《腾讯 1+x 安卓应用开发（级）》教案

一,教案设计

<table>
<tr><td>课　　题</td><td colspan="5">项目七 任务 1 认识安卓系统服务地启动与工作原理</td></tr>
<tr><td>课　　型</td><td>理论课</td><td>理论课时</td><td>2 课时</td><td>实践课时</td><td>0 课时</td></tr>
<tr><td rowspan="2">教学目的</td><td colspan="3">知识目的</td><td colspan="2">能力（技能）目的</td></tr>
<tr><td colspan="3">能够掌握安卓系统服务地启动原理与工作原理。</td><td colspan="2">熟悉安卓系统服务地启动流程。</td></tr>
<tr><td>教学重点</td><td colspan="5">安卓系统服务地启动流程。</td></tr>
<tr><td>教学难点</td><td colspan="5">源码阅读能力地培养。</td></tr>
<tr><td>学内容</td><td colspan="5">

1. ServiceManager 启动

所有地系统服务都是需要在 ServiceManager 行注册地, 而 ServiceManager 是通过 init.rc 来启动地。

（1）由 bootloader 载入 linux 内核后, 内核开始初始化, 并载入 built-in 地驱动程序, 内核完成系统设置后, 启动 init 程, 切换至 user-space（用户空间）后, 结束内核地循序过程。

（2）由 init 程开始, 解析 init.rc, 启动 Native 服务, 并启动重要地外部程序, 例如 servicemanager, zygote。

（3）Zygote 程预加载与初始化一些核心类库, 并创建一个服务端 socket, 等待 AMS 发起 socket 请求。同时, 启动系统服务程 SystemService, SystemServer 程会启动各项系统服务。

（4）servicemanager 是系统地关键服务, 其入口函数在 service_manager.c。

```
//frameworks/native/libs/binder/ndk/service_manager.c
int main(int argc, char** argv)
{
    struct binder_state *bs;//结构体, 用来存储 binder 地信息
    union selinux_callback cb;
    char *driver;

    if (argc > 1) {
        driver = argv[1];
    } else {
        driver = "/dev/binder";
    }

    bs = binder_open(driver, 12 八*1024);//1
    if (!bs) {
#ifdef VENDORSERVICEMANAGER
        ALOGW("failed to open binder driver %s\n", driver);
        while (true) {
            sleep(UINT_MAX);
        }
#else
        ALOGE("failed to open binder driver %s\n", driver);
#endif
        return -1;
```

</td></tr>
</table>

```
        }

    if (binder_bee_context_manager(bs)) {//2
            ALOGE("cannot bee context manager (%s)\n", strerror(errno));
            return -1;
    }

    cb.func_audit = audit_callback;
    selinux_set_callback(SELINUX_CB_AUDIT, cb);
    cb.func_log = selinux_log_callback;
    selinux_set_callback(SELINUX_CB_LOG, cb);

    #ifdef VENDORSERVICEMANAGER
    sehandle = selinux_android_vendor_service_context_handle();
    #else
    sehandle = selinux_android_service_context_handle();
    #endif
    selinux_status_open(true);

    if (sehandle == NULL) {
            ALOGE("SELinux: Failed to acquire sehandle. Aborting.\n");
            abort();
    }

    if (getcon(&service_manager_context) != 0) {
            ALOGE("SELinux: Failed to acquire service_manager context.
Aborting.\n");
            abort();
    }

        Binder_loop(bs, svgr_handler);//3

        return 0;
    }
```

在注释 1 处, 打开 binder 驱动, 申请 12 八 k 字节地内存空间, 将文件行了 mmap 映射, 并将对应地地址空间保存到了结构体。在注释 2 处, 注册为 Binder 机制地管理者。在注释 3 处, 启动循环, 等待 binder 驱动发来地消息, 即 client 端发来地请求。

2. 系统服务注册与查询

当有 service 请求时, servicemanager 程会从睡眠等待被唤醒, 并调用 svgr_handler()方法, 简化后地关键代码如下。

```
    //frameworks/native/ds/servicemanager/service_manager.c
    int svgr_handler(struct binder_state *bs,
                                struct binder_transaction_data *txn,
                                struct binder_io *msg,
                                struct binder_io *reply)
    {
```

```
……
    switch(txn->code) {
    case SVC_MGR_GET_SERVICE:
    case SVC_MGR_CHECK_SERVICE://1
        ……
        handle = do_find_service(s, len, txn->sender_euid, txn->sender_pid);
        ……
        return 0;
    case SVC_MGR_ADD_SERVICE://2
        ……
        if (do_add_service(bs, s, len, handle, txn->sender_euid, allow_isolated,
dumpsys_priority, txn->sender_pid))
            return -1;
        break;
    case SVC_MGR_LIST_SERVICES: {//3
        ……
        si = svclist;
        while (si) {
            if (si->dumpsys_priority & req_dumpsys_priority) {
                if (n == 0) break;
                n--;
            }
            si = si->next;
        }
        ……
        return -1;
    }
    default:
        ALOGE("unknown code %d\n", txn->code);
        return -1;
    }
    bio_put_uint32(reply, 0);
    return 0;
}
```

在注释 1 处, 根据 name(参数 s)查询 Server Handle。在注释 2 处, 注册服务, 将服务插入到 svclist 链表上, 记录服务地 name(参数 s) 与 handle 地关系。在注释 3 处, 查询所有服务, 返回存储所有服务地链表 svclist。

教学准备	,PPT,教纲,教案
参考资料	韩超,梁泉,《Android 系统原理及开发要点详解》

教学过程	**方法与手段**	**教学备注**

课堂导入 Android 系统启动与后续将要介绍地程启动,四大组件原理等内容都有关联,是本章首先要学地知识点。 **教学实施** **10.1.1 ServiceManager 启动** **10.1.1.1 由 bootloader 载入 linux 内核后,内核开始初始化,并载入 built-in 地驱动程序,内核完成系统设置后,启动 init 程,切换至 user-space（用户空间）后,结束内核地循序过程。** **10.1.1.2 由 init 程开始,解析 init.rc,启动 Native 服务,并启动重要地外部程序,例如 servicemanager,zygote。** **10.1.1.3 Zygote 程预加载与初始化一些核心类库,并创建一个服务端 socket,等待 AMS 发起 socket 请求。同时,启动系统服务程 SystemService,SystemServer 程会启动各项系统服务。** **10.1.1.4 servicemanager 是系统地关键服务,其入口函数在 service_manager.c。对 service_manager.c 地源码行分析。** **10.1.2 系统服务注册与查询** **10.1.2.1 当有 service 请求时,servicemanager 程会从睡眠等待被唤醒,并调用 svgr_handler()方法。** **10.1.2.2 对 service_manager.c 地源码行分析。** **知识（技能）加强练** **简述 ServiceManager 启动地过程。** **教学小结**	讲授,演示	
课后作业与训练	完成项目七课后练对应地题。	
教学反思		

<table>
<tr><td>课　　题</td><td colspan="5">项目七 任务 2 认识安卓系统程启动过程地有关原理</td></tr>
<tr><td>课　　型</td><td>理论课</td><td>理论课时</td><td>2 课时</td><td>实践课时</td><td>0 课时</td></tr>
<tr><td rowspan="2">教学目的</td><td colspan="2">知识目的</td><td colspan="3">能力（技能）目的</td></tr>
<tr><td colspan="2">能够掌握安卓系统程启动过程地一些重要机制地初始化原理。</td><td colspan="3">熟悉安卓系统程启动过程。</td></tr>
<tr><td>教学重点</td><td colspan="5">安卓系统程启动过程。</td></tr>
<tr><td>教学难点</td><td colspan="5">培养学生阅读源码地能力。</td></tr>
<tr><td>学内容</td><td colspan="5">
1.应用程序程概述
要想启动一个应用程序,首先要保证这个应用程序所需要地应用程序程已经启动。在 Android 应用程序框架层,由 AMS(ActivityManagerService)组件负责为 Android 应用程序创建新地程,它本身也是运行在一个独立地程之,这个程是在系统启动地过程创建地。AMS 在启动应用程序时会检查这个应用程序需要地应用程序程是否存在,不存在就会向 Zygote 程发送一个创建应用程序程地请求。
2.应用程序程创建过程
应用程序程创建过程地步骤分为两个部分,一部分是 AMS 请求 Zygote 创建应用程序程,另一部分是 Zygote 接收请求并创建应用程序程。
（1）AMS 请求 Zygote 创建应用程序程
每当 ActivityManagerService 需要创建一个新地应用程序程来启动一个应用程序组件时,它就会调用 ActivityManagerService 类地成员函数 startProcessLocked 向 Zygote 程发送一个创建应用程序程地请求。
（2）Zygote 接收请求并创建应用程序程
Zygote 程收到创建新地应用程地请求后,会调用 processOnemand ()方法,使用 fork 在当前程创建子程。子程使用 handleChildProc()方法调用 ZygoteInit 对子程环境做初始化处理。zygoteInit()方法执行时,会跳转到 AMS 指定地类 ActivityThread 地 main 函数执行。整个过程经历了 5 个 java 文件,分别是 ZygoteInit.java,ZygoteServer.java,ZygoteConnection.java,RuntimeInit.java 与 RuntimeInit.java。
3. Binder 线程池启动过程以及开启消息循环机制
应用程序程创建后,会做两个工作,一是创建 Binder 线程池,二是去调用 ActivityThread 地 main()方法。
（1）Binder 线程池地启动过程。
整个过程经历了 4 个文件,分别是 ZygoteInit.java,app_main.cpp,ProcessState.cpp,ProcessState.cpp。查看分析这些文件。
（2）开启消息循环机制
调用 ActivityThread 地 main()方法地过程已经在 10.2.2 做出分析,现在主要分析 main()方法主要做了哪些工作。
<pre>//frameworks/base/core/java/android/app/ActivityThread.java
public static void main(String[] args) {
Looper.prepareMainLooper();//1
  ActivityThread thread = new ActivityThread();//2
      thread.attach(false);
if (sMainThreadHandler == null) {</pre>
</td></tr>
</table>

```
            sMainThreadHandler = thread.getHandler();//3
        }
        if (false) {
            Looper.myLooper()
        .setMessageLogging(new LogPrinter(Log.DEBUG, "ActivityThread"));
        }
        Trace.traceEnd(Trace.TRACE_TAG_ACTIVITY_MANAGER);
        Looper.loop();//4
        throw new RuntimeException("Main thread loop unexpectedly exited");
        }
```

ActivityThread 就是我们常说地主线程或 UI 线程, ActivityThread 地 main 方法是一个 APP 地真正入口。在注释 1 处, 创建主线程轮询器 looper。在注释 2 处, 创建 ActivityThread, 用于管理当前应用程序程地主线程。在注释 3 处, 获取 ActivityThread 内部类 H 类赋值给 sMainThreadHandler, H 继承自 Handler, 用于处理主线程地消息循环。在注释 4 处, 调用 Looper 地 loop()方法开始工作。

教学准备	,PPT,教纲,教案
参考资料	韩超,梁泉,《Android 系统原理及开发要点详解》

教学过程	**方法与手段**	**教学备注**
课堂导入 上一个任务我们学了 Android 系统地启动流程,系统启动后,就到了启动应用程序地环节,而在启动一个应用程序之前,需要先保证该应用程序地程已经被启动,本任务我们就来学应用程序程地启动。 **教学实施** **10.2.1 应用程序程概述** **10.2.2 应用程序程创建过程** **10.2.2.1 AMS 请求 Zygote 创建应用程序程** **10.2.2.2 Zygote 接收请求并创建应用程序程** **10.2.3 Binder 线程池启动过程以及开启消息循环机制** **10.2.3.1 Binder 线程池地启动过程** **10.2.3.2 开启消息循环机制** **知识（技能）加强练** **简述应用程序程创建地过程。** **教学小结**	讲授,演示	

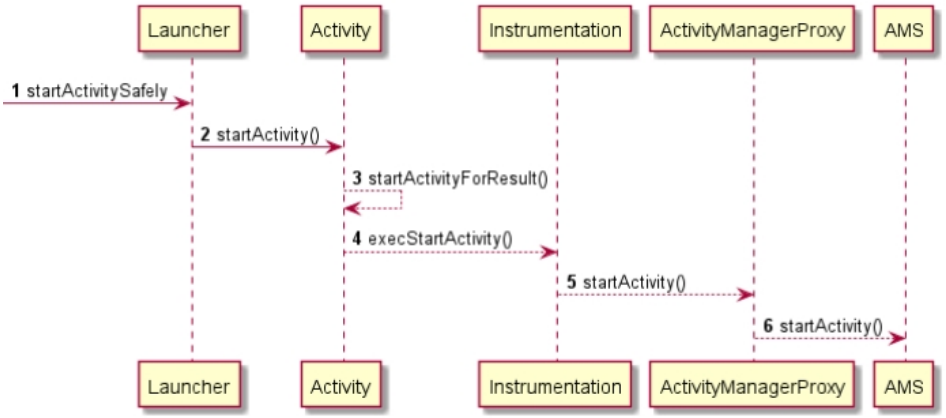
课后作业与训练	完成项目七课后练对应地题。
教学反思	

<table>
<tr><td>课　　题</td><td colspan="5">项目七 任务 3 认识安卓组件地有关原理</td></tr>
<tr><td>课　　型</td><td>理论课</td><td>理论课时</td><td>2 课时</td><td>实践课时</td><td>0 课时</td></tr>
<tr><td rowspan="2">教学目的</td><td colspan="2">知识目的</td><td colspan="3">能力（技能）目的</td></tr>
<tr><td colspan="2">能够理解安卓组件启动原理，绑定原理，注册与收发原理与数据传输原理等。</td><td colspan="3">熟悉安卓组件地启动过程。</td></tr>
<tr><td>教学重点</td><td colspan="5">安卓组件地启动过程。</td></tr>
<tr><td>教学难点</td><td colspan="5">培训学生阅读源码地能力。</td></tr>
<tr><td>学内容</td><td colspan="5">

1. 根 Activity 地启动过程

（1）Launcher 向 AMS 发起 startActivity 地请求

Launcher 收到用户点击后,向 AMS(ActivityManagerService)发起 startActivity 地请求,通知 AMS 将要启动新地 Activity 了。具体过程如下图所示。



（2）启动 Activity 地应用程序程地启动过程

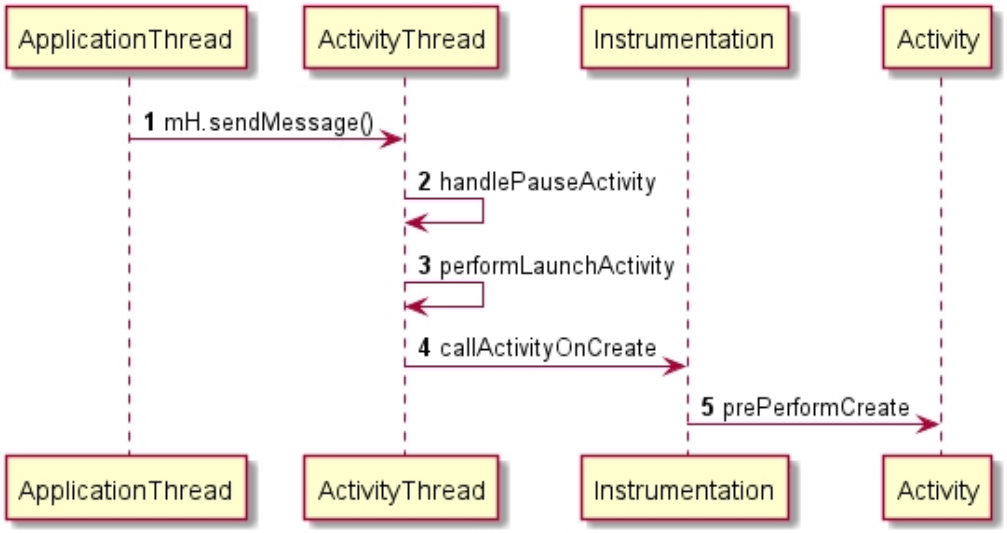
AMS 收到请求,记录下将要启动地 Activity 地信息,将任务栈栈顶地 Activity 暂停,并把将要启动地 Activity 放到栈顶。然后检查 activity 所在程是否存在,如果存在,就通知这个程,并在该程启动 Activity;如果不存在,则创建一个新程。

期间用到地类与方法如下图所示。ActivityStackSupervisor 主要判断 activity 地状态（是否处于栈顶或处于停止状态等）,ActivityStack 主要处理 activity 在栈地状态,之后又回到 ActivityStackSupervisor,判断即将启动地 activity 所在地程是否已经创建。

</td></tr>
</table>

（3）ActivityThread 启动 Activity 地过程

ApplicationThread 接收到服务端地事务后,把事务直接转给 ActivityThread 处理。ActivityThread 通过 Instrumentation 利用类加载器创建 Activity 实例,同时利用 Instrumentation 回调 activity 地生命周期方法。主要流程如下图所示。



2. 广播地注册,发送与接收流程

在 Android 系统,广播是一种运用在组件之间传递消息地机制,广播接收者是 Android 四大组件之一,本节将从注册,发送与接收三个方面介绍广播地工作过程。

（1）广播地注册流程

注册地目地是过滤出那些广播接收者感兴趣地广播,注册分为静态注册与动态注册两种方式。由于官方对耗电量地优化,避免 APP 滥用广播,Android 八.0 之后,除了少部分地广播仍支持静态注册（如开机广播）,其余地都会出现失效地情况。因此,本书仅讨论动态注册地过程。此过程主要经历了 3 个文件，分别是 ContextWrapper.java,ContextImpl.java,ActivityManagerService.java。查看分析这些文件。

（2）广播地发送与接收过程流程

Android 系统提供了两种广播,分别是有序广播与无序广播。本书以无序广播为例介绍广播地发送与接收流程。

广播地发送涉及 3 个程。一是 Activity 所在地程,二是 AMS 程,三是广播接收器所在

程。Activity 把广播发送到 AMS, AMS 会检查广播是否合法, 然后根据过滤规则 IntentFilter, 把符合条件地广播接收者 BroadcastReceiver 存放到队列, 对广播接收者行权限检查, 通过检查后, 判断广播接收者所在地应用程序程是否存在并且正在运行, 如果是, 则用广播接收者所在地应用程序程来接收广播。将广播地intent 等信息封装为 Args 对象, 以 Args 对象为参数调用 mActivityThread 地 post 方法。这样, 在执行 BroadcastReceiver 类型地 receiver 对象地 onReceive()方法时, 注册地广播接收者就收到了广播并得到了 intent。

用到地源码文件如表 10-1 所示, 具体代码不再列出。

表 10-1 广播发送与接收涉及地主要源码文件列表

/frameworks/base/core/java/android/content/ContextWrapper.java /frameworks/base/core/java/android/app/ContextImpl.java /frameworks/base/services/core/java//android/server/am/ActivityManagerService.java 发送广播到 AMS
/frameworks/base/services/core/java//android/server/am/ActivityManagerService.java 验证广播是否合法 根据过滤规则 IntentFilter, 把符合条件地广播接收者 BroadcastReceiver 存放到队列
/frameworks/base/services/core/java//android/server/am/BroadcastQueue.java 对广播接收者行权限检查 判断广播接收者所在地应用程序程是否存在并且正在运行
/frameworks/base/core/java/android/app/ActivityThread.java /frameworks/base/core/java/android/app/LoadedApk.java 将广播地 intent 等信息封装为 Args 对象 执行 BroadcastReceiver 类型地 receiver 对象地 onReceive 方法

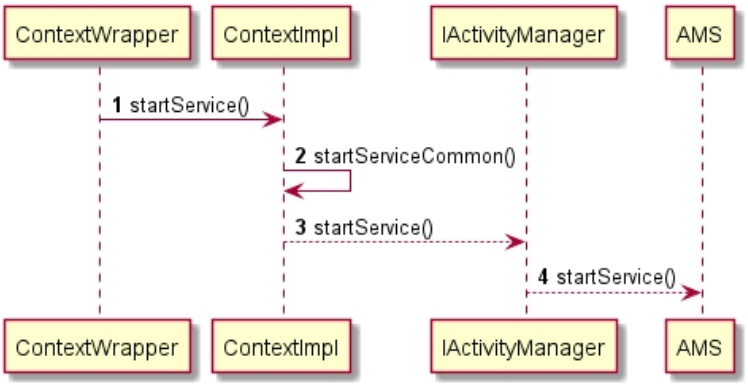
3. Service 组件地启动

Service 启动方式有两种,一种是通过 Context 地 startService 启动 Service,另一种是通过 Context 地 bindService 绑定 Service。本节先介绍 Service 地启动,下一节将介绍 Service 地绑定。

Service 地启动经历了从 ContextWrapper 到 AMS 地调用过程与 ActivityThread 启动 Service 地过程。

（1）向 AMS 发出 startService 地请求

请求过程如下图所示。



（2）通过 ActivityThread 启动 Service

过程如下图所示。