Karol Kasperek Kamil Sulkowski

Laboratorium nr 1
SKPS

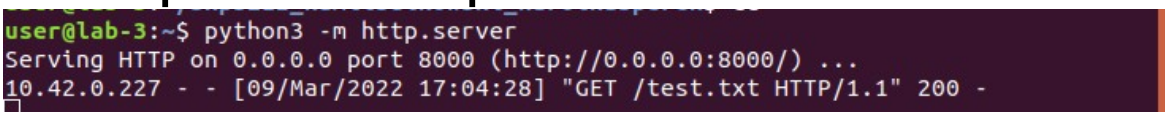
1. Przygotowanie środowiska.

Po otrzymaniu zestawu początkowego przeszliśmy do składania zestawu. Zestaw został złożony i zatwierdzony przez prowadzącego. Wszystkie urządzenia i kable zostały ułożone w sposób ułatwiający pracę na komputerze.

2. Pierwsze uruchomienie RPi.

Uruchomienie zasilania spowodowało włączenie się urządzenia. Wpisanie komendy ***tio /dev/ttyUSB0*** skutkowało poprawnym włączeniem się **tio**. Otrzymaliśmy informację *connected.* Pojawiły się logi z bootloadera i uzyskaliśmy możliwość zalogowania się. Załadował się system ratunkowy Raspberry Pi OS. Wpisanie podanych danych logowania zakończyło się poprawnym wejściem. Sprawdziliśmy stan połączenia sieciowego na RPi poleceniem ***ifconfig*** oraz ***ping 10.42.0.1*.**

3. Kopiowanie plików na RPi



Na komputerze host postawiliśmy serwer HTTP przy użyciu polecenia: ***python3 -m http.server****.* W katalogu roboczym stworzyliśmy plik test.txt. Przy użyciu komendy ***wget*** plik został pobrany na urządzenie RPi.

4.Kompilacja obrazu linuxa w buildroot

a) Obraz dla Raspberry Pi 4B z initramfs

Ściągnęliśmy plik ***buildroot-2021.08.tar.bz*.** Przy użyciu komendy ***tar -xf*** rozpakowaliśmy jego zawartość w docelowym katalogu. Zbudowaliśmy Buildroota wykonując polecenie ***make raspberrypi4_64_defconfig*** (domyślna konfiguracja). Następnie poleceniem ***make menuconfig*** zmieniliśmy konfigurację:

- Toolchain – zaznaczyliśmy external toolchain

- włączyliśmy initramfs

- wyłączyliśmy ext2/3/4

- włączyliśmy kompresję (gzip)

Po zmianach w konfiguracji zbudowaliśmy obraz poleceniem ***make***.

b) Obraz dla Raspberry Pi 4B bez initramfs

Na początku usunęliśmy stary obraz poleceniem ***make linux-dirclean***. Sprawdziliśmy w konfiguracji (poleceniem ***make menuconfig***) czy wszystko jest odpowiednio ustawione:

- Toolchain – zaznaczyliśmy external toolchain

- wyłączyliśmy initramfs

- włączyliśmy ext2/3/4

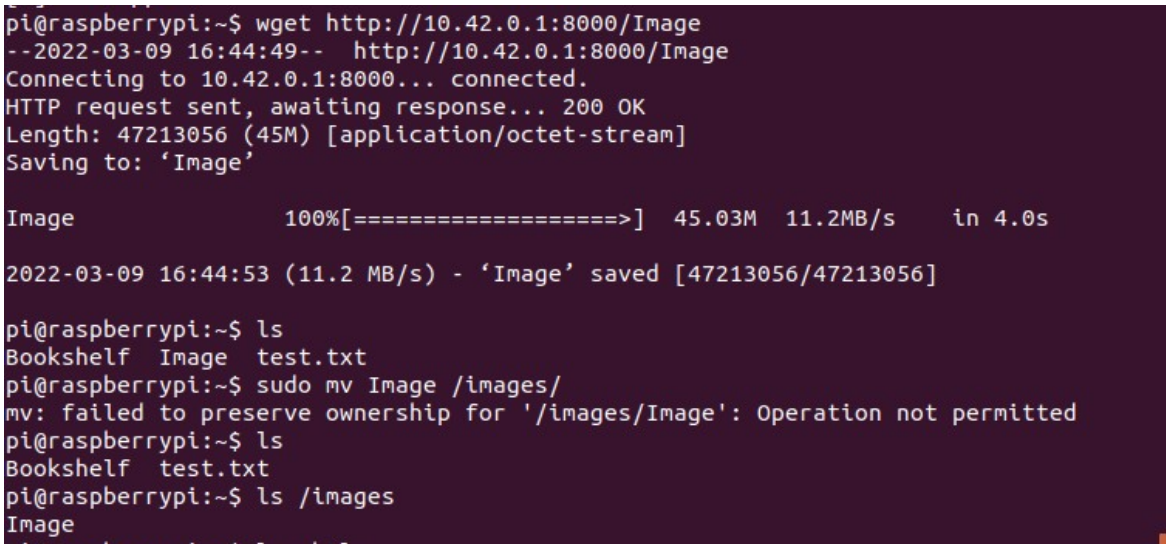
- włączyliśmy kompresję (gzip).

Ponownie zbudowaliśmy obraz poleceniem ***make***. Budowanie zakończono pomyślnie

4. Uruchomienie zbudowanego obrazu

a) Initramfs

Skopiowaliśmy plik Image z katalogu /output/images/ przez serwer HTTP. Na RPi ściągnęliśmy go przy użyciu komendy ***wget*** i przenieśliśmy go komendą ***sudo mv Image /images/.*** W konsoli otrzymaliśmy błąd ‘’Operation not permitted’’, ponieważ nie mamy dostępy do tej partycji z tego poziomu. Jednak obraz został przeniesiony pomyślnie.



Następnie przeszliśmy do restartu RPi przy użyciu komendy ***sudo reboot-h now***. Załadowaliśmy partycję typu FAT na karcie SD o numerze 0 z partycji nr 3 pod adres ${kernel_addr_r} plik o nazwie Image: ***fatload mmc 0:3 ${kernel_addr_r} Image***. Uruchomiliśmy obraz linuxa poleceniem: ***booti ${kernel_addr_r} - ${fdt_addr}.*** Działanie zostało zakończone pomyślnie. Plik stworzony na początku po ponownym załadowaniu obrazu nie został zapisany. Prowadzący zatwierdził poprawność działania.

```
Hit any key to stop autoboot:  0
U-Boot> fatload mmc 0:3 ${kernel_addr_r} Image
47213056 bytes read in 4364 ms (10.3 MiB/s)
U-Boot> booti ${kernel_addr_r} - ${fdt_addr}
Moving Image from 0x80000 to 0x200000, end=3050000
## Flattened Device Tree blob at 2eff3800
   Booting using the fdt blob at 0x2eff3800
   Using Device Tree in place at 000000002eff3800, end 000000002f002f2f

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd083]
[    0.000000] Linux version 5.10.46-v8 (user@lab-3) (aarch64-none-linux-gnu-gcc (
GNU Toolchain for the A-profile Architecture 10.2-2020.11 (arm-10.16)) 10.2.1 2020
1103, GNU ld (GNU Toolchain for the A-profile Architecture 10.2-2020.11 (arm-10.16
)) 2.35.1.20201028) #2 SMP PREEMPT Wed Mar 9 17:41:05 CET 2022
[    0.000000] Machine model: Raspberry Pi 4 Model B Rev 1.4
[    0.000000] efi: UEFI not found.
[    0.000000] Reserved memory: created CMA memory pool at 0x000000002ac00000, siz
[    4.993260] Freeing unused kernel memory: 28544K
[    4.998090] Run /init as init process
[    5.010853] mmc1: new high speed SDIO card at address 0001
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Saving random seed: [    5.115064] random: dd: uninitialized urandom read (512 byt
es read)
OK
Starting network: [    5.188646] bcmgenet fd580000.ethernet: configuring instance
for external RGMII (RX delay)
[    5.197483] bcmgenet fd580000.ethernet eth0: Link is Down
udhcpc: started, v1.33.1
udhcpc: sending discover
[    8.260279] bcmgenet fd580000.ethernet eth0: Link is Up - 100Mbps/Full - flow c
ontrol rx/tx
udhcpc: sending discover
udhcpc: sending select for 10.42.0.227
udhcpc: lease of 10.42.0.227 obtained, lease time 3600
deleting routers
adding dns 10.42.0.1
OK

Welcome to Buildroot
buildroot login:
```

b) Bez initramfs

Przekopiowaliśmy dodatkowy plik rootfs.ext2 z host na Rpi

```
pi@raspberrypi:~$ wget http://10.42.0.1:8000/rootfs.ext2
--2022-03-09 17:35:21--  http://10.42.0.1:8000/rootfs.ext2
Connecting to 10.42.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62914560 (60M) [application/octet-stream]
Saving to: ‘rootfs.ext2’

rootfs.ext2         100%[===================>]  60.00M  11.2MB/s    in 5.3s

2022-03-09 17:35:27 (11.2 MB/s) - ‘rootfs.ext2’ saved [62914560/62914560]
```

oraz nowy plik obrazu Image.

Następnie nagraliśmy system plików komendą ***sudo dd if=rootfs.ext2 of=/dev/mmcblk0p4 bs=4096.*** W konsoli U-boot powtórzyliśmy czynności z pierwszej części dodatkowo wpisując komendę ***setenv bootargs console=tty1 console=ttyAMA0,115200 root=/dev/mmcblk0p4 rootfstype=ext4 rootwait.*** Przy

```
pi@raspberrypi:~$ wget http://10.42.0.1:8000/rootfs.ext2
--2022-03-09 18:23:06--  http://10.42.0.1:8000/rootfs.ext2
Connecting to 10.42.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 104857600 (100M) [application/octet-stream]
Saving to: ‘rootfs.ext2.1’

rootfs.ext2.1       100%[===================>] 100.00M  11.2MB/s    in 8.9s

2022-03-09 18:23:15 (11.2 MB/s) - ‘rootfs.ext2.1’ saved [104857600/104857600]

pi@raspberrypi:~$ wget http://10.42.0.1:8000/Image
--2022-03-09 18:23:19--  http://10.42.0.1:8000/Image
Connecting to 10.42.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21719552 (21M) [application/octet-stream]
Saving to: ‘Image’

Image               100%[===================>]  20.71M  11.2MB/s    in 1.8s

2022-03-09 18:23:21 (11.2 MB/s) - ‘Image’ saved [21719552/21719552]

pi@raspberrypi:~$ sudo dd if=rootfs.ext2 of=/dev/mmcblk0p4 bs=4096
15360+0 records in
15360+0 records out
62914560 bytes (63 MB, 60 MiB) copied, 3.49126 s, 18.0 MB/s
pi@raspberrypi:~$ sudo mv Image /images/
mv: failed to preserve ownership for '/images/Image': Operation not permitted
pi@raspberrypi:~$ sudo reboot -h now
```

pierwszej próbie uruchomienia obrazu Linuxa pojawiał się błąd Kernel Panic. Po zdiagnozowaniu problemu zwiększyliśmy rozmiar partycji boot, lecz nie przyniosło to pozytywnego rezultatu. Rozwiązaniem problemu okazało się zwiększenie rozmiaru systemu plików.

```
[    3.190140]  ret_from_fork+0x10/0x38
[    3.193724] SMP: stopping secondary CPUs
[    3.197656] Kernel Offset: disabled
[    3.201146] CPU features: 0x0240022,61002000
[    3.205416] Memory Limit: none
[    3.208484] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs o
n unknown-block(179,4) ]---

PM_RSTS: 0x00001000
RPi: BOOTLOADER release VERSION:c2f8c388 DATE: Apr 29 2021 TIME: 17:11:29 BOOTMODE
: 0x00000006 part: 0 BUILD_TIMESTAMP=1619712685 0x03a0765c 0x00c03114 0x000acff6
PM_RSTS: 0x00001000
```

Następnie sprawdziliśmy, czy system korzysta z systemu plików na karcie SD. Stworzyliśmy plik test.txt, który po ponownym uruchomieniu systemu nadal istniał.