Embedded Android+Automotive

Solutions

Version v 12.0.4.1

March 2022

Table of Contents

1 Preparation

No solution for this lab

2 The Android Open Source Project

2.1 Build AOSP

No solution given for this exercise

2.2 Repo and lunch

Running `repo info` gives a summary of the manifest and the projects that it contains:

```
$ repo info
Manifest branch: refs/tags/android-12.0.0_r26
Manifest merge branch: refs/heads/android-12.0.0_r26
Manifest groups: all,-notdefault
----------------------------
[...]
```

Running godir

```
$ godir SurfaceFlinger.cpp
Creating index... Done
$ pwd
/home/training/aosp/frameworks/native/services/surfaceflinger
```

Use **cgrep** to find all C/C++ files that contain a main() function:

```
$ cgrep "main*("
./main_surfaceflinger.cpp:80:int main(int, char**) {
./tests/fakehwc/SFFakeHwc_test.cpp:2009:int main(int argc, char** argv) {
./tests/waitforvsync/waitforvsync.cpp:32:int main(int /*argc*/, char** /*argv*/) {
./tests/vsync/vsync.cpp:54:int main(int /*argc*/, char** /*argv*/)
./tests/unittests/libsurfaceflinger_unittest_main.cpp:44:int main(int argc, char **argv) {
./tests/BufferGeneratorShader.h:29:void main() {
./tests/BufferGeneratorShader.h:314:void main() {
```

The command to search make files (and also blueprint files) is **mgrep**. Searching for protobuf produces this list:

```
$ mgrep protobuf
./CompositionEngine/Android.bp:23:        "libprotobuf-cpp-lite",
./layerproto/Android.bp:15:        "libprotobuf-cpp-lite",
./layerproto/Android.bp:52:            static_libs: ["libprotobuf-java-nano"],
./Android.bp:52:        "libprotobuf-cpp-lite",
./TimeStats/timestatsproto/Android.bp:12:        "libprotobuf-cpp-lite",
./TimeStats/Android.bp:11:        "libprotobuf-cpp-lite",
./tests/Android.bp:59:        "libprotobuf-cpp-full",
./tests/Android.bp:101:        "libprotobuf-cpp-full",
```

2.3 The build ID

The build is stored in `build/core/build_id.mk`. It contains this line:

```
BUILD_ID=SQ1A.220105.002
```


3 Device configuration

3.1 Image files

Using the **file** command to describe the contents of the image files shows the following:

```
$ cd $OUT
$ file *.img
cache.img:               Linux rev 1.0 ext4 filesystem data [...]
dtb.img:                 ASCII text
encryptionkey.img:       DOS/MBR boot sector; partition 1 [...]
ramdisk-debug.img:       gzip compressed data, from Unix
ramdisk.img:             gzip compressed data, from Unix
ramdisk-qemu.img:        gzip compressed data, from Unix
super_empty.img:         data
super.img:               data
system.img:              Linux rev 1.0 ext2 filesystem data [...]
system-qemu.img:         DOS/MBR boot sector; partition 1 [...]
userdata.img:            Linux rev 1.0 ext4 filesystem data [...]
userdata-qemu.img:       Linux rev 1.0 ext4 filesystem data
vbmeta.img:              data
vendor_boot-debug.img:   data
vendor_boot.img:         data
vendor.img:              Linux rev 1.0 ext2 filesystem data [...]
vendor-qemu.img:         DOS/MBR boot sector; partition 1 [...]
```

Note that system.img and vendor.img are ext2, not ext4. The ext2 filesystem is a subset of ext4 but without journalling. Since system and vendor are read-only a journal has no benefit and just takes up some space in the partition. So, it is more efficient to use ext2 on read-only partitions.

3.2 List packages

Using script `list-product-packages.sh` to count the number of packages and modules shows this:

```
$ $HOME/android/list-product-packages.sh -b | wc -l
910
```

3.3 View the build log

No solution given for this exercise

3.4 (Optional) generate a product makefile dependency graph

No solution given for this exercise

4 The kernel

4.1 Kernel version and modules

The version string of the pre-built kernel is

```
# uname -a
Linux localhost 5.10.43-android12-9-00031-g02d62d5cece1-ab7792588 #1 SMP PREEMPT Mon Oct 4 2
1:48:11 UTC 2021 i686
```

Here is a list of the kernel modules:

```
# lsmod
zram                   28672  1
zsmalloc               36864  1 zram
vmw_vsock_virtio_transport    24576  2
virtio_pmem            16384  1
virtio_pci             32768  0
virtio_net             57344  0
virtio_mmio            20480  0
virtio_mem             28672  0
virtio_input           20480  0
virtio_console         40960  0
virtio_blk             24576  6
virtio_balloon         28672  0
virtio_rng             16384  0
virtio_gpu             90112  35
virtio_dma_buf         16384  1 virtio_gpu
virt_wifi_sim          20480  0
virt_wifi              24576  2 virt_wifi_sim
vcan                   16384  0
tpm_vtpm_proxy         16384  0
tpm                    65536  1 tpm_vtpm_proxy
test_stackinit         28672  0
test_meminit           16384  0 [permanent]
system_heap            20480  0 [permanent]
snd_intel8x0           40960  0
snd_hda_intel          45056  0
snd_intel_dspcfg       16384  1 snd_hda_intel
snd_hda_codec_realtek   155648  0
snd_hda_codec_generic    94208  1 snd_hda_codec_realtek
snd_hda_codec         159744  3 snd_hda_intel,snd_hda_codec_realtek,snd_hda_codec_generic
snd_hda_core          110592  4 snd_hda_intel,snd_hda_codec_realtek,snd_hda_codec_generic,sn
snd_ac97_codec        229376  1 snd_intel8x0
slcan                  16384  0
rtc_test               16384  0
pulse8_cec             24576  0
psmouse               126976  0
net_failover           20480  1 virtio_net
nd_virtio              16384  1 virtio_pmem
md_mod                172032  0
mac80211_hwsim         69632  0
mac80211              933888  1 mac80211_hwsim
lzo                    16384  0
lzo_rle                16384  0
ledtrig_audio          16384  1 snd_hda_codec_generic
hci_vhci               16384  1
gs_usb                 20480  0
```



```
goldfish_sync          20480  0
goldfish_pipe          28672  0
goldfish_battery       16384  0
goldfish_address_space    20480  0
gnss_cmdline           16384  0
gnss_serial            16384  1 gnss_cmdline
failover               16384  1 net_failover
dummy_cpufreq          16384  0
cfg80211              954368  4 virt_wifi_sim,virt_wifi,mac80211_hwsim,mac80211
ac97_bus               16384  1 snd_ac97_codec
```

4.2 Integrate the kernel into your device

This is the default kernel from prebuilts:

```
$ cd $HOME/aosp
$ ../android/list-product-copy-files.sh | grep kernel
kernel/prebuilts/5.10/x86_64/kernel-5.10
```

And, these are the default kernel modules

```
$ get_build_var BOARD_VENDOR_RAMDISK_KERNEL_MODULES | tr " " "\n"
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/ac97_bus.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/cfg80211.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/dummy-cpufreq.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/failover.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/gnss-cmdline.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/gnss-serial.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/goldfish_address_space.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/goldfish_battery.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/goldfish_pipe.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/goldfish_sync.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/gs_usb.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/hci_vhci.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/ledtrig-audio.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/lzo-rle.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/lzo.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/mac80211.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/mac80211_hwsim.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/md-mod.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/nd_virtio.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/net_failover.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/psmouse.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/pulse8-cec.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/rtc-test.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/slcan.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-ac97-codec.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-hda-codec-generic.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-hda-codec-realtek.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-hda-codec.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-hda-core.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-hda-intel.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-intel-dspcfg.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/snd-intel8x0.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/system_heap.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/test_meminit.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/test_stackinit.ko
```



```
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/tpm.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/tpm_vtpm_proxy.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/vcan.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virt_wifi.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virt_wifi_sim.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio-gpu.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio-rng.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_balloon.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_blk.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_console.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_dma_buf.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_input.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_mem.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_mmio.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_net.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_pci.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/virtio_pmem.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/vmw_vsock_virtio_transport.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/zram.ko
kernel/prebuilts/common-modules/virtual-device/5.10/x86-64/zsmalloc.ko
```

Add this to `device/sirius/marvin/device.mk`

```
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/bzImage:kernel
```

Add this to `device/sirius/marvin/BoardConfig.mk`

```
BOARD_VENDOR_RAMDISK_KERNEL_MODULES := $(wildcard device/sirius/marvin/ko/*.ko)
```

After rebuilding and booting Cuttlefish, the kernel version string is now:

```
$ uname -a
Linux localhost 5.10.66-android12-9-00018-g87a74496ed4a #1 SMP PREEMPT Wed Jan 12 17:15:55 U
TC 2022 i686
```


5 Bootloaders

No solution for this lab

6 Boot storage layout

No solution for this lab

7 ADB and logcat

7.1 ADB

No solution give for this exercise

7.2 Using ADB to install an app

No solution give for this exercise

7.3 Logcat

The output of **adb logcat -g -b all** on Emulator is:

```
$  adb logcat -g -b all
main: ring buffer is 2 MiB (1 MiB consumed), max entry is 5120 B, max payload is 4068 B
radio: ring buffer is 2 MiB (717 KiB consumed), max entry is 5120 B, max payload is 4068 B
events: ring buffer is 2 MiB (35 KiB consumed), max entry is 5120 B, max payload is 4068 B
system: ring buffer is 2 MiB (146 KiB consumed), max entry is 5120 B, max payload is 4068 B
crash: ring buffer is 2 MiB (0 B consumed), max entry is 5120 B, max payload is 4068 B
stats: ring buffer is 64 KiB (0 B consumed), max entry is 5120 B, max payload is 4068 B
kernel: ring buffer is 2 MiB (1 MiB consumed), max entry is 5120 B, max payload is 4068 B
```

This command displays error messages

```
$ adb logcat *:e
```

It also displays messages of higher priority than error, i.e. **F**atal

To show messages for SurfaceFlinger, you can use either of these commands:

```
$ adb logcat SurfaceFlinger *:s
$ adb logcat -s SurfaceFlinger
```

To show only messages of priority error or higher from SurfaceFlinger, use either of these commands:

```
$ adb logcat SurfaceFlinger:e *:s
$ adb logcat -s SurfaceFlinger:e
```


8 Android start-up

8.1 Start-up scripts

Listing the start-up scripts parsed by init shows this:

```
# dmesg | grep "init: Parsing"
[    7.113036] init: Parsing file /system/etc/init/hw/init.rc...
[    7.117150] init: Parsing file /init.environ.rc...
[    7.117829] init: Parsing file /system/etc/init/hw/init.usb.rc...
...
```

A total of 137 rc files are parsed by init

The hardware platform is:

```
# getprop ro.hardware
cutf_cvm
```

Hence the location of the hardware init scripts is `/vendor/etc/init/hw/init.cutf_cvm.rc`

This value is read from the bootconfig:

```
# cat /proc/bootconfig
androidboot.hardware = "cutf_cvm"
[...]
```

8.2 Boot scripts

Add the following to `device/sirius/marvin/init.cutf_cvm.rc`

```
[...]
on boot
    write /data/vendor/message.txt "Reached on boot"
[...]
```

8.3 Native daemons

```
# getprop | grep init.svc
[init.svc.adbd]: [running]
[init.svc.apexd]: [stopped]
[init.svc.apexd-bootstrap]: [stopped]
[init.svc.apexd-snapshotde]: [stopped]
[init.svc.audioserver]: [running]
[init.svc.bootanim]: [stopped]
[...]
```

8.4 Stop and start

No solution given for this exercise

8.5 (Optional) Boot animation

Answer to question: the default boot animation is hard-coded into the bootanimation application: there is no zip file. If you look at `frameworks/base/cmds/bootanimation/BootAnimation.cpp` you can see that the default animation is created by function `BootAnimation::android()`, using images from `frameworks/base/core/res/assets/images`

8.6 (Optional) Properties

No solution given for this exercise

9 Android packages and modules

9.1 Hello world

The Android.bp file should look like this:

```
cc_binary {
    name: "helloworld-bp",
    srcs: ["helloworld.c"],
    vendor: true,
}
```

Here is the updated module list, including helloworld-bp

```
$ refreshmod
Refreshing modules (building module-info.json)...
$ allmod | grep helloworld
helloworld-bp
```

```
$ ls  $OUT/vendor/bin/helloworld*
/home/ubuntu/aosp/out/target/product/marvin/vendor/bin/helloworld-bp
```

9.2 Add your module to system images

```
PRODUCT_PACKAGES += helloworld-bp
```

9.3 Android.mk

The Android.mk file should look like this

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE := helloworld-mk
LOCAL_SRC_FILES := helloworld.c
LOCAL_VENDOR_MODULE := true

include $(BUILD_EXECUTABLE)
```

```
$ ls  $OUT/system/bin/helloworld*
/home/chris/aosp/out/target/product/marvin/vendor/bin/helloworld-bp
/home/chris/aosp/out/target/product/marvin/vendor/bin/helloworld-mk
```

```
$ refreshmod
Refreshing modules (building module-info.json)...
$ allmod | grep helloworld
helloworld-bp
helloworld-mk
```

```
PRODUCT_PACKAGES += helloworld-mk
```


9.4 List modules

On android-11.0.0_r27, allmod repots:

```
$ allmod | wc -l
38754
```


10 SELinux

10.1 Adding sepolicy for a web server

In `device/sirius/marvin/sepolicy`, you will need to create file `httpd.te` with these lines:

```
type httpd, domain;
type httpd_exec, vendor_file_type, exec_type, file_type;
type httpd_data_file, file_type, vendor_file_type;
init_daemon_domain(httpd)
```

And you will need to add to `device/sirius/marvin/sepolicy/file_contexts`

these lines:

```
/vendor/bin/busybox u:object_r:httpd_exec:s0
/vendor/etc/html(/.*)? u:object_r:httpd_data_file:s0
```

Once you have built the images and rebooted the target, you should find that the http daemon starts:

```
# dmesg | grep httpd
 init: Parsing file /vendor/etc/init/httpd.rc...
 init: starting service 'httpd'...
```

... and you can read a web page from it

But, there are many avc messages from SELinux complaining about httpd.

The simple solution is to use **audit2allow** to convert the avc messages into allow rules:

```
$ adb pull /sys/fs/selinux/policy
$ adb logcat -b all -d | audit2allow -p policy
#============= httpd ==============
allow httpd httpd_data_file:dir search;
allow httpd httpd_data_file:file { getattr open read };
allow httpd node:tcp_socket node_bind;
allow httpd port:tcp_socket name_bind;
allow httpd self:capability net_bind_service;
allow httpd self:tcp_socket { accept bind create listen read setopt shutdown write };
```

You can copy/paste the httpd section into `sepolicy/httpd.te`

11 Device configuration, part 2

11.1 Overlay

The initial list of overlays is:

```
$ get_build_var DEVICE_PACKAGE_OVERLAYS
device/google/cuttlefish/shared/overlay device/google/cuttlefish/shared/phone/overlay
```

To create the overlay directory

```
$ croot
$ cd device/sirius/marvin
$ mkdir overlay
```

Then edit device.mk and add

```
DEVICE_PACKAGE_OVERLAYS += $(LOCAL_PATH)/overlay
```

Create the path to the file you want to overlay

```
$ mkdir -p overlay/frameworks/base/packages/SettingsProvider/res/values
$ cd overlay/frameworks/base/packages/SettingsProvider/res/values
```

Then create the file default.xml and add

```
<resources>
    <!-- Disable the lockscreen -->
    <bool name="def_lockscreen_disabled">true</bool>
</resources>
```


12 The Android Framework

12.1 Looking at services

The statusbar service is implemented by IStatusBarService, as you can see from this:

```
# service list | grep statusbar
49      statusbar: [com.android.internal.statusbar.IStatusBarService]
```

You can find the AIDL code using `godir IStatusBarService`, which takes you to `frameworks/base/core/java/com/android/internal/statusbar`

The first two interfaces are:

```
interface IStatusBarService
{
    void expandNotificationsPanel();
    void collapsePanels();
[...]
```

So, calling statusbar interface 1 displays the notification screen, and interface 2 hides it again.

12.2 Create a binder interface

No solution required

12.3 Implement the service

To include simple service and manager in the target images, add this to your `device.mk`

```
PRODUCT_PACKAGES += \
    simple-service \
    com.example.simplemanager
```

12.4 Testing

After booting, you can see that the Simple service app is running, with UID system:

```
# ps -A | grep simple
system    672   104   502660 26232         0 b6d06d94 S com.example.simpleservice
```

Also, you can see that the SimpleManager appears as a Binder service:

```
# service list | grep simple
7       simpleservice: [com.example.simplemanager.ISimpleManager]
```

You can call the two interfaces using service call. Interface 1 adds two 32-bit integers:

```
# service call simpleservice 1 i32 3 i32 6
Result: Parcel(00000000 00000009   '........')
```


And, the answer is 9. Note that although the command-line integers are in decimal, the contents of the returned parcel are in hex, so adding 64 and 128 looks like this:

```
# service call simpleservice 1 i32 64 i32 128
Result: Parcel(00000000 000000c0   '........')
```

... because in hex, 0x40 + 0x80 = 0xc0

Likewise, you can test that the second interface echos a string

```
# service call simpleservice 2 s16 "Hello world"
Result: Parcel(
  0x00000000: 00000000 0000000b 00650048 006c006c '........H.e.l.l.'
  0x00000010: 0020006f 006f0077 006c0072 00000064 'o. .w.o.r.l.d...')
```

Note that the 8-bit ASCII string is converted to 16-bit Unicode

12.5 (Optional) Extend the service

Add function `getPidTid` to `simple-manager/com/example/simplemanager/ISimpleManager.aidl`

```
interface ISimpleManager {
    int addInts(int a, int b);
    String echoString(String s);
    String getPidTid();
 }
```

Add this to `simple-manager/com/example/simplemanager/SimpleManager.java`

```
public String getPidTid() {
        try {
                return this.service.getPidTid();
        } catch (RemoteException e) {
                throw new RuntimeException("getPidTid", e);
        }
}
```

Add this to `simple-service/src/com/example/simpleservice/ISimpleServiceImpl.java`

```
import android.os.Process;

[...]

        public String getPidTid() {
                Log.d(LOGTAG, "getPidTid");
                return(Integer.toString(android.os.Process.myPid()) +
                        ", " +
                        Integer.toString(android.os.Process.myTid()));
        }
```

12.6 Worked solution

There are worked solutions in `$HOME/android/solutions`:


```
$ croot
$ cp -a $HOME/android/solutions/framework/simple-service vendor/example
$ cp -a $HOME/android/solutions/framework/simple-manager vendor/example
$ cp -a $HOME/android/solutions/selinux/simpleservice/sepolicy/* device/sirius/marvin/sepolicy
$ cp -a $HOME/android/solutions/selinux/simpleservice/sepolicy-private device/sirius/marvin
```

Then, edit `device/sirius/marvin/device.mk` and add this to the end

```
PRODUCT_PACKAGES += com.example.simplemanager simple-service
```

And, edit `device/sirius/marvin/BoardConfig.mk` and add this line:

```
BOARD_PLAT_PRIVATE_SEPOLICY_DIR += device/sirius/marvin/sepolicy-private
```

Then build

```
$ m
```


13 Android applications and activities

13.1 Worked solution for Platform libraries - simple manager

```
$ croot
$ cp -a $HOME/android/solutions/applications/simple-manager-app vendor/example
```

Then, edit `device/sirius/marvin/device.mk` and add this to the end

```
PRODUCT_PACKAGES += simple-manager-app
```

13.2 Step by step ...

13.3 Applications started at boot time

Find persistent applications:

```
$ adb shell dumpsys package packages > packages.txt
```

Looking through the file for applications with the PERSISTENT flag, you should find these

com.android.networkstack	network stack
android	the framework (not an app)
com.android.ons	Opportunistic Network Service
com.android.se	Secure element
com.android.cellbroadcastservice	Receives network wide broadcasts
com.android.service.ims	IP multimedia and voice service
com.android.networkstack.tethering	Tethering
com.android.emulator.multidisplay	
com.android.phone	The phone app
com.example.simpleservice	
com.android.systemui	System UI - notification and navigation bars

13.4 Build the sample application

The apk file is installed into `/system_ext/app`:

```
ls /system_ext/app/simple-manager-app/
oat  simple-manager-app.apk
```

The local data store is in `/data/user/0/com.example.simplemanagerapp`

```
# ls -l /data/user/0/com.example.simplemanagerapp
drwxrws--x 2 u0_a76 u0_a76_cache 3452 2022-02-14 10:24 cache
drwxrws--x 2 u0_a76 u0_a76_cache 3452 2022-02-14 10:24 code_cache
drwxrwx--x 2 u0_a76 u0_a76       3452 2022-02-16 12:25 files

# ls -l /data/user/0/com.example.simplemanagerapp/files/
-rw-rw---- 1 u0_a76 u0_a76 13 2022-02-16 12:25 myfile.txt
```


This directory is the **internal storage** area for the app. Directory `cache` is for temporary storage

14 Permissions and users

There is a worked solution to the first three exercises in `android/solutions/permissions/simple-service`:

```
$ croot
$ cp -r $HOME/android/solutions/permissions-and-users/simple-service/* \
vendor/example/simple-service
$ cp -r $HOME/android/solutions/permissions-and-users/simple-manager-app/* \
vendor/example/simple-manager-app
$ m
```

14.1 Permissions

Add this to `simple-service/AndroidManifest.xml`

```
<manifest ...>

    <permission android:name="com.example.simpleservice.SIMPLE"
        android:protectionLevel="dangerous" />

  <application ...
```

When you install it on the target, you can see the new permission:

```
# dumpsys package perm | grep simple
  Permission [com.example.simpleservice.SIMPLE] (c41f30b):
    sourcePackage=com.example.simpleservice
    perm=Permission{5a405e8 com.example.simpleservice.SIMPLE}
    packageSetting=PackageSetting{2142230 com.example.simpleservice/1000}
```

14.2 Check permissions

Here is an example of the addInts function with a check for the SIMPLE permission:

```
import android.content.pm.PackageManager; // for PackageManager.PERMISSION_GRANTED
[...]

public int addInts(int a, int b) {
        Log.d(LOGTAG, "addInts");

        int uid = getCallingUid();
        int pid = getCallingPid();
        Log.d(LOGTAG, "Message from PID " + pid + " UID " + uid);

        int perm = mContext.checkPermission("com.example.simpleservice.SIMPLE",
                                pid, uid);
        if (perm == PackageManager.PERMISSION_GRANTED)
                Log.d(LOGTAG, "Granted");
        else
                Log.d(LOGTAG, "Denied");

        return a + b;
}
```


This is the log when calling it from a root shell

```
05-24 08:18:44.231   736   775 D ISimpleServiceImpl: addInts
05-24 08:18:44.231   736   775 D ISimpleServiceImpl: Message from PID 1027 UID 0
05-24 08:18:44.232   736   775 D ISimpleServiceImpl: Granted
```

This is the log when calling it from a "shell" shell

```
05-24 08:24:10.209   736   775 D ISimpleServiceImpl: addInts
05-24 08:24:10.210   736   775 D ISimpleServiceImpl: Message from PID 1045 UID 2000
05-24 08:24:10.211   736   775 D ISimpleServiceImpl: Denied
```

Here is an example of the echoString function which enforces the SIMPLE permission:

```
public String echoString(String s) {
        Log.d(LOGTAG, "echoString");
        mContext.enforceCallingPermission("com.example.simpleservice.SIMPLE",
                "echoString permission not granted");
        return s;
}
```

This is what happens when calling it from a root shell

```
# service call simpleservice 2 s16 "Hello world"
Result: Parcel(
  0x00000000: 00000000 0000000b 00650048 006c006c '........H.e.l.l.'
  0x00000010: 0020006f 006f0077 006c0072 00000064 'o. .w.o.r.l.d...')
```

This is what happens when calling it from a "shell" shell

```
$ service call simpleservice 2 s16 "Hello world"
Result: Parcel(
  0x00000000: ffffffff 0000005b 00630065 006f0068 '....[...e.c.h.o.'
  0x00000010: 00740053 00690072 0067006e 00700020 'S.t.r.i.n.g. .p.'
  0x00000020: 00720065 0069006d 00730073 006f0069 'e.r.m.i.s.s.i.o.'
  0x00000030: 0020006e 006f006e 00200074 00720067 'n. .n.o.t. .g.r.'
  0x00000040: 006e0061 00650074 003a0064 00750020 'a.n.t.e.d.:. .u.'
  0x00000050: 00640069 00320020 00300030 00200030 'i.d. .2.0.0.0. .'
  0x00000060: 006f0064 00730065 006e0020 0074006f 'd.o.e.s. .n.o.t.'
  0x00000070: 00680020 00760061 00200065 006f0063 ' .h.a.v.e. .c.o.'
  0x00000080: 002e006d 00780065 006d0061 006c0070 'm...e.x.a.m.p.l.'
  0x00000090: 002e0065 00690073 0070006d 0065006c 'e...s.i.m.p.l.e.'
  0x000000a0: 00650073 00760072 00630069 002e0065 's.e.r.v.i.c.e...'
  0x000000b0: 00490053 0050004d 0045004c 0000002e 'S.I.M.P.L.E.....')
```

14.3 Requesting permissions

Initially, simplemanager-app is killed with an exception when is tries to call a function in simple-server:

```
05-24 08:44:30.966   736   779 D ISimpleServiceImpl: echoString
05-24 08:44:30.970  1076  1076 D AndroidRuntime: Shutting down VM
--------- beginning of crash
05-24 08:44:30.972  1076  1076 E AndroidRuntime: FATAL EXCEPTION: main
05-24 08:44:30.972  1076  1076 E AndroidRuntime: Process: com.example.simplemanagerapp, PID:
1076
05-24 08:44:30.972  1076  1076 E AndroidRuntime: java.lang.IllegalStateException: Could not
```



```
execute method for android:onClick
[...]
```

Add a uses-permission tag to the manifest for simple-manager-app:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.simplemanagerapp">
    <uses-permission android:name="com.example.simpleservice.SIMPLE" />
```

Add code to request the permission. The solution is for the echoString function only: you would need to do the same for addInts.

```
import android.content.pm.PackageManager;  // For PERMISSION_GRANTED

public class SimpleManagerActivity extends Activity
{
    private static final int MY_PERMISSIONS_REQUEST_CODE = 1;
[...]

    public void onButtonEchostring(View view)
    {
        if (checkSelfPermission("com.example.simpleservice.SIMPLE")
                != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{"com.example.simpleservice.SIMPLE"},
                MY_PERMISSIONS_REQUEST_CODE);
            }
        else {
            tv2.setText(mSimpleManager.echoString("Hello"));
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
            String permissions[], int[] grantResults)
    {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_CODE: {
                if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    // permission was granted

                } else {

                    // permission denied,
                    // Disable the functionality that depends on this permission.
                }
                return;
            }
        }
    }
}
```

You can see that the permission has been requested by the app:

```
$ adb shell dumpsys package > packages2.txt
```

Then look through packages2.txt for this sequence:


```
[...]
  Package [com.example.simplemanagerapp] (xxxxxxx):
    userId=xxxxx
    pkg=Package{xxxxxx com.example.simplemanagerapp}
[...]
      runtime permissions:
        com.example.simpleservice.SIMPLE: granted=true, flags=[ USER_SET]
[...]
```

In this case the user has granted the permission

14.4 (Optional) User IDs

No solution given for this exercise

14.5 (Optional) Group IDs

No solution given for this exercise

15 Hardware Abstraction Layers

15.1 Listing HALs

No solution given for this exercise

15.2 Implementing the Lights HAL

There is a worked solution of liblights in `$HOME/android/solutions/light-hal`

16 AIDL for HAL

16.1 AIDL Lights HAL

16.2 Worked solution

```
$ croot
$ cp -a $HOME/android/solutions/light-hal-aidl vendor/example
```

Edit `device/sirius/marvin/device.mk` and add

```
PRODUCT_PACKAGES += android.hardware.lights-service.marvin
```

Edit `device/sirius/marvin/sepolicy/file_contexts` and add:

```
/vendor/bin/hw/android.hardware.lights-service.marvin u:object_r:hal_light_default_exec:s0
```

Then build marvin

```
$ m
```

16.3 (Optional) Build and run the VTS for lights

No solution for this exercise

17 Calling native code: JNI

17.1 Write the Java code

vendor/example/simple-service/src/com/example/simpleservice/ISimpleServiceImpl.java

```
package com.example.simpleservice;

import android.content.Context;
import android.util.Log;
import com.example.simplemanager.ISimpleManager;

class ISimpleServiceImpl extends ISimpleManager.Stub {
        private static final String LOGTAG = "ISimpleServiceImpl";

        public native int addIntsNative(int a, int b);
        public native String echoStringNative(String s);

        static {
                System.loadLibrary("simple-jni");
        }

        public int addInts(int a, int b) {
                Log.d(LOGTAG, "addInts");
                return addIntsNative(a, b);
        }

        public String echoString(String s) {
                Log.d(LOGTAG, "echoString");
                return echoStringNative(s);
        }
}
```

17.2 Write the C code

vendor/example/simple-jni/simple-jni.c

```
#define LOG_TAG "simplejni"
#include <cutils/log.h>
#include <stdint.h>
#include <string.h>
#include <jni.h>

JNIEXPORT jint JNICALL
Java_com_example_simpleservice_ISimpleServiceImpl_addIntsNative(JNIEnv* env,
                jobject thiz, jint a, jint b)
{
        ALOGI("%s\n", __func__);
        // Add one so that we can see that the calculation was done (wrongly) here
        return a + b + 1;
}

JNIEXPORT jstring JNICALL
Java_com_example_simpleservice_ISimpleServiceImpl_echoStringNative(JNIEnv* env,
                jobject thiz, jstring s)
```



```
{
        ALOGI("%s\n", __func__);
        // Return a constant string to identify this function
        return (*env)->NewStringUTF(env, "simple-jni");
}
```

Android.bp

```
cc_library_shared {
    name: "libsimple-jni",
    srcs: ["simple-jni.c"],
    shared_libs: ["liblog"],
}
```

17.3 Test

No solution given for this exercise

18 The Android graphics stack

18.1 Window manager

Default display size and density:

```
$ wm size
Physical size: 720x1280
$ wm density
Physical density: 320
```

18.2 SurfaceFlinger

The command "dumpsys SurfaceFlinger" dumps the internal state of Surface Flinger. This part tells you about the OpenGL and EGL implementation:

```
[...]
SurfaceFlinger global state:

 ------------RE-----------------
EGL implementation : 1.5 Android META-EGL
[...]
GLES: Google Inc. (Google), ANGLE (Google, Vulkan 1.2.0 (SwiftShader Device (LLVM 10.0.0)
(0x0000C0DE)), SwiftShader driver-5.0.0), OpenGL ES 3.1.0 (ANGLE 2.1.17365 git hash: acdff2aa8be4)
[...]
```


19 Android Automotive

19.1 Running Android Automotive

Features

```
# pm list features
[...]
feature:android.hardware.type.automotive
```


20 The vehicle HAL

20.1 The vehicle HAL

Check that the HAL is running:

```
marvincar:/ # lshal
 All HIDL binderized services (registered with hwservicemanager)
VINTF R Interface                                                  Thread Server Clients
[...]
DM,FC Y android.hardware.automotive.vehicle@2.0::IVehicle/default  0/2    380    872 173 207 189
[...]
```

20.2 Vendor properties

No solution for this exercise

20.3 Testing

`MAKE_TEA` is 0x21402001

`TEA_STATUS` is 0x21402002

`KETTLE_WATER_TEMPERATURE` is 0x21602003

20.4 Testing using SocketComm

No solution for this exercise

20.5 (Optional) vehiclehaltest

The code shown below will read the two vendor properties:

```
        VehiclePropValue mVendor1;
        mVendor1.prop =
            static_cast < int32_t > (VehicleProperty::VENDOR_STATIC);
        invokeGet(&mVendor1);
        printf("Vendor static: %d\n", mVendor1.value.int32Values[0]);

        VehiclePropValue mVendor2;
        mVendor2.prop =
            static_cast < int32_t > (VehicleProperty::VENDOR_VAR);
        invokeGet(&mVendor2);
        printf("Vendor var: %f\n", mVendor2.value.floatValues[0]);
```

There is a worked solution in `$HOME/android/solutions/auto-hal/vehiclehaltest`

20.6 (Optional) vendor properties in types.hal

No solution for this exercise

20.7 (Optional) Read the properties

No solution for this exercise

20.8 (Optional) Subscribe to a vehicle event

No solution for this exercise

20.9 (Optional) Generating fake changes to a property

No solution for this exercise

21 The Car Service

21.1 The car system service

No solution give for this exercise

21.2 A car application

No solution give for this exercise

21.3 Reading vendor properties from an application

There is a worked solution in `$HOME/android/solutions/auto-car-api/examplecar`

