParameterEx(handle, 1103, (unsigned char* szSerialNumber, &retSize);
```

4.1.10. sensorSetParameterEx

```
[函数]
    int sensorSetParameterEx(void *handle, int paramCode, char *paramValue, int paramLen);
[功能]
    获取参数
[参数]
    handle[in]
        设备句柄(见sensorOpen)
    parmaCode[in]
        参数代码
            1: 指纹图像宽(int)
            2: 指纹图像高(int)
            1103: 设备序列号(String)
    paramValue[in]
        参数值
    paramLen[in]
        参数值数据长度
```

[返回值]
    0 表示成功
    其他失败
[备注]
    本接口目前不需要使用到

4.1.11. sensorReboot

[函数]
    int sensorReboot(void* handle)
[功能]
    重启设备(需先调用sensorOpen成功)
[参数]
    handle[in]
        设备句柄(见sensorOpen)
[返回值]
    0 表示成功
    其他失败

4.1.12. sensorStatus

[函数]
    int sensorStatus(void* handle)
[功能]
    获取设备状态
[参数]
    handle[in]
        设备句柄(见sensorOpen)
[返回值]
    0 正常
    -99998 SLK20R bulk端点异常，需要重启设备
    其他错误 (连续多次(如5次)失败后，重启设备)

4.2. 算法接口说明

4.2.1. 功能列表

函数	功能描述
BIOKEY_INIT_SIMPLE	初始化算法
BIOKEY_CLOSE	释放算法
BIOKEY_EXTRACT	提取模板(传入图像宽高必须与BIOKEY_INIT_SIMPLE初始化宽高一致)
BIOKEY_GETLASTQUALITY	获取最近一次提取模板的模板质量
BIOKEY_GENTEMPLATE	生成登记模板
BIOKEY_VERIFY	1:1比对
BIOKEY_SET_PARAMETER	设置算法参数
BIOKEY_DB_ADD	添加登记模板到1:N内存中
BIOKEY_DB_DEL	删除模板
BIOKEY_DB_CLEAR	清空模板
BIOKEY_DB_COUNT	获取已添加模板数
BIOKEY_IDENTIFYTEMP	1:N 比对
BIOKEY_EXTRACT_GRAYSCALEDATA	提取模板

4.2.2. BIOKEY_INIT_SIMPLE

[函数]
    void* BIOKEY_INIT_SIMPLE(int License, int width, int height, BYTE *Buffer);
[功能]
    初始化算法
[参数]
    license[in]
        固定传0
    width[in]
        指纹图像宽
    height[in]
        指纹图像高
    Buffer[in]
        固定传NULL
[返回值]
    算法句柄
[备注]
    设备[sensorOpen]连接成功后方可调用算法初始化

4.2.3. BIOKEY_CLOSE

[函数]
    int BIOKEY_CLOSE(HANDLE Handle);
[功能]
    释放算法
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
[返回值]
    1 表示成功
    其余表示失败

4.2.4. BIOKEY_EXTRACT

[函数]
    int BIOKEY_EXTRACT(HANDLE Handle, BYTE* PixelsBuffer, BYTE *Template, int PurposeMode);
[功能]
    提取指纹模板
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    pixelsBuffer[in]
        指纹RAW图像数据(宽高必须与BIOKEY_INIT_SIMPLE一致), 见sensorCapture
    Template[out]
        返回指纹模板数据(建议分配2048字节)
    PurposeMode[in]
        固定传0
[返回值]
    > 0 表示提取成功, 返回模板数据实际长度

4.2.5. BIOKEY_GETLASTQUALITY

[函数]
    int BIOKEY_GETLASTQUALITY(HANDLE Handle);
[功能]
    获取最近一次提取模板的模板质量
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
[返回值]
    模板质量(0~100)
[备注]
    质量不太准确，仅供参考

4.2.6. BIOKEY_GENTEMPLATE

[函数]
    int BIOKEY_GENTEMPLATE(HANDLE Handle, BYTE *Templates[], int TmpCount, BYTE *GTemplate);
[功能]
    生成登记特征(多个模板之中取最好)
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    Templates[in]
        模板数组
    TmpCount[in]
        模板个数
    GTemplate[out]
        返回最好的模板(建议分配2048字节)
[返回值]
    >0 表示成功，值为最好模板的实际数据长度

4.2.7. BIOKEY_VERIFY

[函数]
    int BIOKEY_VERIFY(HANDLE Handle, BYTE *Template1, BYTE *Template2);
[功能]
    比对两个模板
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    Template1[in]
        比对模板数据
    Template2[in]
        比对模板数据
[返回值]
    返回分数(0~1000)，推荐阈值50

4.2.8. BIOKEY_SET_PARAMETER

[函数]
    int BIOKEY_SET_PARAMETER(HANDLE Handle, int ParameterCode, int ParameterValue);
[功能]
    设置参数(不要随意设置)
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    ParameterCode[in]

参数代码
1 表示设置比对阈值
4 表示设置旋转角度
ParameterValue[in]
参数值
[返回值]
1 表示成功
[备注]
除比对阈值/旋转角度外,其他未说明参数代码不要随意设置

4.2.9. BIOKEY_DB_ADD

[函数]
int BIOKEY_DB_ADD(HANDLE Handle, int TID, int TempLength, BYTE *Template);
[功能]
添加模板到1:N内存中
[参数]
Handle[in]
算法句柄(见BIOKEY_INIT_SIMPLE)
TID[in]
模板ID(必须>0)
TempLength[in]
模板数据长度
Template[in]
登记模板数据
[返回值]
>0 表示成功
其他为失败
[备注]
模板只存在内存中, BIOKEY_DB_CLEAR/BIOKEY_CLOSE/程序退出均被释放

4.2.10. BIOKEY_DB_DEL

[函数]
int BIOKEY_DB_DEL(HANDLE Handle, int TID);
[功能]
从内存中删除一个模板
[参数]
Handle[in]
算法句柄(见BIOKEY_INIT_SIMPLE)
TID[in]
模板ID(见BIOKEY_DB_ADD)
[返回值]
1 表示成功

4.2.11. BIOKEY_DB_CLEAR

[函数]
int BIOKEY_DB_CLEAR(HANDLE Handle);
[功能]
清空内存中全部模板
[参数]
Handle[in]
算法句柄(见BIOKEY_INIT_SIMPLE)
[返回值]
1 表示成功

4.2.12. BIOKEY_DB_COUNT

```
[函数]
    int BIOKEY_DB_COUNT(HANDLE Handle);
[功能]
    获取已添加模板的数量
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
[返回值]
    模板数量
```

4.2.13. BIOKEY_IDENTIFYTEMP

```
[函数]
    int BIOKEY_IDENTIFYTEMP(HANDLE Handle, BYTE *Template, int *TID, int *Score);
[功能]
    1:N识别
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    Template[in]
        指纹模板数据
    TID[out]
        返回识别成功的指纹ID
    Score[out]
        返回识别成功的分数(推荐阈值70)
[返回值]
    成功返回1
```

4.2.14. BIOKEY_EXTRACT_GRAYSCALEDATA

```
[函数]
    int APICALL BIOKEY_EXTRACT_GRAYSCALEDATA(void* Handle, unsigned char* PixelsBuffer, int
width, int height, unsigned char* Template, int maxTmpLen, int PurposeMode);
[功能]
    提取特征
[参数]
    Handle[in]
        算法句柄(见BIOKEY_INIT_SIMPLE)
    pixelsBuffer[in]
        指纹RAW图像数据，见sensorCapture
    width[in]
        图像宽
    height[in]
        图像高
    Template[out]
        返回指纹模板数据(建议分配2048字节)
    PurposeMode[in]
        固定传0
[返回值]
    > 0 表示提取成功，返回模板数据实际长度
```

5. 附录

5.1 名词解释

以下的定义将帮助你理解指纹识别应用基本功能,以及帮助快速完成指纹识别应用集成开发。

- **指纹(1:1)比对**

  指纹 （1:1） 比对也叫做“指纹验证”,根据用户 ID 和指纹模板来验证该用户“是不是”符合其身份；或者比对一组登记模板和比对模板是不是来自同一个手指。

- **指纹(1:N)比对**

  指纹 （1:N） 比对也叫做“指纹辨识”,是指在不知道用户 ID 的情况下,仅根据输入的指纹,在指纹数据库中进行检索,返回符合阈值条件的用户名、相似度等信息,得出“有没有”该用户的结论的过程。