目录

简介

本文档主要介绍Delta-II OTA流程实现过程中的关键位置, 以及主要修改, 红色字体标识放弃的设计, 蓝色字体标识Delta-II OTA流程中新增的部分;
限于篇幅, 未将所有源代码列出, 仅仅讨论了OTA过程的主要关键节点;

编译选项

- 常规升级包制作，在Delta II项目工程目录下执行命令：
  make otapackage
- 默认不对Recovery进行imgdiff(Delta-II要求可以单独更新recovery.img，因此不做差分)：
  TARGET_USE_RECOVERY_DIFF=false make otapackage
- 默认不检查版本号(Delta-II)，如果要求检查版本号(不允许降级)，则：
  TARGET_USE_NO_PREREQ=false make otapackage
- 默认对所有的OTA包文件进行加密(Delta-II)，如果不需要进行加密，则：
  TARGET_USE_NO_PREREQ=true make otapackage
- 默认OTA包不包含bootloader的更新(Delta-II)，如果需要制作带有bootloader的OTA更新包，则：
  TARGET_USE_NO_PREREQ=true make otapackage

升级脚本

- 比较版本号，禁止降级(Delta-II OTA不再使用)

  (!less_than_int(1488016244, getprop("ro.build.date.utc"))) || abort("Can't install this package (2017年 02月 25日 星期六 17:50:44 CST) oer newer build (" + getprop("ro.build.date") + ").");

- 比对产品名称是否匹配

  getprop("ro.product.device") == "delta2" || abort("This package is for \"delta2\" devices; this is a \"" + getprop("ro.product.device") + "\".");

- 设置UI显示

  led_ui(2);

- 打印进度

  show_progress(0.800000, 80);

- 设置环境变量upgrade_step

  set_bootloader_env("upgrade_step", "3");

- 格式化/dev/block/system分区为ext4文件系统

  format("ext4", "EMMC", "/dev/block/system", "0", "/system");

- 挂载/dev/block/system文件系统

  mount("ext4", "EMMC", "/dev/block/system", "/system");

- 释放升级包中recovery/目录下的文件到/dev/block/system分区中

  package_extract_dir("recovery", "/system");

- 释放升级包中system/目录下的文件到/dev/block/system分区中

  package_extract_dir("system", "/system");

- 创建/dev/block/system分区中的所有符号链接

  symlink("busybox", "/system/xbin/[", "/system/xbin/[[",
  "/system/xbin/adjtimex", "/system/xbin/arp", "/system/xbin/ash",
  … …

- 设置/system分区的文件权限

  set_metadata_recursive("/system", "uid", 0, "gid", 0, "dmode", 0755, "fmode", 0644, "capabilities", 0x0, "selabel", "u:object_r:system_file:s0");
  set_metadata_recursive("/system/bin", "uid", 0, "gid", 2000, "dmode", 0755, "fmode", 0755, "capabilities", 0x0, "selabel", "u:object_r:system_file:s0");
  set_metadata("/system/bin/clatd", "uid", 0, "gid", 2000, "mode", 0755, "capabilities", 0x0, "selabel", "u:object_r:clatd_exec:s0");
  … …

- 如果由Touch固件需要更新，则：

  touchupdate("/system/lib/touch_firmware_v3.bin");

- 接触/system文件系统的挂载

  unmount("/system");

- 更新recovery，并给出进度提示(按照Delta-II的OTA流程要求，此时Recovery已经时最新的，不需要在更新)；

  show_progress(0.100000, 10);

- 更新boot分区以及bootloader分区(Delta-II要求默认不对Bootloader进行更新)

  write_raw_image(package_extract_file("boot.img"), "boot");
  write_raw_image(package_extract_file("bootloader.img"), "bootloader");

- 写入升级成功的指示

  set_bootloader_env("upgrade_step", "1");
  show_progress(0.100000, 0);

- 延迟3s，关闭UI的呈现

  sleep(3);
  led_ui(255);

OTA包目录结构

```
out/target/product/delta2/delta2-ota-xxxxxx.zip(Delta-II要求对该包内除encode_list以外的所有文件进行加密)
├── boot.img                // 正常启动的Kernel与Ramdisk
├── bootloader.img          // 引导(Delta-II默认不包含)
├── file_contexts           // SeLinux File Context
├── logo.img                // 开机logo
├── META-INF
│   ├── CERT.RSA            // 保存了公钥、所采用的加密算法等信息
│   ├── CERT.SF             // 对摘要的签名文件
│   ├── com
│   │   ├── android
│   │   │   ├── metadata
│   │   │   └── otacert
│   │   └── google
│   │       └── android
│   │           ├── update-binary
│   │           └── updater-script
│   └── MANIFEST.MF         // 摘要文件
├── recovery                // Recovery安装脚本以及安装包(Delta-II默认不包含)
│   ├── etc
│   └── recovery-from-boot.p
├── recovery.img      // Delta-II新增, 不对Recovery进行imgdiff, 直接使用img文件
├── encode_list             // Delta-II新增, 该文件默认不加密, 列出已经加密过的文件
└── system                  // 系统分区文件
    ├── bin
    ├── build.prop
    ├── chksum_list
    ├── etc
    ├── framework
    ├── lib
    ├── media
    ├── usr
    ├── vendor
    └── xbin
```

启动过程

```
preboot ->
     if itest ${upgrade_step} == 1; then
          // Delta-II 要求此处不能重设Env
          defenv; setenv upgrade_step 2; saveenv;
     …
     run check_rebootmode; ->
          // 如果是复位出厂设置, 则执行defenv进行环境变量的复位操作, Delta-II 要求此处不能重设Env
          if test ${reboot_mode} = factory_reset; then
               defenv;
          fi;
```

系统构建

build/core/Makefile

```
# 对于recovery.img文件系统的打包
touch_test_binary := $(call intermediates-dir-for,EXECUTABLES,touch_test)/touch_test
encode_binary := $(call intermediates-dir-for,EXECUTABLES, encode)/encode
touch_update_binary := $(call intermediates-dir-for,EXECUTABLES, touch_update)/touch_update
… …
ifneq ($(touch_test_binary),)
        $(info Copy $(touch_test_binary) to recovery, added by James.Li)
        cp -f $(touch_test_binary) $(TARGET_RECOVERY_ROOT_OUT)/sbin/
endif
ifneq ($(encode_binary),)
        $(info Copy $(encode_binary) to recovery, added by James.Li)
        cp -f $(encode_binary) $(TARGET_RECOVERY_ROOT_OUT)/sbin/
endif
ifneq ($(touch_update_binary),)
        $(info Copy $(touch_update_binary) to recovery, added by James.Li)
        cp -f $(touch_update_binary) $(TARGET_RECOVERY_ROOT_OUT)/sbin/
endif

# Delta-II新增OTA打包脚本的参数选项
INTERNAL_OTA_PACKAGE_TARGET := $(PRODUCT_OUT)/$(name).zip
$(INTERNAL_OTA_PACKAGE_TARGET): $(BUILT_TARGET_FILES_PACKAGE) $(DISTTOOLS)
        $(hide) ./build/tools/releasetools/ota_from_target_files -v \
           $(amlogic_flag) \
           $(omit_prereq_flag) \
           -p $(HOST_OUT) \
           $(wipeopt) \
           $(baksupport) \
           -k $(KEY_CERT_PAIR) \
           $(recovery_not_patch) \
           $(no_encode) \
           $(wipe_cache_opt) \
           $(dm_verity) \
           $(no_prereq) \
           $(bootloader_opt) \
           $(recover_diff) \
           $(BUILT_TARGET_FILES_PACKAGE) $@
…
# Delta-II新增OTA版本号更新，其中upmver为模组版本号更新，定义在core/main.mk
.PHONY: upover
upover:
       $(hide) ./device/amlogic/$(TARGET_PRODUCT)/update-version.py device/amlogic/$(TARGET_PRODUCT)/system.prop
ro.product.otapackage.version

.PHONY: upver
upver: upmver upover

otapackage: upver droidcore dist_files $(INTERNAL_OTA_PACKAGE_TARGET)
```

build/core/main.mk

```
# Delta-II新增模组版本号更新
# Update module software version
.PHONY: upmver
upmver:
       $(hide) ./device/amlogic/$(TARGET_PRODUCT)/update-version.py device/amlogic/$(TARGET_PRODUCT)/system.prop
ro.module.sw.version

 # Building a full system-- the default is to build droidcore
droid: upmver droidcore dist_files
```

build/tools/releasetools/ota_from_target_file

```
# 为Delta-II OTA脚本增加必要的参数选项
OPTIONS = common.OPTIONS -> build/tools/releasetools/common.py: OPTIONS
      ...
      OPTIONS.encode_list = []
      ...
      OPTIONS.encode=True
      OPTIONS.bootloader=False

# 为Delta-II OTA打包脚本编写的加密方法
```

```
def EncodeFile(fpath, append=''):
    ...

# 主函数
if __name__ == '__main__':
    main(sys.argv[1:]) ->
        args = common.ParseOptions(...) -> build/tools/releasetools/common.py: ParseOptions()
            # 为Delta-II OTA准备的参数解析部分
            opts, args = getopt.getopt(..., "no_encode", "bootloader"] + list(extra_long_opts)))
            for o, a in opts:
                if o in ("-h", "--help"):
                    Usage(docstring)
                    sys.exit()
                elif o in ("-a", "--amlogic"):
                    ...
                ...
                else:
                    if extra_option_handler is None or not extra_option_handler(o, a): -> \
                            build/tools/releasetools/ota_from_target_files: main.option_handler() ->
                        if o in ("-b", "--board_config"):
                            pass
                        elif o in ("-k", "--package_key"):
                            ...
                        ...
                        # Libratone
                        elif o in ("--no_encode"):
                            OPTIONS.encode = False
                        elif o in ("--bootloader"):
                            OPTIONS.bootloader = True
                        else:
                            return False
                        return True
# 只处理整包更新的情况
if OPTIONS.incremental_source is None:
    WriteFullOTAPackage(input_zip, output_zip) ->
        if not OPTIONS.omit_prereq:
            ts = GetBuildProp("ro.build.date.utc", OPTIONS.info_dict)
            ts_text = GetBuildProp("ro.build.date", OPTIONS.info_dict)
        # 在OTA打包的扩展脚本中添加了相关的updater-script语句的生成代码
        device_specific.FullOTA_InstallBegin() -> \
                build/tools/releasetools/common.py: DeviceSpecificParams.FullOTA_InstallBegin()
            return self._DoCall("FullOTA_InstallBegin") -> \
                    device/amlogic/coco/releasetools.py: FullOTA_InstallBegin() ->
                self.script.SetUIStatus(2) ->\
                        build/tools/releasetools/edify_generator.py: EdifyGenerator.SetUIStatus() ->
                    self.script.append('led_ui(%d);' % val)
        # Delta-II OTA要求所有的system/路径下的文件都进行加密
        symlinks = CopySystemFiles(input_zip, output_zip) ->
            for info in input_zip.infolist():
                if IsSymlink(info):
                    ...
                else
                if output_zip is not None:
                    if substitute and fn in substitute:
                        ...
                    else
                        data = EncodeFile(fpath=info.filename, append='system/')
                    ...
                    output_zip.writestr(info2, data)
# 略过的代码部分也相应的进行了加密，下文将以Recovery为例介绍img文件加密的实现
…
# 处理Recovery的打包过程，也同样进行加密
if OPTIONS.recovery_not_patch:
    if recovery_img_exists:
        common.CheckSize(recovery_img.data, "recovery.img", OPTIONS.info_dict)
        common.ZipWriteStr(output_zip, "recovery.img", recovery_img.data) ->\
                build/tools/releasetools/common.py: ZipWriteStr() ->
            if OPTIONS.encode and filename != 'encode_list':
                ori_fobj = open(ori_fpath, 'w+')
                if ori_fobj:
                    ...
                    encode_cmd = ['encode', '-e', '-key:168', ori_fpath, ecd_fpath]
                    p = Run(encode_cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
                    if OPTIONS.encode and filename != 'encode_list':
                        zip.writestr(zinfo, _data)
                    else
                        zip.writestr(zinfo, data)
```

```
                    # 对Touch固件升级的单独处理
                    touch_fw_ver = GetBuildProp('ro.product.firmware.revision', OPTIONS.info_dict)
                    if touch_fw_ver != None:
                        fw_fname = "touch_firmware_v%d.bin" % (int(touch_fw_ver))
                        if os.path.isfile(OPTIONS.input_tmp + '/SYSTEM/lib/' + fw_fname):
                            script.TouchUpdate(fw_full_name)
                    # 对于bootloader增加可选参数相关的控制代码
                    if OPTIONS.bootloader:
                        if bootloader_img_exist:
                            if _bootloader_img_data:
                                common.ZipWriteStr(output_zip, "bootloader.img", _bootloader_img_data)
```

 # 说明：zip写入时的加密过程有些繁琐，原因在于虽然ZipWriteStr()和CopySystemFiles()虽然最终都是调用zip.writestr()方法来写入文件到zip，但是无法重载zip.writestr()方法，因此导致必须分别进行处理；

device/amlogic/coco/coco.mk

```
# 增加make参数开关控制
TARGET_USE_RECOVERY_DIFF :=true
TARGET_USE_NO_PREREQ := true
TARGET_USE_NO_ENCODE := false
TARGET_NEED_BOOTLOAER := false
# 增加OTA脚本的拷贝
PRODUCT_COPY_FILES += \
        device/amlogic/common/ota.sh:system/bin/ota.sh
# 检查如果存在Touch固件，则进行拷贝，如果没有，什么都不做
touch_firmware_ver := $(shell cat device/amlogic/$(TARGET_PRODUCT)/system.prop | grep ro.product.firmware.revision)
ifeq ($(touch_firmware_ver), )
$(info device/amlogic/$(TARGET_PRODUCT)/system.prop里没有版本号?)
else
touch_firmware_ver_num := $(shell echo $(touch_firmware_ver) | awk -F= '{print $$2}')
#$(info touch_firmware_ver_num: $(touch_firmware_ver_num))
touch_firmware := device/amlogic/$(TARGET_PRODUCT)/touch_firmware_v${touch_firmware_ver_num}.bin
touch_firmware_exist := $(shell if [ -f $(touch_firmware) ]; then echo ok; fi)
ifeq ($(touch_firmware_exist), ok)
PRODUCT_COPY_FILES += \
        $(touch_firmware):system/lib/$(shell basename $(touch_firmware))
endif
endif
```

针对Recovery

通过调用关系介绍具体的更改；

```
main() ->
     if (update_patch != NULL)
          status = install_package(update_patch, &wipe_cache, TEMPORARY_INSTALL_FILE); -> really_install_package(path,
wipe_cache); -> try_update_binary(path, &zip, wipe_cache); ->
               // 读取encode_list
               ok = init_encode_list(zip);
               // 读取update-binary并解密
               const ZipEntry* binary_entry = mzFindZipEntry(zip, ASSUMED_UPDATE_BINARY_NAME);
               … …
               ok = mzExtractZipEntryToFile(zip, binary_entry, fd);
               if(is_encode_file(binary_entry))
                    int ret = decode_file(binary);
               … …
               pid_t pid = fork();
               if (pid == 0)
                    execv(binary, (char* const*)args); --> updater.c:main() ->
                         … …
                         const ZipEntry* script_entry = mzFindZipEntry(&za, SCRIPT_NAME);
                         mzReadZipEntry(&za, script_entry, script, script_entry->uncompLen)
                         // Delta-II要求必须对所有的解压缩过程进行解密
                         if(is_encode_file(script_entry))
                              roledata((unsigned char *)script, script_entry->uncompLen, 1, 168);
                         … …
                         RegisterInstallFunctions(); ->
                              … …
                              // 注册Delta-II要求的LED_UI显示状态命令以及Touch固件升级命令
                              RegisterFunction("led_ui", LedUI);
                              RegisterFunction("touchupdate", TouchUpdate);
                         char* result = Evaluate(&state, root); ->
                              Value* v = expr->fn(expr->name, state, expr->argc, expr->argv); -->
                                   … …
                                   // Delta-II的LED_UI实现方法为：管道给sepres_test进程相应的命令进行LED UI的切换
                                   led_ui(2); --> LedUI ->
                                        ReadArgs(state, argv, 1, &led_state)
                                        pipe_fd = open(FIFO_NAME, O_WRONLY|O_NONBLOCK);
                                        res = write(pipe_fd, &l_char, 1);
                                   … …
                                   package_extract_dir("system", "/system"); --> PackageExtractDirFn ->
                                        ZipArchive* za = ((UpdaterInfo*)(state->cookie))->package_zip;
                                        bool success = mzExtractRecursive(za, zip_path, dest_path, \
                                                  MZ_EXTRACT_FILES_ONLY, &timestamp, NULL, NULL, \
                                                  sehandle); ->
                                        for (i = 0; i < pArchive->numEntries; i++)
                                             ZipEntry *pEntry = pArchive->pEntries + i;
                                             if (pEntry->fileName[pEntry->fileNameLen-1] == '/')
                                                  … …
                                             else
                                                  … …
                                             if (!(flags & MZ_EXTRACT_FILES_ONLY)
                                                       && mzIsZipEntrySymlink(pEntry))
                                                  … …
                                             else
                                                  … …

                                             if(is_encode_file(pEntry))
                                                  int ret = decode_file(targetFile);
                                   … …
                                   // Delta-II Touch固件更新的实现方法为：调用touch_update进程执行升级命令
                                   touchupdate("/system/lib/touch_firmware_v3.bin"); --> TouchUpdate ->
                                        sprintf(cmd_buf, "touch_update %s", touch_fw_path);
                                        int ret = system(cmd_buf);
                                   … …
                                   write_raw_image(package_extract_file("logo.img"), "logo"); --> \
                                             PackageExtractFileFn ->
                                        // Detal-II的OTA同样需要对PackageExtractFileFn()解压文件的过程添加解密的部分
                                        if (argc == 2)
                                             if(is_encode_file(entry))
                                                  int ret = decode_file(dest_path);
                                        else
                                             if(is_encode_file(entry))
                                                  int ret = decode_file(dest_path);
                                   … …
```