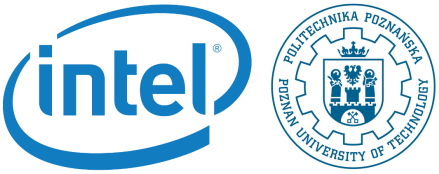

Compiling Linux kernel

Rafal Kapela

During this labaroatory you will configure and compile Linux kernel for Galileo platform

Note, that in order to follow instructions in this lab session you need to have prepared the following configuration:

- PC computer with Linux Kubuntu installed on it in the lab.
- Galileo2 development board.
- Arduino mezanine board.
- I2C display extension.
- Photoresistor extension.
- Potentiometer extension.
- An SD card reader for PC computer.
- Power supply.
- USB UART cable.
- An SD card prepared on the last laboratory session.

The setup is shown in the following picture.

1 Installing

1. In order to run the UPM dependend applications, you need to install the UPM libraries. Prepare system for compilation of the library. NOTE that you'll need to fix missing repositories:

```
apt-get update --fix-missing
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys <REF-KEY>
...
apt-get install build-essential cmake git pkg-config unzip swig python-dev vim
apt-get update
```

The last `apt-getupdate` should not give you any warning or errors.

2. Download and build mraa library. In $HOME directory:

```
wget-O mraa.zip https://github.com/intel-iot-devkit/mraa/archive/master.zip
unzip mraa.zip
mkdir mraa-master/build && cd $_
cmake .. -DBUILDSWIGNODE=OFF
make
make install
```

3. Download the upm library:

   `wget -O upm.zip https://github.com/intel-iot-devkit/upm/archive/master.zip`

4. Extract library and build it (it'll take long time! - almost two hours!!! Ask lecturer about the `screen` command):

```
unzip upm.zip
mkdir upm-master/build && cd $_
cmake -DBUILDSWIGNODE=OFF -DBUILDSWIGPYTHON=OFF -DBUILDEXAMPLES=ON ..
make
make install
```

5. Now check if the LCD control application is fully functional.

2

1. Run the resizepart command of parted to expand the root partition (actually the 2nd partition on the SD card). It will ask several questions, but that is expected. Do Not Panic.

   `parted /dev/mmcblk0 resizepart 2`

   First will be two scary looking messages complaining about the GPT table not being at the end of the device, and also asking if you want to use the entire device. In both cases, just allow parted to fix the problem. It actually knows what it is doing.

```
Error: The backup GPT table is not at the end of the disk, as it should be.
This might mean that another operating system believes the disk is smaller.
Fix, by moving the backup to the end (and removing the old backup)?
```

```
parted: invalid token: 2
Fix/Ignore/Cancel? f

Warning: Not all of the space available to /dev/mmcblk0 appears to be used, you
can fix the GPT to use all of the space (an extra 5986304 blocks) or continue
with the current setting?
Fix/Ignore? f
```

Now it is going to propt you for the details of the resize operation. The OS is on partition 2, and you do want to chane it even though are currently using it. You are just growing it, so this is safe.

```
Partition number? 2

Warning: Partition /dev/mmcblk0p2 is being used. Are you sure you want to
continue?
Yes/No? yes

Now, tell it the size of your SD card. SD cards are always slightly
    smaller their advertised size, so provide a value about 10% smaller
    than the size of the card. The value 2020MB used below is just
    an example. Use whatever number is right for your SD card.

End?  [944MB]? 2020MB

Error: Partition(s) 2 on /dev/mmcblk0 have been written, but we have been
    unable to inform the kernel of the change, probably because it/they
    are in use.  As a result, the old partition(s) will remain in use.
    You should reboot now before making further changes.
Ignore/Cancel? I
```

2. You will need to reboot now so the kernel will see the new partion table.
3. First, make sure the filesystem is really ext3.

   ```
   tune2fs -j /dev/mmcblk0p2
   ```

4. If a journaling inode was created, then you'll need to reboot once more for this step to take effect.
5. The final step is to resize the filesystem

   ```
   resize2fs /dev/mmcblk0p2
   ```

6. Your system should now have a larger filesystem.

3 Preparing Ubuntu kernel for Galileo2

1. Download and patch the Linux kernel for Galileo2 board.

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
cd linux-stable/
git reset --hard 531ec28f9f26f78797124b9efcf2138b89794a1e
git am <patch-directory>/00*
```

2. Use `.config` file which is provided with the patches! Simply comy it to the kernel directory.

3. Compile and install kernel drivers.

```
make ARCH=i386 -j 8
mkdir ../install_dir
make ARCH=i386 INSTALL_MOD_PATH=../install_dir modules_install
```

4. On the Galilio2, prepare a copy of the vmlinuz file in `\boot` directory.

5. Also copy the drivers installed in `../install_dir` to `/lib/modules/` directory on the Galileo2 board.

6. Reboot the system and check if all the functionality is still there. The I2C shouldn't work wince there's not ready driver yet.