FROM EMBEDDED LINUX BUILD SYSTEMS TO EDGE COMPUTING
CEZARY DYNAK
2018-10-24
CEZARY DYNAK

ABOUT ME
SOME TIME AGO...

- Wrocław University of Science and Technology
- Faculty of Electronics / W4
- Department of Cybernetics and Robotics / K7
- Laboratory 07/C-3 (diablo/sequoia)
- Automatyka i Robotyka / Embedded Robotics
CURRENTLY

Node.js developer
IoT back-end (cloud)

cdynak@spyro-soft.com
• github.com/cdynak
• gitlab.com/cdynak
• linkedin.com/in/cdynak
• last.fm/user/cezdyn
PLAN

CDYNAK.GITLAB.IO/LINUX2018

- About my IoT project
- Embedded Linux build systems
- Building and deploying - embedded
- Building and deploying - containers
- Edge computing
- Conclustion
ABOUT MY IOT PROJECT
PROJECT HISTORY

- requirements: multi-user, multi-platform, multi-protocol, but one instance per customer
- 1st version: ANSI C + LAMP + JavaScript
- [back-end] rewritten in Node.js
- various deployments (main topic of this presentation)
- going to OpenSource
- github.com/newterm/szarp (C++/wxWidgets, MIT)
- github.com/RapidScada/scada (C#, Apache)
- github.com/SCADA-LTS/Scada-LTS (Java, GPLv2)
- github.com/trombastic/PyScada (Python, GPLv3)
- oscada.org/websvn/listing.php?repname=OpenSCADA (C++, GPLv2)
EMBEDDED LINUX BUILD SYSTEMS
FOR A LOT OF PEOPLE IT ALL STARTS WITH...

- https://raspberrypi.org
- https://beagleboard.org/black
- http://wiki.litesom.grinn-global.com
- https://wandboard.org
- http://pandaboard.org
<table>
<thead>
<tr>
<th>name</th>
<th>Raspberry Pi 1</th>
<th>Raspberry Pi 2</th>
<th>BeagleBone Black</th>
<th>PandaBoard</th>
<th>Wandboard Quad</th>
<th>Asus Eee PC 1215n</th>
</tr>
</thead>
<tbody>
<tr>
<td>release date</td>
<td>April 2012</td>
<td>February 2015</td>
<td>April 2013</td>
<td>October 2010</td>
<td>February 2013</td>
<td>August 2010</td>
</tr>
<tr>
<td>target price</td>
<td>$35</td>
<td>$35</td>
<td>$45</td>
<td>$174</td>
<td>$129</td>
<td>$499</td>
</tr>
<tr>
<td>word size</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit/64-bit</td>
</tr>
<tr>
<td>SoC</td>
<td>Broadcom BCM2835</td>
<td>Broadcom BCM2835</td>
<td>Texas Instruments AM3358/9</td>
<td>Texas Instruments OMAP4430</td>
<td>Freescale i.MX6 Quad</td>
<td>Intel Atom</td>
</tr>
<tr>
<td>architecture</td>
<td>ARM Cortex-A7</td>
<td>ARM Cortex-A8</td>
<td>ARM Cortex-A8</td>
<td>ARM Cortex-A9</td>
<td>ARM Cortex-A9 x86</td>
<td></td>
</tr>
<tr>
<td>CPU frequency</td>
<td>700 MHz</td>
<td>1000 MHz</td>
<td>1000 MHz</td>
<td>1000 MHz</td>
<td>1000 MHz</td>
<td>1800 MHz</td>
</tr>
<tr>
<td>RAM size</td>
<td>512 GB DDR3</td>
<td>1 GB</td>
<td>512 MB DDR3</td>
<td>1 GB</td>
<td>2GB DDR3</td>
<td>2GB DDR3</td>
</tr>
<tr>
<td>Power source</td>
<td>5 V (Micro US- B/GPIO)</td>
<td>5 V (Micro US- B/GPIO)</td>
<td>Mini USB / 5 V jack</td>
<td>5Y</td>
<td>5V</td>
<td>19V</td>
</tr>
<tr>
<td>USB</td>
<td>2 (via the on- board 5-port USB hub)</td>
<td>4 (via the on- board 5-port USB hub)</td>
<td>USB 2.0</td>
<td>two USB host ports and one USB On- The-Go</td>
<td>USB 3.0</td>
<td>USB 2.0 + USB 3.0</td>
</tr>
<tr>
<td>Network</td>
<td>10/100 Mbit/s Ethernet on the USB hub</td>
<td>10/100 Mbit/s Ethernet on the USB hub</td>
<td>Ethernet Fast Ethernet (MII based)</td>
<td>10/100 Ethernet on USB hub</td>
<td>GbE</td>
<td>10/100 Ethernet (MII based)</td>
</tr>
<tr>
<td>storage</td>
<td>microSDHC slot</td>
<td>microSDHC slot</td>
<td>4GB eMMC / mi- croSDHC slot</td>
<td>SDHC slot</td>
<td>microSDHC</td>
<td>SATA (default 320 GB HDD)</td>
</tr>
</tbody>
</table>

Table 2.1: Development boards comparison
DEVELOPMENT BOARDS...

BUT WHY?

- for prototyping
- for promotion
- for learning
- for fun
HOW TO MAKE IT WORK?

- Download dedicated OS image from web
- or... build own image!
wget https://downloads.raspberrypi.org/raspbian_latest

timed pv sdc1.img | dd of=/dev/sdb bs=4M oflag=dsync
SO...

WHAT ARE THE EMBEDDED LINUX BUILD SYSTEMS?!
**Embedded Linux build system** is a set of software development tools, that create Linux distribution with the use of cross-compiler and produce complete operating system image, to be deployed on an embedded device.
GENERALIZED OS BUILDERS STRUCTURE

- Source code
- Host OS requirements
- Cross-compilation toolchain
- Target OS configuration
- Produced output
TESTING AND COMPARING

- Representative set of embedded devices
- Available open source distribution builders
<table>
<thead>
<tr>
<th>name</th>
<th>Buildroot</th>
<th>OpenWrt</th>
<th>LTB</th>
<th>PTXdist</th>
<th>Yocto Project</th>
<th>CLFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>last stable release date</td>
<td>2017.11</td>
<td>2017.01</td>
<td>2013.02</td>
<td>2018.01</td>
<td>2017.10</td>
<td>2014.10</td>
</tr>
</tbody>
</table>

Table 3.1: Embedded Linux build systems overview
BUT THERE ARE OTHERS...

- linux from scratch
- debootstrap
- wind river
REASONS BEHIND USING VM

- stable internet connection
- repeatable
sudo apt install python-openstackclient

source ...-openrc.sh

openstack server create --flavor 'c2-30' --image 'Debian 9' \
--network 'Ext-Net' --key-name 'rsa-local' 'osseu2018'

openstack server list

ssh debian@...
sudo apt update
sudo apt install make gcc g++ libncurses-dev unzip git patch python rsync bc bzip2
wget https://buildroot.org/downloads/buildroot-2018.08.1.tar.gz
tar -zxf buildroot-2018.08.1.tar.gz
cd buildroot-2018.08.1/
make raspberrypi_defconfig
make
ls output/images/sdcard.img
sudo apt install make gcc g++ libncurses-dev unzip git gawk file zlib1g-dev
git clone -b v18.06.1 https://github.com/openwrt/openwrt
cd openwrt/
./scripts/feeds update -a
./scripts/feeds install -a
make defconfig
make menuconfig
# Target System (Broadcom BCM27xx)
# Target Profile (Raspberry Pi B/B+/CM/Zero/ZeroW)
time make -j

cp build_dir/target-arm Armstrong1176jzf-s+vfp_musl-1.1.16_eabi/linux-bcm2708_bcm2708/tmp/lede-
sudo apt install make gcc g++ libncurses-dev unzip rpm bison patch tcl zlib1g-dev

wget https://github.com/downloads/midnightyell/RPi-LTIB/raspberrypi-tools-9c3d7b6-1.i386.rpm
sudo mkdir -p /opt/ltib/pkgs/
sudo cp raspberrypi-tools-9c3d7b6-1.i386.rpm /opt/ltib/pkgs/

sudo dpkg --add-architecture i386
sudo apt update
sudo apt install zlib1g:i386 libstdc++:i386

wget http://download.savannah.nongnu.org/releases/ltib/ltib-13-2-1-sv.tar.gz
tar -xzf ltib-13-2-1-sv.tar.gz
cd ltib-13-2-1-sv/

time ./ltib

# Platform choice (Raspberry Pi with BCM2835 SoC)

cp output/images/sdcard.img ~
make
sudo make install
cd ..
wget http://public.pengutronix.de/software/ptxdist/ptxdist-2016.06.0.tar.bz2
tar -xjf ptxdist-2016.06.0.tar.bz2
cd ptxdist-2016.06.0
./configure
make
sudo make install

cd ..
wget https://public.pengutronix.de/oselas/toolchain/OSELAS.Toolchain-2016.06.1.tar.bz2
tar -xjf OSELAS.Toolchain-2016.06.1.tar.bz2
cd OSELAS.Toolchain-2016.06.1/
ptxdist-2016.06.0 select ptxconfigs/arm-1136jfs-linux-gnueabihf_gcc-5.4.0_glibc-2.23_bin
ptxdist-2016.06.0 migrate
time ptxdist-2016.06.0 go

git clone https://git.pengutronix.de/cgit/DistroKit/
cd DistroKit/
ptxdist-2018.01.0 platform configs/platform-rpi/platformconfig
ptxdist-2018.01.0 migrate
time ptxdist-2018.01.0 images
cp platform-rpi/images/hd.img ~
wget http://commondatastorage.googleapis.com/git-repo-downloads/repo
chmod a+x repo
sudo mv repo /usr/local/bin/

sudo apt install make gcc g++ unzip git
sudo apt install gawk diffstat texinfo build-essential chrpath

export MACHINE=raspberrypi

mkdir yoctoproject
cd yoctoproject
repo init -u https://github.com/cdynak/yocto-manifest -m $MACHINE.xml
repo sync

source poky/oe-init-build-env
% MACHINE=wandboard DISTRO=poky source setup-environment build
% vi conf/bblayers.conf
time bitbake core-image-minimal
cp tmp/deploy/images/$MACHINE/*img ~
TESTS

- With respect to the target OS
- With respect to the host OS
- Intel(R) Xeon(R) CPU E3-1270 v6 @ 3.80GHz (8 cores) and 32 GiB RAM
<table>
<thead>
<tr>
<th>name</th>
<th>Raspberry Pi 1</th>
<th>Raspberry Pi 2</th>
<th>BeagleBone Black</th>
<th>PandaBoard Quad</th>
<th>Wandboard Quad</th>
<th>Asus Eee PC</th>
<th>1215n</th>
</tr>
</thead>
<tbody>
<tr>
<td>real time</td>
<td>17m33.619s</td>
<td>20m15.836s</td>
<td>26m12.604s</td>
<td>11m41.240s</td>
<td>11m39.519s</td>
<td>17m40.604s</td>
<td></td>
</tr>
<tr>
<td>user time</td>
<td>73m47.908s</td>
<td>74m30.368s</td>
<td>55m52.408s</td>
<td>49m39.200s</td>
<td>48m33.172s</td>
<td>53m31.832s</td>
<td></td>
</tr>
<tr>
<td>sys time</td>
<td>3m54.620s</td>
<td>3m58.084s</td>
<td>3m13.860s</td>
<td>2m31.172s</td>
<td>2m27.688s</td>
<td>2m30.832s</td>
<td></td>
</tr>
<tr>
<td>buildroot/</td>
<td>5.4G</td>
<td>5.4G</td>
<td>6.0G</td>
<td>4.8G</td>
<td>4.7G</td>
<td>5.9G</td>
<td></td>
</tr>
<tr>
<td>sdcard.img</td>
<td>93M</td>
<td>93M</td>
<td>77M</td>
<td>69M</td>
<td>61M</td>
<td>121M</td>
<td></td>
</tr>
<tr>
<td>boot time</td>
<td>4.926830</td>
<td>5.575466</td>
<td>4.028472</td>
<td>4.926830</td>
<td>3.235903</td>
<td>19.957294</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Buildroot build comparison

<table>
<thead>
<tr>
<th>name</th>
<th>Raspberry Pi 1</th>
<th>Raspberry Pi 2</th>
<th>BeagleBone Black</th>
<th>PandaBoard Quad</th>
<th>Wandboard Quad</th>
<th>Asus Eee PC</th>
<th>1215n</th>
</tr>
</thead>
<tbody>
<tr>
<td>real time</td>
<td>12m2.398s</td>
<td>12m5.585s</td>
<td>13m19.282s</td>
<td>-</td>
<td>3m32.177s</td>
<td>5m42.104s</td>
<td></td>
</tr>
<tr>
<td>user time</td>
<td>62m52.308s</td>
<td>62m24.408s</td>
<td>66m25.504s</td>
<td>-</td>
<td>43m14.808s</td>
<td>42m35.520s</td>
<td></td>
</tr>
<tr>
<td>sys time</td>
<td>3m19.248s</td>
<td>3m16.664s</td>
<td>3m27.880s</td>
<td>-</td>
<td>2m33.428s</td>
<td>2m34.988s</td>
<td></td>
</tr>
<tr>
<td>openwrt/</td>
<td>8.4G</td>
<td>8.1G</td>
<td>8.5G</td>
<td>-</td>
<td>8.1G</td>
<td>9.0G</td>
<td></td>
</tr>
<tr>
<td>sdcard.img</td>
<td>286M</td>
<td>277M</td>
<td>254M</td>
<td>-</td>
<td>273M</td>
<td>303M</td>
<td></td>
</tr>
<tr>
<td>boot time</td>
<td>8.492709</td>
<td>10.559345</td>
<td>7.952850</td>
<td>-</td>
<td>6.982082</td>
<td>25.640092</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: OpenWrt build comparison
<table>
<thead>
<tr>
<th>name</th>
<th>Raspberry Pi 1</th>
<th>Raspberry Pi 2</th>
<th>BeagleBone Black</th>
<th>PandaBoard</th>
<th>Wandboard Quad</th>
<th>Asus Eee PC 1215n</th>
</tr>
</thead>
<tbody>
<tr>
<td>real time</td>
<td>27m17.629s</td>
<td>20m8.447s</td>
<td>20m8.447s</td>
<td>-</td>
<td>-</td>
<td>25m43.452</td>
</tr>
<tr>
<td>user time</td>
<td>87m36.840s</td>
<td>74m21.468s</td>
<td>74m21.468s</td>
<td>-</td>
<td>-</td>
<td>81m15.935</td>
</tr>
<tr>
<td>sys time</td>
<td>9m22.424s</td>
<td>4m31.152s</td>
<td>4m31.152s</td>
<td>-</td>
<td>-</td>
<td>7m12.955</td>
</tr>
<tr>
<td>Toolchain</td>
<td>15G</td>
<td>15G</td>
<td>15G</td>
<td>-</td>
<td>-</td>
<td>15G</td>
</tr>
<tr>
<td>real time</td>
<td>28m7.177s</td>
<td>25m44.733s</td>
<td>25m44.733s</td>
<td>-</td>
<td>-</td>
<td>27m42.724</td>
</tr>
<tr>
<td>user time</td>
<td>60m39.560s</td>
<td>58m52.088s</td>
<td>58m52.088s</td>
<td>-</td>
<td>-</td>
<td>59m54.615</td>
</tr>
<tr>
<td>sys time</td>
<td>4m13.768s</td>
<td>3m59.556s</td>
<td>3m59.556s</td>
<td>-</td>
<td>-</td>
<td>4m9.943</td>
</tr>
<tr>
<td>DistroKit/</td>
<td>5.5G</td>
<td>7.3G</td>
<td>7.3G</td>
<td>-</td>
<td>-</td>
<td>6.1G</td>
</tr>
<tr>
<td>sdcard.img</td>
<td>84M</td>
<td>81M</td>
<td>80M</td>
<td>-</td>
<td>-</td>
<td>105M</td>
</tr>
<tr>
<td>boot time</td>
<td>5.230498s</td>
<td>5.299048s</td>
<td>4.98392</td>
<td>-</td>
<td>-</td>
<td>20.59385</td>
</tr>
</tbody>
</table>

Table 4.3: PTXdist build comparison

<table>
<thead>
<tr>
<th>name</th>
<th>Raspberry Pi 1</th>
<th>Raspberry Pi 2</th>
<th>BeagleBone Black</th>
<th>PandaBoard</th>
<th>Wandboard Quad</th>
<th>Asus Eee PC 1215n</th>
</tr>
</thead>
<tbody>
<tr>
<td>real time</td>
<td>34m17.598s</td>
<td>34m24.622s</td>
<td>31m25.839s</td>
<td>19m0.417s</td>
<td>35m19.749s</td>
<td>38m29.501s</td>
</tr>
<tr>
<td>user time</td>
<td>201m4.080s</td>
<td>201m43.528s</td>
<td>178m14.064s</td>
<td>97m14.776s</td>
<td>265m9.384s</td>
<td>239m19.756s</td>
</tr>
<tr>
<td>sys time</td>
<td>14m32.588s</td>
<td>14m40.556s</td>
<td>13m8.288s</td>
<td>9m18.528s</td>
<td>14m0.868s</td>
<td>13m29.216s</td>
</tr>
<tr>
<td>yoctoprocject</td>
<td>24G</td>
<td>24G</td>
<td>26G</td>
<td>21G</td>
<td>24G</td>
<td>25G</td>
</tr>
<tr>
<td>sdcard.img</td>
<td>53M</td>
<td>53M</td>
<td>48M</td>
<td>49M</td>
<td>28M</td>
<td>26M</td>
</tr>
<tr>
<td>boot time</td>
<td>4.983091s</td>
<td>3.951380s</td>
<td>4.093724s</td>
<td>5.159032</td>
<td>3.539520</td>
<td>25.293039</td>
</tr>
</tbody>
</table>

Table 4.4: Yocto Project build comparison
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vCPUs</td>
<td>8 x 3.1GHz</td>
<td>8 x 2.3GHz</td>
<td>4 x 3.1GHz</td>
<td>4 x 2.3GHz</td>
<td>2 x 3.1GHz</td>
<td>2 x 2.3GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>30 GB</td>
<td>30 GB</td>
<td>15 GB</td>
<td>15 GB</td>
<td>7 GB</td>
<td>7 GB</td>
</tr>
<tr>
<td>price</td>
<td>1.542</td>
<td>1.049</td>
<td>0.765</td>
<td>0.519</td>
<td>0.395</td>
<td>0.272</td>
</tr>
<tr>
<td>real time</td>
<td>24m13.863s</td>
<td>24m52.772s</td>
<td>34m21.638s</td>
<td>37m10.730s</td>
<td>50m4.480s</td>
<td>57m17.304s</td>
</tr>
<tr>
<td>user time</td>
<td>73m20.016s</td>
<td>75m56.208s</td>
<td>71m51.860s</td>
<td>81m13.392s</td>
<td>72m39.468s</td>
<td>82m59.928s</td>
</tr>
<tr>
<td>sys time</td>
<td>8m1.556s</td>
<td>9m15.448s</td>
<td>7m35.004s</td>
<td>9m39.060s</td>
<td>7m41.560s</td>
<td>9m27.172s</td>
</tr>
<tr>
<td>total cost</td>
<td>0.62 PLN</td>
<td>0.42 PLN</td>
<td>0.43 PLN</td>
<td>0.32 PLN</td>
<td>0.33 PLN</td>
<td>0.26 PLN</td>
</tr>
</tbody>
</table>

Table 4.5: Comparison of VM size and build time (CPU instances)

<table>
<thead>
<tr>
<th>name</th>
<th>R2-30</th>
<th>R2-15</th>
<th>S1-8</th>
<th>S1-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCPUs</td>
<td>2 x 2.4GHz</td>
<td>2 x 2.4GHz</td>
<td>2 x 2.4GHz</td>
<td>1 x 2.4GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>30 GB</td>
<td>15 GB</td>
<td>8 GB</td>
<td>4 GB</td>
</tr>
<tr>
<td>price PLN/h</td>
<td>0.457</td>
<td>0.395</td>
<td>0.148</td>
<td>0.081</td>
</tr>
<tr>
<td>real time</td>
<td>52m33.380s</td>
<td>53m30.474s</td>
<td>65m42.849s</td>
<td>127m16.353s</td>
</tr>
<tr>
<td>user time</td>
<td>77m23.732s</td>
<td>80m41.752s</td>
<td>91m34.000s</td>
<td>101m2.204s</td>
</tr>
<tr>
<td>sys time</td>
<td>7m55.024s</td>
<td>7m36.960s</td>
<td>12m8.204s</td>
<td>15m2.652s</td>
</tr>
<tr>
<td>total cost</td>
<td>0.4 PLN</td>
<td>0.35 PLN</td>
<td>0.16 PLN</td>
<td>0.17 PLN</td>
</tr>
</tbody>
</table>

Table 4.6: Comparison of VM size and build time (RAM instances)
BUILDING AND DEPLOYING - CONTAINERS
CURRENTLY THERE IS ONLY ONE KING
HOW TO MAKE IT WORK?

- Download dedicated OS image from web
- or... build own image!
sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common
curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/stable"
sudo apt update
sudo apt install docker-ce
sudo usermod -aG docker $USER
# logout
docker pull alpine
docker images
docker run alpine ls -l
docker run -it alpine /bin/sh
BUILDING LINUX CONTAINERS
cd output/images
mkdir extra extra/etc extra/sbin extra/lib extra/lib64

touch extra/etc/resolv.conf
touch extra/sbin/init
cp /lib/x86_64-linux-gnu/libpthread.so.0 /lib/x86_64-linux-gnu/libc.so.6 extra/lib
cp /lib64/ld-linux-x86-64.so.2 extra/lib64

cp rootfs.tar fixup.tar
tar rvf --overwrite fixup.tar -C extra .
docker import - basic-system < fixup.tar
docker run -it basic-system /bin/sh
EDGE COMPUTING
What is edge computing? - The Verge
https://www.theverge.com/.../edge-computing-cloud-google-microsoft-apple-amazon

May 7, 2018 - Allow me to introduce you to “edge” computing. Edge is a buzzword. Like “IoT” and “cloud” before it, edge means everything and nothing.
Like VoIP:

... a methodology and group of technologies for ...
EDGE COMPUTING

- A paradigm: performing computation closer to data sources
- The idea behind edge computing
- From "lancard" to "gateway"
- Moving code execution between cloud and embedded
- Escaping from vendor/cloud lock-in...
How will the end of Net Neutrality affect Internet of Things

Sorry babe but our internet provider disapprove of our connection...
I have a crossover cable!
CONCLUSIONS
BUILD SYSTEMS ARE BETTER THAN DISTRIBUTIONS (FOR EMBEDDED)

- For professional usage
- Latest versions of kernel and packages
- Easy update and portability
COMPARABILITY

- Experience from one build system is useful on the others
- Software is quite easily portable between build systems
- Don't hesitate to experiment - try other builders
MOVING CODE BETWEEN CLOUD AND EDGE

- Keep small code footprint
- Avoid unnecessary 3rd party libraries
- Avoid/eliminate "dirty hacks"
- Automate as much as you can
- Don't be afraid to rewrite your code
DESKTOP

Share of Personal Computing Platforms

© asymco.com
MOBILE
Global Smartphone Market Share By Platform

Source: IDC, Strategy Analytics
EMBEDDED

Worldwide Unit Shipments of Embedded/Real-time Operating Systems

Commercially-Licensed Embedded/Real-time OSs

Open Source, Freely and/or Publicly Available Real-time OSs

Open Source, Freely and/or Publicly Available Linux

Commercial Linux

Note: More than one-third of embedded projects feature no formal OS or an in-house developed OS and are not depicted in the chart above.
OPEN QUESTIONS

- Why only Linux?
- What about MINIX, BSD, Darvin and HURD?
- Embedded Unix build systems?
- (yes, I know about Linux Standard Base and other "great standarizations"...)
THANK YOU FOR ATTENTION
BIBLIOGRAPHY IS INSIDE SOURCE CODE

cdynak.gitlab.io/linux2018

gitlab.com/cdynak/linux2018

github.com/cdynak/embedded-linux-build-systems
DISCUSSION / QUESTIONS?