SSE3052: Embedded Systems Practice

Jinkyu Jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Agenda

1. Attach 7 segment display to (virtual) device.
2. Write a device driver.
3. Write a system call & user program to manipulate the display.
Attach 7 Seg. Display
Attach 7 Seg. Display

• Downgrade device & Android
  – Device: Nexus 5X => Nexus S (WVGA)
  – Android: Oreo (API 26) => Nougat (API 24)

• Download 3_26.zip from
  http://csl.skku.edu/SSE3052S19/Resources
### Choose a device definition

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Size</th>
<th>Resolution</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>5.4&quot; FWVGA</td>
<td>5.4&quot;</td>
<td>480x854</td>
<td>mdpi</td>
</tr>
<tr>
<td>Wear</td>
<td>5.1&quot; WVGA</td>
<td>5.1&quot;</td>
<td>480x800</td>
<td>mdpi</td>
</tr>
<tr>
<td>Phone</td>
<td>4.7&quot; WXGA</td>
<td>4.7&quot;</td>
<td>720x1280</td>
<td>xhdpi</td>
</tr>
<tr>
<td></td>
<td>4.65&quot; 720p (Galaxy ...)</td>
<td>4.65&quot;</td>
<td>720x1280</td>
<td>xhdpi</td>
</tr>
<tr>
<td></td>
<td>4&quot; WVGA (Nexus S)</td>
<td>4.0&quot;</td>
<td>480x800</td>
<td>hdpi</td>
</tr>
<tr>
<td></td>
<td>3.7“ WVGA (Nexus ...)</td>
<td>3.4&quot;</td>
<td>480x800</td>
<td>hdpi</td>
</tr>
<tr>
<td></td>
<td>3.7” FWVGA slider</td>
<td>3.7&quot;</td>
<td>480x854</td>
<td>ldpi</td>
</tr>
<tr>
<td></td>
<td>3.4” WQVGA</td>
<td>3.4&quot;</td>
<td>240x432</td>
<td>ldpi</td>
</tr>
</tbody>
</table>

**4" WVGA (Nexus S)**

- **Size:** normal
- **Ratio:** long
- **Density:** hdpi
<table>
<thead>
<tr>
<th>Release Name</th>
<th>API Level</th>
<th>ABI</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oreo</td>
<td>26</td>
<td>x86_64</td>
<td>Android 8.0 (Google APIs)</td>
</tr>
<tr>
<td>Nougat Download</td>
<td>25</td>
<td>x86_64</td>
<td>Android 7.1.1 (Google APIs)</td>
</tr>
<tr>
<td>Nougat</td>
<td>24</td>
<td>x86_64</td>
<td>Android 7.0 (Google APIs)</td>
</tr>
<tr>
<td>Nougat Download</td>
<td>24</td>
<td>x86_64</td>
<td>Android 7.0</td>
</tr>
<tr>
<td>Nougat Download</td>
<td>24</td>
<td>x86_64</td>
<td>Android 7.0</td>
</tr>
<tr>
<td>Marshmallow Download</td>
<td>23</td>
<td>x86_64</td>
<td>Android 6.0 (Google APIs)</td>
</tr>
<tr>
<td>Marshmallow Download</td>
<td>23</td>
<td>x86_64</td>
<td>Android 6.0</td>
</tr>
<tr>
<td>Marshmallow Download</td>
<td>23</td>
<td>x86_64</td>
<td>Android 6.0</td>
</tr>
<tr>
<td>Lollipop Download</td>
<td>22</td>
<td>x86_64</td>
<td>Android 5.1 (Google APIs)</td>
</tr>
<tr>
<td>Lollipop Download</td>
<td>22</td>
<td>x86_64</td>
<td>Android 5.1</td>
</tr>
<tr>
<td>Lollipop Download</td>
<td>22</td>
<td>x86_64</td>
<td>Android 5.1</td>
</tr>
</tbody>
</table>

Questions on API level? See the [API level distribution chart](#).
A collection of images and configuration data that indicates how to populate the window. Each skin can have several “layouts” (e.g. "landscape" and "portrait") corresponding to different orientation / physical configurations of the emulated device.
Attach 7 Seg. Display

• Untar "skin_android-24_WVGA800.tgz".
  – Note: The file contains image files of 7 segment display.

• Untar & place all image files to relevant directory.
  – Ex) ~/Android.Sdk/platforms/android-24/skins WVGA800/
Attach 7 Seg. Display

- Download emulator source code.
  - (We do not use the emulator in SDK.)
  
  ```
  $mkdir ~/bin
  $PATH=~/bin:$PATH
  $curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  $chmod a+x ~/bin/repo
  $mkdir emu-2.2-release
  $cd !$
  $repo init -u https://android.googlesource.com/platform/manifest
  -b emu-2.2-release
  $repo sync
  ```
Attach 7 Seg. Display

- Place “qemu_segment.patch” under emu-2.2-release/external/qemu
- Patch.
  ```bash
  $ patch -p1 < qemu_segment.patch
  ```
- Build.
  ```bash
  $./android-rebuild.sh```
Attach 7 Seg. Display

- Run new emulator.
  
  $\text{export ANDROID_SDK_ROOT}=/path/to/android-sdk
  
  $/path/to/qemu/objs/emulator -avd [avd-name]$
Device Driver

• Downgrade kernel version

$ cd ~/goldfish
$ git checkout -b android-goldfish-n-3.10-n-dev
origin/android-goldfish-3.10-n-dev
Device Driver

(You won't need to write a device driver.)

• Place “goldfish_kernel.patch” under goldfish.

• Patch
  
  ```bash
  $patch -p1 < goldfish_kernel.patch
  ```

• Build
  
  ```bash
  $export ARCH=x86_64
  $export CROSS_COMPILE=~/x86_64-linux-android-4.9/bin/x86_64-linux-android-
  $make x86_64_emu_defconfig
  $make menuconfig
  =>'Device drivers/Misc devices/Android Goldifsh 7 segment
  $make -j4
  $cp arch/x86/boot/bzImage ~/Android.Sdk/system-images/android-24/google_apis/x86_64/kernel-qemu
  ```
goldfish/drivers/misc/goldfish_segment.c

DEVICE DRIVER
goldfish_segment.c

static int goldfish_segment_probe(struct platform_device *pdev) {
  ...
}

static int goldfish_segment_remove(struct platform_device *pdev) {
  ...
}

static const struct of_device_id goldfish_segment_of_match[] = {
  ...
};
MODULE_DEVICE_TABLE(of, goldfish_segment_of_match);

static const struct ACPI_device_id goldfish_segment_acpi_match[] = {
  ...
};
MODULE_DEVICE_TABLE(acpi, goldfish_segment_acpi_match);
static struct platform_driver goldfish_segment_device = {
    .probe = goldfish_segment_probe,
    .remove = goldfish_segment_remove,
    .driver = {
        .name = "goldfish_segment",
        .of_match_table = goldfish_segment_of_match,
        .acpi_match_table = ACPI_PTR(goldfish_segment_acpi_match),
    }
};

module_platform_driver(goldfish_segment_device);
Platform Driver

<linux/platform_device.h>

```c
struct platform_driver {
    int (*probe)(struct platform_device *);
    int (*remove)(struct platform_device *);
    void (*shutdown)(struct platform_device *);
    int (*suspend)(struct platform_device *, pm_message_t state);
    int (*resume)(struct platform_device *);
    struct device_driver driver;
    const struct platform_device_id *id_table;
};
```

... 

```c
#define module_platform_driver(__platform_driver)  
    module_driver(__platform_driver,  
                  platform_driver_register,  
                  platform_driver_unregister)
```
#goldfish_segment.c

```c
#define GOLDFISH_SEGMENT_READ(data, addr) \  (readl(data->reg_base + addr))
#define GOLDFISH_SEGMENT_WRITE(data, addr, x) \  (writel(x, data->reg_base + addr))
...
GOLDFISH_SEGMENT_WRITE(data, SEGMENT0, 0x24); // "1"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT1, 0x5d); // "2"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT2, 0x6d); // "3"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT3, 0x2e); // "4"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT4, 0x6b); // "5"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT5, 0x7a); // "6"
GOLDFISH_SEGMENT_WRITE(data, SEGMENT6, 0x27); // "7"
```
I/O Memory

- `readl()`? `writel()`?

  - Slide #272 "I/O Memory and Ports"
7 Segment Display

- 0x24? 0x5d? 0x6d?...

<table>
<thead>
<tr>
<th>A</th>
<th>&quot;1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0001b</td>
<td>0010 0100b = 0x24</td>
</tr>
<tr>
<td>F</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>0000 0010b</td>
<td>0101 1101b = 0x5d</td>
</tr>
<tr>
<td>B</td>
<td>&quot;3&quot;</td>
</tr>
<tr>
<td>0000 0100b</td>
<td>0110 1101b = 0x6d</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>0000 1000b</td>
<td>...</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>0001 0000b</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>0010 0000b</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>0100 0000b</td>
<td></td>
</tr>
</tbody>
</table>

SSE3052: Embedded Systems Practice, Spring 2019, Jinkyu Jeong (jinkyu@skku.edu)
Exercise

1. Write a **system call** that takes an integer as a parameter and display the integer to 7 segment display.
   – If an integer is larger than 9999999, then display "0000000".

2. Write a program that invokes the implemented system call with parameter input from a user.
   – int main(int arcs, int *argv[])
   – Ex) ./a.out 2019326
   – Ex) ./a.out 1000

• Do you remember,
  – how to add new system call?
  – how to compile new user program?
  – how to move executable binary?
  – Please check the last lecture...
Lab Report

• Format: yourstudentID_lab1.pdf
  E-mail to: sunghwan.kim@csi.skku.edu
  Deadline: 3/27 (Wed.) 23:59