

# 鸿蒙操作系统应用与开发

## 实验手册

版本:1.0

---

## 前 言

## 简介

本书为《鸿蒙操作系统应用与开发》实验手册。是 ArkTs 基础程序基于鸿蒙系统的操作实验手册，主要介绍了 ArkTs 基础程序基于鸿蒙系统编译和运行等操作。

## 内容描述

本实验指导书共包含 2 个部分，介绍了 ArkTs 基础程序基于鸿蒙系统的编译、安装和运行等操作。

- ArkTs 搭建基础运行环境；
- ArkTs 程序开发；

## 实验环境说明

### 组网说明

本实验环境面向搭建本地实验环境。实验环境包括 Windows 操作系统或 Mac 操作系统、安装有鸿蒙系统的模拟器。

### 设备介绍

为满足 ArkTs 程序设计实验手册需要，建议每套实验环境采用以下配置。

---

| 名称        | 配置   | OS版本   |
|-----------|--|--|
| Windows系统 | 内存：16GB及以上<br>硬盘：100GB及以上<br>分辨率：1280*800像素及以上 | 操作系统：Windows10 64位、Windows11 64位                   |
| Mac系统     | 内存：8GB及以上<br>硬盘：100GB及以上<br>分辨率：1280*800像素及以上  | 操作系统：macOS(X86) 11/12/13/14<br>macOS(ARM) 12/13/14 |
| 鸿蒙模拟器     | 模拟器默认内存为4G，运行过程中内存不足时，可能会出现模拟器卡顿或者闪退。          | API12及以上   |

### 软件介绍

本实验设计的软件及连接如下表，请提前准备好相关软件（源码中以提供所需工具包）。

| 软件名称          | 使用说明  |
|---------------|---|
| DevEco Studio | <a href="https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-tools-overview-V5">https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-tools-overview-V5</a> |

提示：实验所需环境均已配置完成，请登录各自机器完成实验操作。

# 目录

---

|  |           |
|--|-----------|
| 简介 .....   | I         |
| 内容描述 .....   | I         |
| 实验环境说明 .....                                       | I         |
| <b>1 ArkTs 程序设计实验手册 .....</b>                      | <b>1</b>  |
| <b>1.1 实验介绍 .....</b>                              | <b>1</b>  |
| 1.1.1 关于本实验 .....                                  | 1         |
| 1.1.2 实验目的 .....                                   | 1         |
| 1.1.3 实验规划 .....                                   | 1         |
| <b>1.2 DevEco Studio 开发环境搭建 .....</b>              | <b>1</b>  |
| 1.2.1 安装 DevEco Studio .....                       | 1         |
| 1.2.2 创建模拟器 .....                                  | 2         |
| 1.2.3 开发及运行环境验证 .....                              | 5         |
| <b>1.3 程序开发 - 基于 ArkUI 组件的简易计算器开发 .....</b>        | <b>8</b>  |
| 1.3.1 实验介绍 .....                                   | 8         |
| 1.3.2 代码开发 .....                                   | 9         |
| 1.3.3 代码编译运行 .....                                 | 11        |
| 1.3.4 代码验证调测 .....                                 | 11        |
| 1.3.5 实验总结与课后思考 .....                              | 12        |
| <b>1.4 程序开发 - 基于 NDK 的排序程序开发 .....</b>             | <b>13</b> |
| 1.4.1 实验介绍 .....                                   | 13        |
| 1.4.2 代码开发 .....                                   | 13        |
| 1.4.3 代码验证调测 .....                                 | 14        |
| 1.4.4 实验总结与课后思考 .....                              | 17        |
| <b>1.5 程序开发 - 基于 ArkData 用户首选项的数据持久化程序开发 .....</b> | <b>17</b> |
| 1.5.1 实验介绍 .....                                   | 17        |
| 1.5.2 代码开发 .....                                   | 17        |
| 1.5.3 代码验证调测 .....                                 | 19        |

---

|   |           |
|---|-----------|
| 1.5.4 实验总结与课后思考 .....                                     | 23        |
| <b>1.6 程序开发 - 基于 Basic Services Kit 的应用账号管理程序开发 .....</b> | <b>23</b> |
| 1.6.1 实验介绍 .....  | 23        |
| 1.6.2 代码开发 .....  | 23        |
| 1.6.3 代码验证调测 .....  | 27        |
| 1.6.4 实验总结与课后思考 .....                                     | 31        |
| <b>1.7 程序开发 - 基于系统相关 Kit 的用户身份认证程序开发 .....</b>            | <b>31</b> |
| 1.7.1 实验介绍 .....  | 31        |
| 1.7.2 代码开发 .....  | 31        |
| 1.7.3 代码验证调测 .....  | 35        |
| 1.7.4 实验总结与课后思考 .....                                     | 37        |
| <b>1.8 实验环境清理 .....</b>                                   | <b>37</b> |

---

# 1

# ArkTs 程序设计实验手册

---

## 1.1 实验介绍

### 1.1.1 关于本实验

本实验主要介绍的 ArkTs 程序编译后在鸿蒙系统安装运行。通过本实验,您将能够掌握在 ArkTs 程序的编译,熟悉在鸿蒙系统的安装和运行的查看。

### 1.1.2 实验目的

- 掌握 ArkTs 程序的编写以及编译。
- 熟悉 ArkTs 程序在鸿蒙系统下的开发, 调试和运行。
- 熟悉 NDK。

### 1.1.3 实验规划

本实验需要用到一台安装有 Windows10 64 位或 Windows11 64 位的主机, 要求内存为 16GB 及以上, 推荐为 32GB, 硬盘为 100GB 及以上, 分辨率: 1280\*800 像素及以上。

## 1.2 DevEco Studio 开发环境搭建

### 1.2.1 安装 DevEco Studio

整体步骤参考官网链接:

<https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-tools-overview-V5>

步骤 1 进入下载页面 <https://developer.huawei.com/consumer/cn/download/>, 选择最新版本下载。

---

步骤 2 参考页面进行安装：

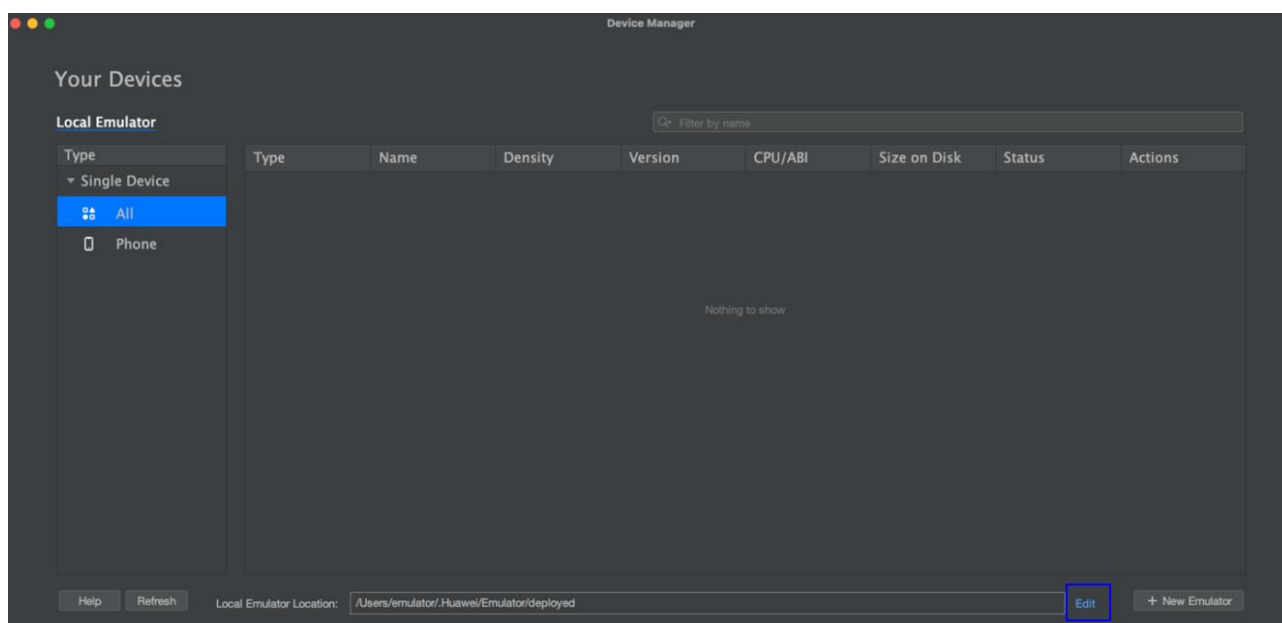
<https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-software-install-V5>

## 1.2.2 创建模拟器

整体步骤参考链接：

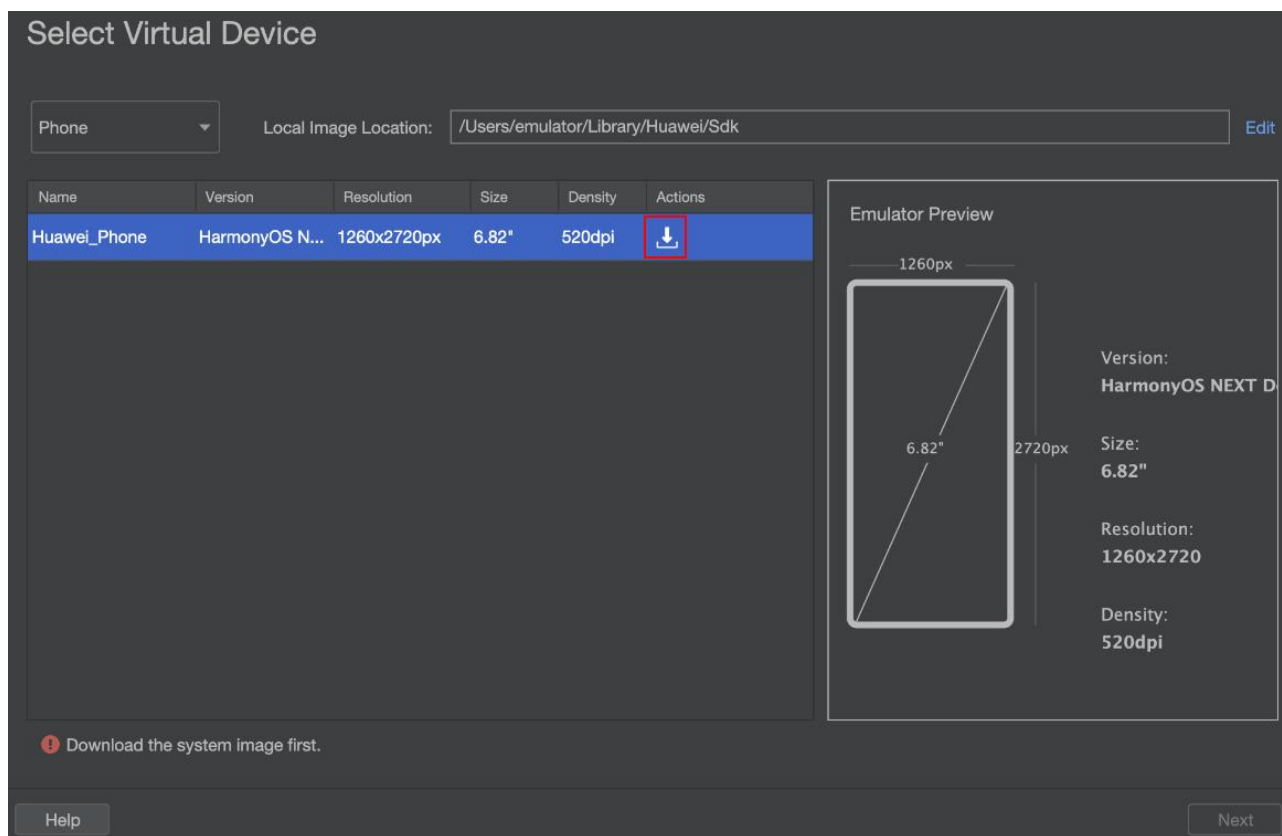
<https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/ide-emulator-create-V5>

步骤 1 点击菜单栏的 Tools > Device Manager，点击右下角的 Edit 设置模拟器实例的存储路径 Local Emulator Location，Mac 默认存储在 ~/.Huawei/Emulator/deployed 下，Windows 默认存储在 C:\Users\xxx\AppData\Local\Huawei\Emulator\deployed 下。



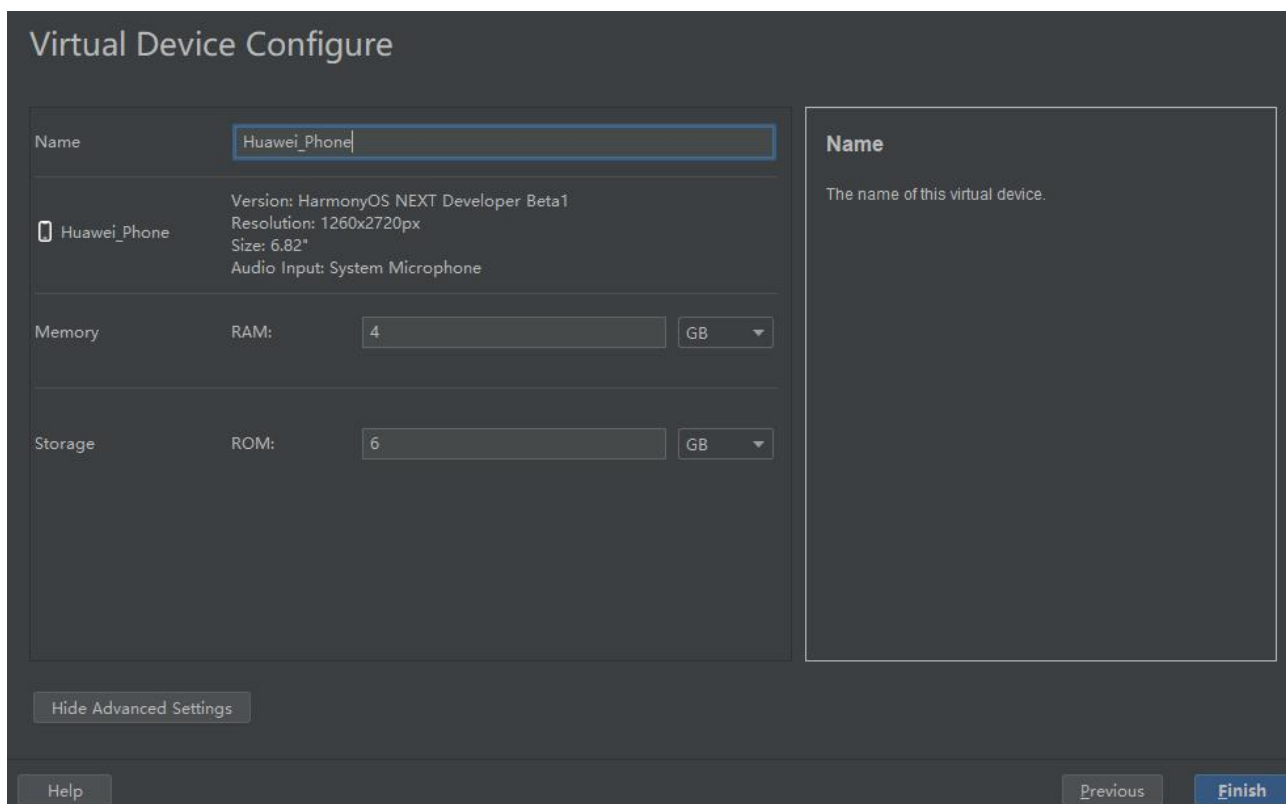
步骤 2 在 Local Emulator 页签中，单击右下角的 New Emulator 按钮，创建一个模拟器。


在模拟器配置界面，可以选择一个默认的设备模板，首次使用时会提示“Download the system image first”，请点击设备右侧的下载模拟器镜像，您也可以在该界面更新或删除不同设备的模拟器镜像。单击 Edit 可以设置镜像文件的存储路径。Mac 默认存储在 ~/.Library/Huawei/Sdk 下，Windows 默认存储在 C:\Users\xxx\AppData\Local\Huawei\Sdk 下。

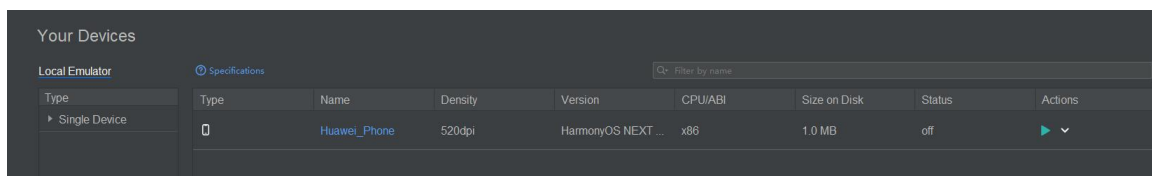


步骤 3 单击 Next，核实确定需要创建的模拟器的名称，内存和存储空间，然后单击 Finish 创建模拟器。

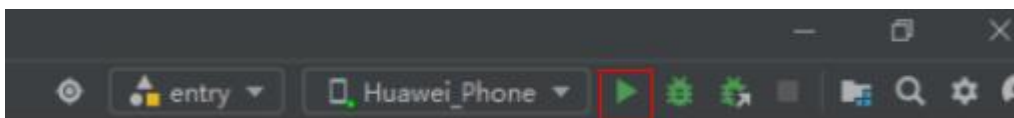




步骤 4 在设备管理器页面，单击  启动模拟器。



步骤 5 单击 DevEco Studio 的 Run > Run'模块名称'或 .



步骤 6 DevEco Studio 会启动应用/服务的编译构建与推包，完成后应用/服务即可运行在模拟器上。



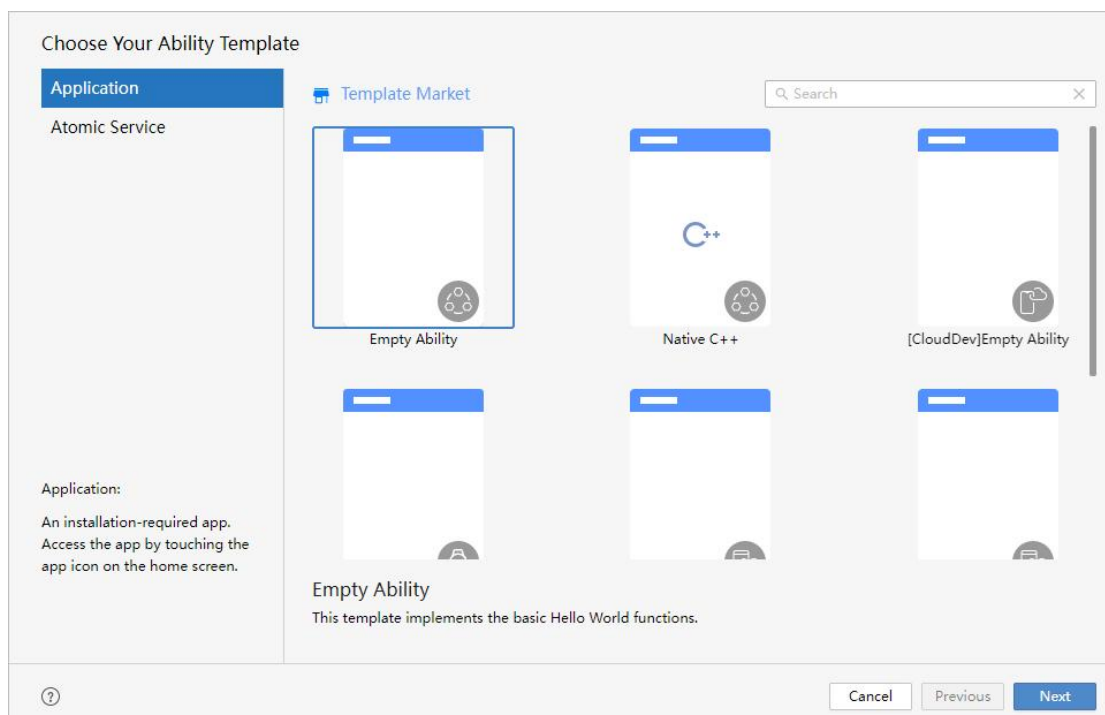
至此，完成了 DevEcoStudio 及模拟器的安装。

### 1.2.3 开发及运行环境验证

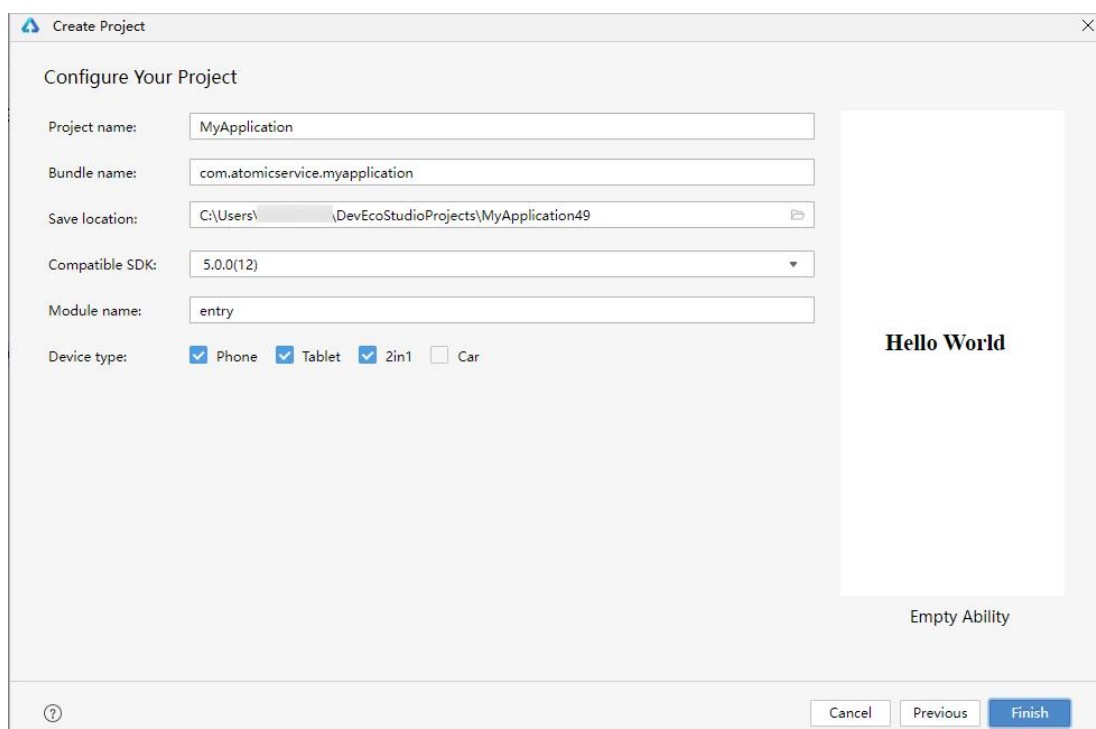
DevEco Studio 安装完成并创建模拟器后，可以通过运行 Hello World 工程来验证环境设置是否正确。接下来以创建一个支持 Phone 设备的工程为例进行介绍。

步骤 1 打开 DevEco Studio，在欢迎页单击 Create Project，创建一个新工程。

步骤 2 根据工程创建向导，选择创建 Application 或 Atomic Service。选择 Empty Ability 模板，然后单击 Next。



步骤 3 填写工程相关信息，单击 Finish。关于各个参数的详细介绍，请参考创建一个新工程。



- **Project name:** 工程的名称，可以自定义，由大小写字母、数字和下划线组成。
- **Bundle name:** 标识应用的包名，用于标识应用的唯一性。

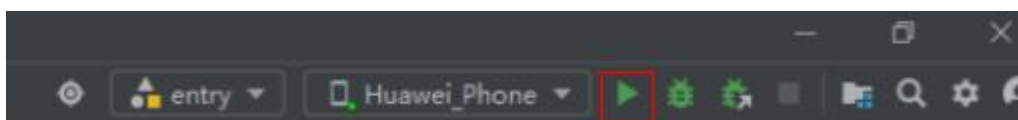
#### 说明

应用包名要求：

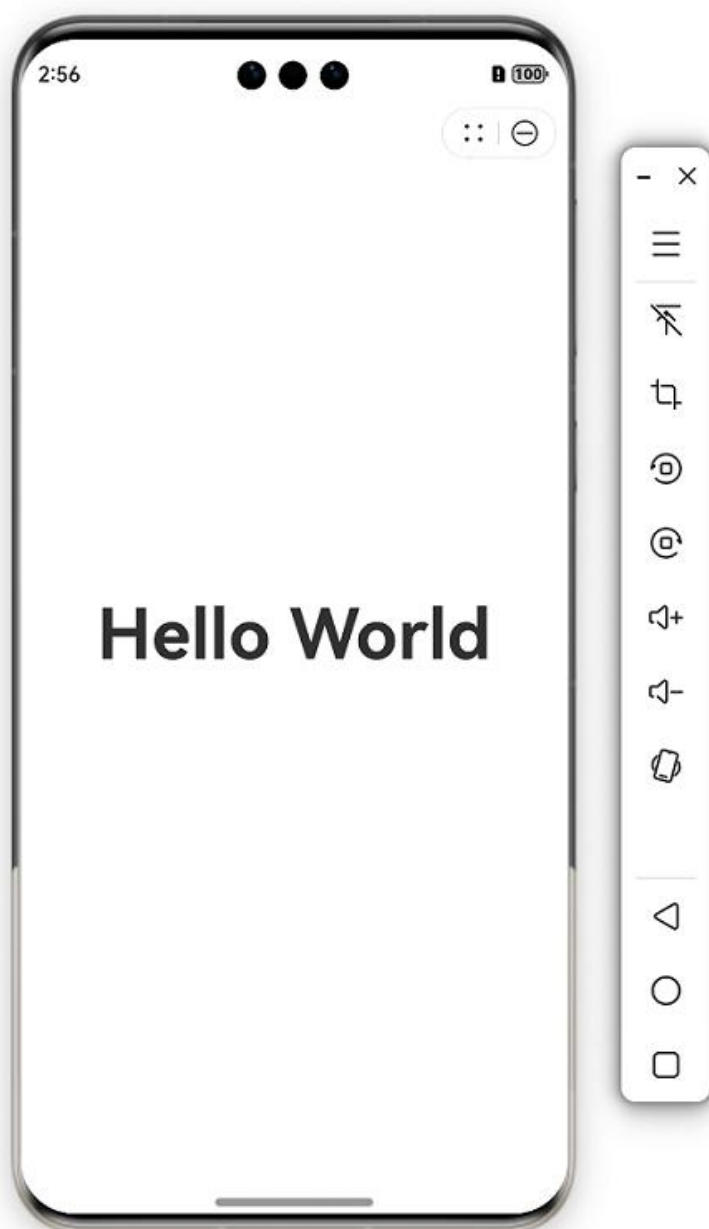
- 必须为以点号 (.) 分隔的字符串，且至少包含三段，每段中仅允许使用英文字母、数字、下划线 (\_)，如“com.example.myapplication”。
  - 首段以英文字母开头，非首段以数字或英文字母开头，每一段以数字或者英文字母结尾，如“com.01example.myapplication”。
  - 不允许多个点号 (.) 连续出现，如“com.example...myapplication”。
  - 长度为 7~128 个字符。
- **Save location:** 工程文件本地存储路径，由大小写字母、数字和下划线等组成，不能包含中文字符。
  - **Compatible SDK:** 兼容的最低 API Version。
  - **Module name:** 模块的名称。
  - **Device type:** 该工程模板支持的设备类型。

注意，工程创建完成后，DevEco Studio 会自动进行工程的同步。

步骤 4 单击 DevEco Studio 的 Run > Run'模块名称'或 。



步骤 5 DevEco Studio 会启动应用/服务的编译构建与推包，完成后应用/服务即可运行在模拟器上。



## 1.3 程序开发 - 基于 ArkUI 组件的简易计算器开发

### 1.3.1 实验介绍

使用 ArkTs 基于基础组件、容器组件，实现一个支持加减乘除混合运算的计算器实现简易计算器。本节涉及到的知识点包括：基础组件、容器组件。

---

涉及知识点:

- [ForEach 组件](#): ForEach 基于数组类型数据执行循环渲染。
- [TextInput 组件](#): 单行文本输入框组件。
- [Image 组件](#): Image 为图片组件，常用于在应用中显示图片。Image 支持加载 string、[PixelMap](#) 和 [Resource](#) 类型的数据源，支持 png、jpg、bmp、svg 和 gif 类型的图片格式。

## 1.3.2 代码开发

软件要求

DevEco Studio 版本: DevEco Studio 5.0.0 Release 及以上

HarmonyOS SDK 版本: HarmonyOS 5.0.0 Release 及以上

硬件要求

设备类型: 华为手机, 模拟器。

HarmonyOS 系统: HarmonyOS 5.0.0 Release 及以上

本实验采用华为提供的代码，实际上课过程中可作为编程题目。建议学生自行完成后参考此代码:

```
// 填入 ArkTs 相关示例代码

//CalculateUtil.ets
parseExpression(expressions: Array<string>): string {
    if (CheckEmptyUtil.isEmpty(expressions)) {
        return 'NaN';
    }
    let len = expressions.length;
    let outputStack: string[] = [];
    let outputQueue: string[] = [];
    expressions.forEach((item: string, index: number) => {
        // Handle % in the expression
        if (item.indexOf(CommonConstants.PERCENT_SIGN) !== -1) {
            expressions[index] = (this.mulOrDiv(item.slice(0, item.length - 1),
                CommonConstants.ONE_HUNDRED, CommonConstants.DIV)).toString();
        }
    });
}
```

```

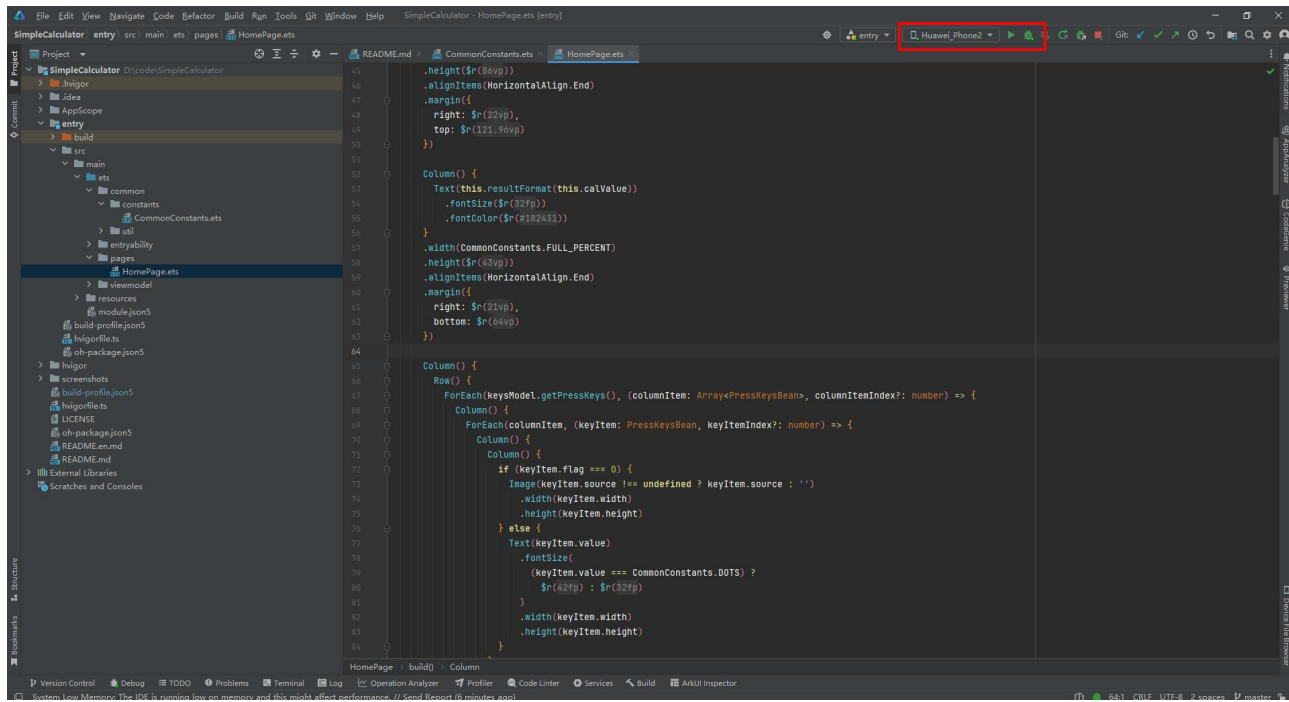
    }
    // Whether the last digit is an operator.
    if ((index === len - 1) && this.isSymbol(item)) {
        expressions.pop();
    }
});
while (expressions.length > 0) {
    let current: string | undefined = expressions.shift();
    if (current !== undefined) {
        if (this.isSymbol(current)) {
            // Processing addition, subtraction, multiplication and division.
            while (outputStack.length > 0 && this.comparePriority(current,
outputStack[outputStack.length - 1])) {
                let popValue: string | undefined = outputStack.pop();
                if (popValue !== undefined) {
                    outputQueue.push(popValue);
                }
            }
            outputStack.push(current);
        } else {
            // Processing the numbers.
            outputQueue.push(current);
        }
    }
}
while (outputStack.length > 0) {
    let popValue: string | undefined = outputStack.pop();
    if (popValue !== undefined) {
        outputQueue.push(popValue);
    }
}
return this.dealQueue(outputQueue);
}

```

// 其它涉及的 ets 及配置文件

### 1.3.3 代码编译运行

#### 步骤 1 代码编译并推送到模拟器安装并运行

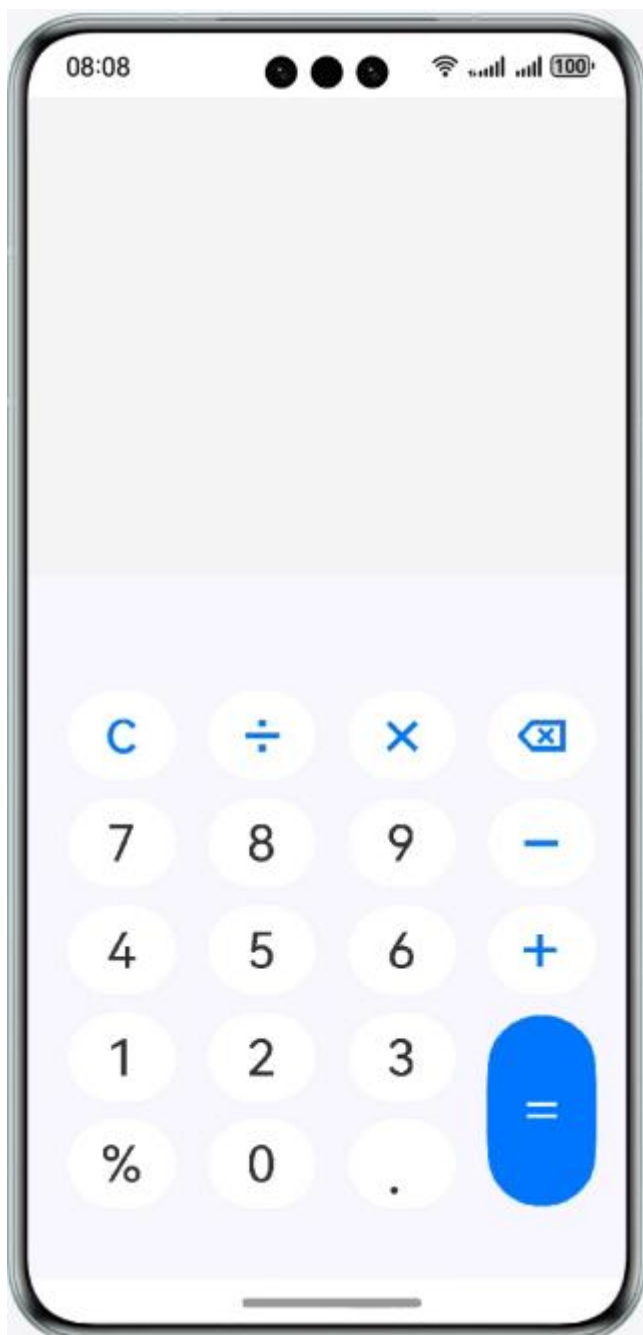


自此实验成功启动。

### 1.3.4 代码验证调测

#### 步骤 1 打开界面后，测试程序运行结果。





代码请下载：

[https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials\\_NEXT-SimpleCalculator](https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials_NEXT-SimpleCalculator)

### 1.3.5 实验总结与课后思考

思考 1：算法是数学吗？和计算机之间的联系是什么？

---

思考 2：你认为一个好的程序应该具备什么的素质？

思考 3：语言之间的共性是什么？为什么会有不同的编程语言？

思考 4：ArkTs 语言的特点是什么？

## 1.4 程序开发 - 基于 NDK 的排序程序开发

### 1.4.1 实验介绍

使用 DevEco Studio 创建一个 Native C++应用，实现计算对给定 3 个整数的排序。

涉及知识点：

- **NDK 开发导读**：NDK（Native Development Kit）是 HarmonyOS SDK 提供的 Native API、相应编译脚本和编译工具链的集合，方便开发者使用 C 或 C++语言实现应用的关键功能。
- **标准库**：目前支持标准 C 库、C++库、OpenSL ES、zlib 等。

### 1.4.2 代码开发

本实验采用华为提供的代码，实际上课过程中可作为编程题目。建议学生自行完成后参考此代码：

```
// 填入 ArkTs 相关示例代码

// index.ets 关键代码
let resultTemp = libHello.mySort(this.numX, this.numY, this.numZ);
this.result = resultTemp;

// hello.cpp

void MyQuickSort(int s[], int left, int right)
{
    if (left < right) {
        int i = left, j = right, x = s[left];
        while (i < j)
        {
```

```
while(i < j && s[j]>= x) // 从右向左找第一个小于 x 的数
    j--;
if(i < j)
    s[i++] = s[j];
while(i < j && s[i]< x) // 从左向右找第一个大于等于 x 的数
    i++;
if(i < j)
    s[j--] = s[i];
}
s[i] = x;
MyQuickSort(s, left, i - 1); // 递归调用
MyQuickSort(s, i + 1, right);
}
}
```

// 其它涉及的 ets 及配置文件  
不涉及，参考代码压缩包

### 1.4.3 代码验证调测

步骤 1 界面效果如图所示

## 调用C程序进行排序

对以下数字进行排序

排序

结果  
0

输入X数值

输入Y数值

输入Z数值

步骤 2 填入 X, Y, Z 的数值。

调用C程序进行排序

对以下数字进行排序

排序

结果  
0

输入X数值 1

输入Y数值 3

输入Z数值 2

步骤 3 提交得到排序结果。

代码请下载：

[https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials\\_NEXT-NativeTemplateDemo](https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials_NEXT-NativeTemplateDemo)

### 1.4.4 实验总结与课后思考

思考 1: Native 是否可以调用 Ts?

思考 2: Ts 和 Native 调用是否有开销?

## 1.5 程序开发 - 基于 ArkData 用户首选项的数据持久化程序开发

### 1.5.1 实验介绍

基于 ArkData 使用@ohos.data.preferences 接口，展示了使用首选项持久化存储数据的功能。本节涉及到的知识点 ArkData 的关键 API 使用。学完本节后你应该学会 ArkData 中首选项的使用。

涉及知识点:

- **用户首选项**: 为应用提供 Key-Value 键值型的数据处理能力，支持应用持久化轻量级数据，并对其修改和查询。数据存储形式为键值对，键的类型为字符串型，值的存储数据类型包括数字型、字符型、布尔型以及这 3 种类型的数组类型。
- **TextInput**: 单行文本输入框组件。
- **Button**: 按钮组件，可快速创建不同样式的按钮。

### 1.5.2 代码开发

约束与限制

---

1. 本示例仅支持标准系统上运行，支持设备：华为手机，模拟器（备注：分布式能力不支持）。
2. DevEco Studio 版本：DevEco Studio 5.0.0 Release 及以上
3. HarmonyOS SDK 版本：HarmonyOS 5.0.0 Release 及以上
4. HarmonyOS 系统：HarmonyOS 5.0.0 Release 及以上

本实验采用华为提供的代码，实际上课过程中可作为编程题目。建议学生自行完成后参考此代码：

```
// 填入 ArkTs 相关示例代码

// 切换主题: 在首页预先设置好几套主体数据, 使用 preferences.getPreferences 获取使用 Preferences 对象,
// 调用 Preferences.get() 读取缓存中的参数, 得到当前应该展示哪一套主体。每次点击切换按钮都会调用
// Preferences.put()来重新修改参数, 然后使用 Preferences.flush()保存并刷新文件内容。源码参考:Index.ets 。
// index.ets 关键代码

async aboutToAppear() {
    //从内存中获取轻量级存储 db 文件
    await this.getPreferencesFromStorage()
    //从轻量级存储 db 文件中获取键名为 theme 的键值
    this.nowTheme = await this.getPreference()
    console.info(`nowTheme__get ${this.nowTheme}`)
    emitter.emit({ eventId: 0, priority: 0 }, {
        data: {
            nowTheme: this.nowTheme
        }
    })
    let index = THEME_NAMES.indexOf(this.nowTheme)
    this.themeDatas = THEMES[index]
}

async getPreferencesFromStorage() {
    let context = getContext(this) as Context
    preferenceTheme = await preferences.getPreferences(context, PREFERENCES_NAME)
}

async putPreference(data: string) {
    Logger.info(TAG, 'Put begin')
    if (preferenceTheme !== null) {
```

```
        await preferenceTheme.put('theme', data)
        await preferenceTheme.flush()
    }
}

async getPreference(): Promise<string> {
    Logger.info(TAG, `Get begin`)
    let theme: string = ""
    if (preferenceTheme !== null) {
        theme = await preferenceTheme.get('theme', 'default') as string;
        return theme;
    }
    return theme;
}

// 其它涉及的 ets 及配置文件
不涉及，参考代码压缩包
```

### 1.5.3 代码验证调测

步骤 1 启动应用，点击顶部 titleBar 的右侧切换按钮，弹出主题菜单，选择任意主题则切换相应的主题界面。









步骤 2 退出应用再重新进入，显示上一次退出前的主题界面

代码请下载

[https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials\\_NEXT-P-references](https://developer.huawei.com/consumer/cn/codelabsPortal/carddetails/tutorials_NEXT-P-references)

## 1.5.4 实验总结与课后思考

思考 1: 对哪些应用来说, ArkData 是必不可少, 哪些不是?

思考 2: 应用升级时场景兼容性除了数据, 还有哪些要思考

思考 3: 考虑分布式对象的使用场景

# 1.6 程序开发 - 基于 Basic Services Kit 的应用账号管理程序开发

## 1.6.1 实验介绍

本示例基于 Basic Services Kit 作为基础服务套件, 为应用开发者提供常用的基础能力。学完本节后, 基本可以掌握应用账号管理相关功能, 包括对应用账号的添加、删除、查询、修改和授权, 以及提供账号将数据写入磁盘和数据同步的能力。

## 1.6.2 代码开发

本示例分为音乐, 视频, 地图三个模块:

- 音乐模块

使用 Navigation, Button, Text, TextInput 组件开发注册, 登录, 修改信息和切换应用页面, createAppAccountManager 方法创建应用账号管理器对象

源码链接: AccountData.ets, AccountModel.ets

接口参考: @ohos.account.appAccount, @ohos.data.preferences, @ohos.router

- 视频模块

使用 Navigation, Button, Text, TextInput 组件开发注册, 登录, 修改信息和切换应用页面, createAppAccountManager 方法创建应用账号管理器对象

源码链接: AccountData.ets, AccountModel.ets

接口参考: @ohos.account.appAccount, @ohos.data.preferences, @ohos.router

- 地图模块

---

使用 Navigation,Button, Text,TextInput 组件开发注册, 登录, 修改信息和切换应用页面, createAppAccountManager 方法创建应用账号管理器对象

源码链接: AccountData.ets, AccountModel.ets

接口参考: @ohos.account.appAccount, @ohos.data.preferences, @ohos.router

本示例仅支持标准系统上运行, 支持设备: 华为手机, 模拟器 (备注: Basic Services Kit 中 usb、热管理、设备认证不支持模拟器)

1. DevEco Studio 版本: DevEco Studio 5.0.0 Release 及以上
2. HarmonyOS SDK 版本: HarmonyOS 5.0.0 Release 及以上
3. HarmonyOS 系统: HarmonyOS 5.0.0 Release 及以上

```
// 填入 ArkTs 相关示例代码
```

```
// AccountData.ets 关键代码
```

```
async getFromStorage(context: common.Context, url: string) {
    let name = url;
    Logger.info(TAG, `Name is ${name}`);
    try {
        storage = await preferences.getPreferences(context, `${name}`);
    } catch (err) {
        Logger.error(`getStorage failed, code is ${err?.code}, message is ${err?.message}`);
    }
    if (storage) {
        Logger.info(TAG, `Create stroage is fail.`);
    }
}

async getStorage(context: common.Context, url: string) {
    storage = storageTemp;
    await this.getFromStorage(context, url);
    return storage;
}
```

```
async putStorageValue(context: common.Context, key: string, value: string, url: string) {
    storage = await this.getStorage(context, url);
    try {
        await storage.put(key, value);
        await storage.flush();
        Logger.info(TAG, `put key && value success`);
    } catch (err) {
        Logger.info(TAG, `aaaaaa put failed`);
    }
    return
}

async hasStorageValue(context: common.Context, key: string, url: string) {
    storage = await this.getStorage(context, url);
    let result: boolean = false;
    try {
        result = await storage.has(key);
    } catch (err) {
        Logger.error(`hasStorageValue failed, code is ${err?.code}, message is ${err?.message}`);
    }
    Logger.info(TAG, `hasStorageValue success result is ${result}`);
    return result;
}

async getStorageValue(context: common.Context, key: string, url: string) {
    storage = await this.getStorage(context, url);
    let getValue: preferences.ValueType = 'null';
    try {
        getValue = await storage.get(key, 'null');
    } catch (err) {
        Logger.error(`getStorageValue failed, code is ${err?.code}, message is ${err?.message}`);
    }
    Logger.info(TAG, `getStorageValue success`);
    return getValue;
}

async deleteStorageValue(context: common.Context, key: string, url: string) {
    storage = await this.getStorage(context, url);
```

---

```
    try {
        await storage.delete(key);
        await storage.flush();
    } catch (err) {
        Logger.error(`deleteStorageValue failed, code is ${err?.code}, message is ${err?.message}`);
    }
    Logger.info(TAG, `delete success`);
    return
}
```

// AccountModel.ets 关键代码

```
async addAccount(username: string) {
    await app.createAccount(username);
    Logger.info(TAG, `addAccount success`);
    return;
}
```

```
async deleteAccount(username: string) {
    await app.removeAccount(username);
    Logger.info(TAG, `deleteAccount success`);
    return;
}
```

```
async setAccountCredential(username: string, credentialType: string, credential: string) {
    await app.setCredential(username, credentialType, credential);
    Logger.info(TAG, `setAccountCredential success`);
    return;
}
```

```
async setAssociatedData(name: string, key: string, value: string) {
    await app.setCustomData(name, key, value);
    Logger.info(TAG, `setAssociatedData success`);
    return;
}
```

```
async getAccountCredential(name: string, credentialType: string) {
```

```
    let result = await app.getCredential(name, credentialType);
    Logger.info(TAG, `getAccountCredential success`);
    return result;
  }

  async getAssociatedData(name: string, key: string) {
    let result = await app.getCustomData(name, key);
    Logger.info(TAG, `getAssociatedData success`);
    return result;
  }

  // 其它涉及的 ets 及配置文件
  不涉及，参考代码压缩包
```

### 1.6.3 代码验证调测

步骤 1 首页选择想要进入的应用，首次进入该应用需要进行注册，如已注册账号则直接登录；





步骤 2 注册页面可设置账号名、邮箱、个性签名、密码（带\*号为必填信息），注册完成后返回登录页面使用注册的账号进行登录；

The image displays two side-by-side mobile app screens. The left screen is the registration page, titled '注册' (Register) in the top right corner. It features a blue header bar with a back arrow and the text '返回' (Return). The form includes fields for '应用名' (App Name) set to 'Music', '\*用户名' (Username) with a placeholder 'xxxxxx' and a note '输入数字、字母或下划线, 15位' (Enter numbers, letters, or underscores, 15 characters), '邮箱' (Email) with a placeholder 'xxxxxx' and a note '输入邮箱格式, 18位' (Enter email format, 18 characters), '个性签名' (Signature) with a placeholder 'xxxxxx' and a note '输入签名, 18位' (Enter signature, 18 characters), '\*密码' (Password) with a placeholder 'xxxxxx' and a note '输入密码, 6-18位' (Enter password, 6-18 characters), and '\*确认密码' (Confirm Password) with a placeholder 'xxxxxx' and a note '再次输入密码, 6-18位' (Re-enter password, 6-18 characters). A blue button at the bottom is labeled '设置完成' (Settings Complete). The right screen is the login page, titled '注册/登录' (Register/Login) in the top right corner. It has the same blue header bar. The form includes '应用名' (App Name) set to 'Music', '\*用户名' (Username) with the value 'admin', and '\*密码' (Password) with a masked input '.....' and an eye icon. Below the password field are two blue buttons: '注册' (Register) and '登录' (Login).

步骤 3 登录后进入账号详情界面，点击**修改信息**按钮可跳转至账号信息修改页面重新设置账号信息；



步骤 4 点击**切换应用**按钮则退出该账号并返回主页面。重新选择想要进入的应用；

步骤 5 点击**删除账号**按钮则会删除该账号所有相关信息；

代码请下载：

[https://gitee.com/harmonyos\\_samples/app-account-manager/blob/master/README.md](https://gitee.com/harmonyos_samples/app-account-manager/blob/master/README.md)

## 1.6.4 实验总结与课后思考

思考 1: 账号被删除后能否恢复? 需要采用哪些措施保证数据不被恢复

# 1.7 程序开发 - 基于系统相关 Kit 的用户身份认证程序开发

## 1.7.1 实验介绍

User Authentication 提供了基于用户在设备本地注册的锁屏口令、人脸和指纹来认证用户身份的能力。学完本节后, 基本可以掌握用户身份认证的各种鉴权场景, 如应用内账号登录、支付认证等。

## 1.7.2 代码开发

- 该 sample 应用在调用接口时需要:

1.允许应用将窗口设置为隐私窗口, 禁止截屏录屏的权限,  
"ohos.permission.PRIVACY\_WINDOW";

2.允许应用使用生物特征识别能力进行身份认证的权限,  
"ohos.permission.ACCESS\_BIOMETRIC";

已在 module.json5 文件中添加。

- 本示例仅支持标准系统上运行, 支持设备: 华为手机, 模拟器 (备注: 仅支持口令认证)。

DevEco Studio 版本: DevEco Studio 5.0.0 Release 及以上

HarmonyOS SDK 版本: HarmonyOS 5.0.0 Release 及以上

HarmonyOS 系统: HarmonyOS 5.0.0 Release 及以上

```
// 填入 ArkTs 相关示例代码
// UserAuthModel.ets
private userAuth?: userAuth.UserAuthInstance;

/**
 * Check whether the capability is supported.
 *
 * @param type UserAuthType.
 */
```

```

queryAvailable(type: userAuth.UserAuthType): boolean {
    let isAvailable: boolean = false;
    if (!type) {
        return false;
    }
    try {
        userAuth.getAvailableStatus(type, userAuth.AuthTrustLevel.ATL1);
        isAvailable = true;
    } catch (error) {
        Logger.error(CommonConstants.TAG, `auth trust level is not supported cause ${error.code}
${error.message}`);
        isAvailable = false;
    }
    return isAvailable;
}

/**
 * Queries whether facial recognition is supported.
 */
isFaceAvailable(): boolean {
    return this.queryAvailable(userAuth.UserAuthType.FACE)
}

/**
 * Querying whether fingerprint recognition is support.
 */
isFingerprintAvailable(): boolean {
    return this.queryAvailable(userAuth.UserAuthType.FINGERPRINT);
}

/**
 * Obtains a facial recognition object.
 */
getFaceAuth(callback: (isSuccess: boolean) => void) {
    if (!callback) {
        return;
    }
    let authType = userAuth.UserAuthType.FACE;

```

---

```

        let authTrustLevel = userAuth.AuthTrustLevel.ATL1;
        this.getAuth(authType, authTrustLevel, callback);
    }

    /**
     * Obtaining a fingerprint recognition object.
     */
    getFingerprintAuth(callback: (isSuccess: boolean) => void) {
        if (!callback) {
            return;
        }
        let authType = userAuth.UserAuthType.FINGERPRINT;
        let authTrustLevel = userAuth.AuthTrustLevel.ATL1;
        this.getAuth(authType, authTrustLevel, callback);
    }

    /**
     * Obtains the user authentication object.
     *
     * @param authType UserAuthType
     * @param authTrustLevel AuthTrustLevel
     */
    getAuth(authType: userAuth.UserAuthType, authTrustLevel: userAuth.AuthTrustLevel, callback:
(isSuccess: boolean) => void) {
        // 设置认证参数
        const authParam: userAuth.AuthParam = {
            challenge: new Uint8Array([49, 49, 49, 49, 49, 49]),
            authType: [authType],
            authTrustLevel: authTrustLevel,
        }
        // 配置认证界面
        const widgetParam: userAuth.WidgetParam = {
            title: '请进行身份认证',
        };

        try {
            this.userAuth = userAuth.getUserAuthInstance(authParam, widgetParam);
            this.userAuth.on('result', {

```

```

        onResult(result) {
            console.info(`userAuthInstance callback result: ${JSON.stringify(result)}`);
            if (result.result === userAuth.UserAuthResultCode.SUCCESS) {
                callback(true);
            }
        }
    });
} catch (error) {
    Logger.error(CommonConstants.TAG, `get auth instance failed cause ${error.code}
${error.message}`);
}
}

/**
 * Convert milliseconds to seconds.
 *
 * @param milliseconds Milliseconds.
 */
convertToSeconds(milliseconds: number): number {
    return Math.ceil(milliseconds / CommonConstants.MILLISECONDS_TO_SECONDS);
}

/**
 * Start authentication.
 */
start() {
    if (!this.userAuth) {
        Logger.error(CommonConstants.TAG, `userAuth is undefined`);
        return;
    }
    try {
        this.userAuth.start();
    } catch (error) {
        Logger.error(CommonConstants.TAG, `authV9 start auth failed, error ${error.code}
${error.message}`);
    }
}
}

```

```
/**
 * Cancel authentication
 */
cancel() {
  if (!this.userAuth) {
    Logger.error(CommonConstants.TAG, `userAuth is undefined`);
    return;
  }
  try {
    this.userAuth.cancel();
  } catch (error) {
    Logger.error(CommonConstants.TAG, `cancel auth failed ${error.code} ${error.message}`);
  }
}
// 其它涉及的 ets 及配置文件参考代码压缩包
```

### 1.7.3 代码验证调测

步骤 1 初次登录界面没有人脸识别和指纹识别登录;





步骤 2 注册时选择人脸识别和指纹识别按钮，再次返回登录界面就会提供人脸识别和指纹识别登录(模拟器不支持);

步骤 3 登录界面防截屏功能(模拟器不支持);


代码请下载: [https://gitee.com/harmonyos\\_samples/UserAuth/blob/master/README.md](https://gitee.com/harmonyos_samples/UserAuth/blob/master/README.md)

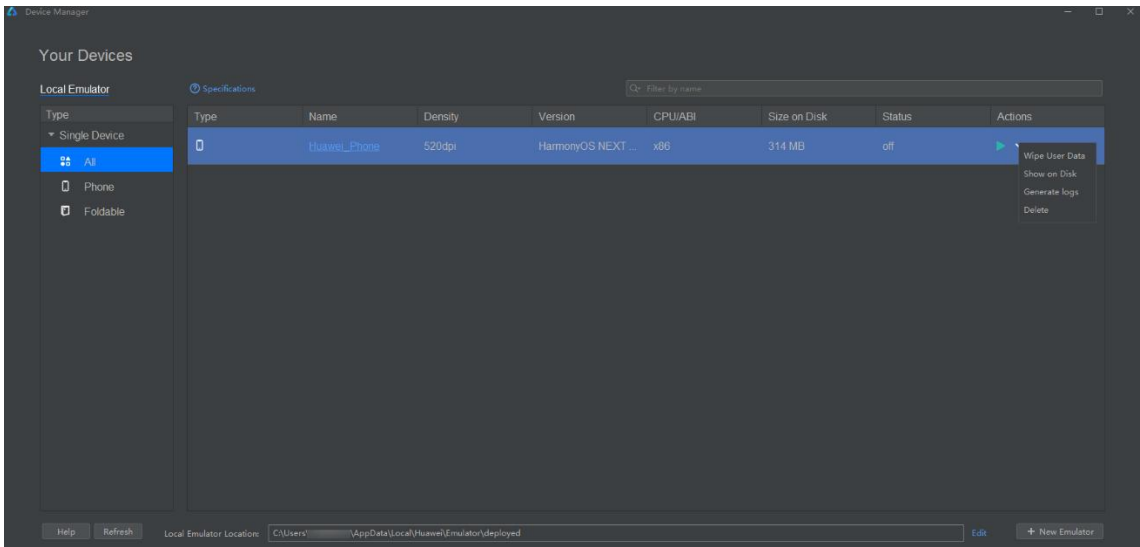
### 1.7.4 实验总结与课后思考

思考 1：除上面以外，还有哪些认证方案，都有哪些优缺点？




## 1.8 实验环境清理

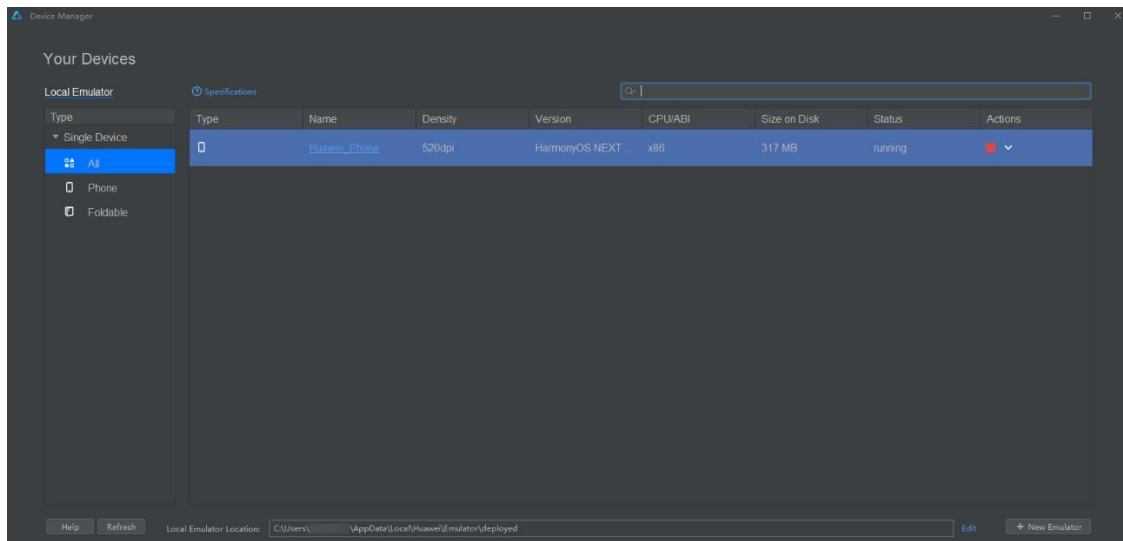
### 步骤 1 清除模拟器中用户数据

在设备管理器页面，单击即可启动模拟器。模拟器启动时会默认携带上一次运行时的用户数据，包括用户上传的文件，安装的应用等。如果是新创建的模拟器，则不会携带用户数据。如果想清除上一次运行时的用户数据，点击 Actions >  > Wipe User Data。



### 步骤 2 关闭模拟器

想要关闭运行时的模拟器，可以在设备管理器页面点击 ，或者点击模拟器工具栏上的关闭按钮 。模拟器关闭后，点击 Actions >  > Delete 可以删除模拟器，并清除模拟器的用户数据和配置信息。



步骤 3 关闭 DevEco Studio 当前项目并退出。