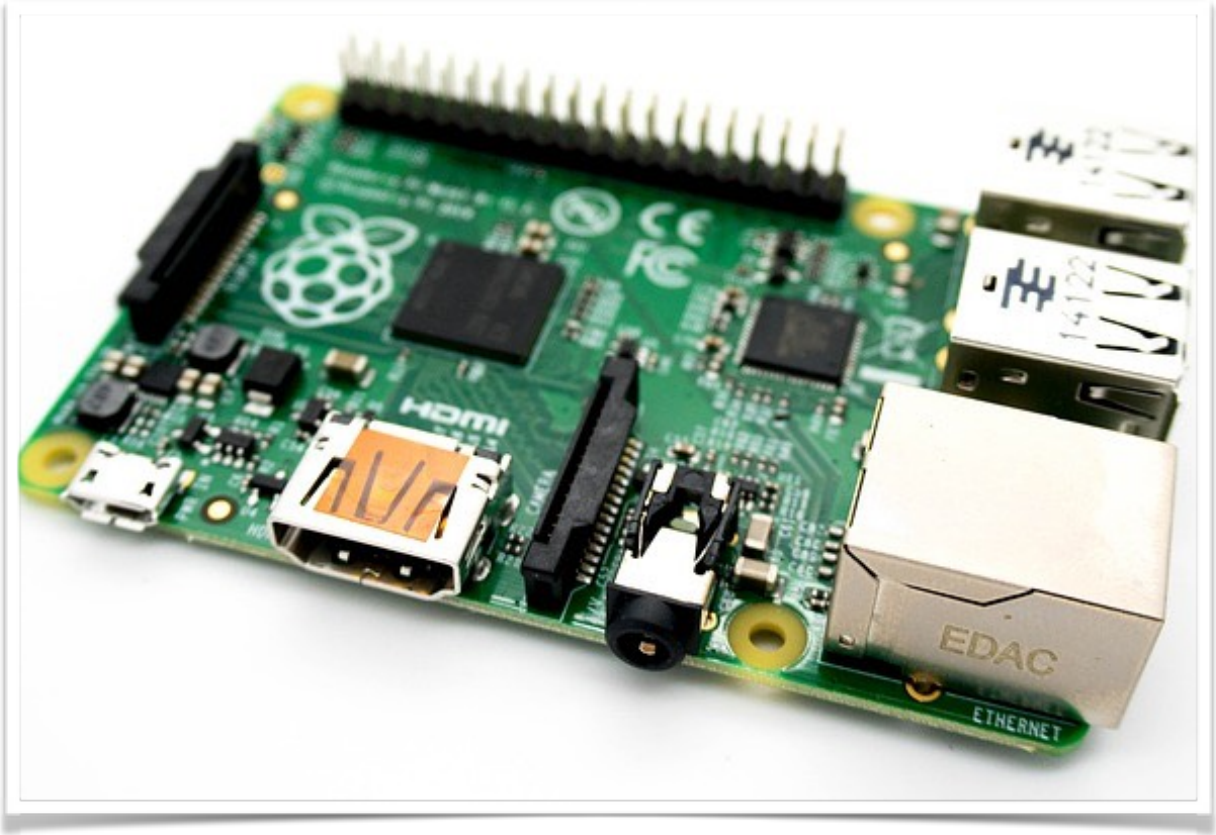
Linux w Systemach Wbudowanych



Laboratorium 3

Mateusz Sadowski

Lato 2018

1. Opis ćwiczenia

Główną częścią ćwiczenia było zbudowanie dwóch oddzielnych systemów: administratora oraz użytkownika. Należało obsłużyć ładowanie odpowiedniego wariantu w zależności od wprowadzonych danych podczas startu systemu przez użytkownika. Dane miały być wprowadzane za pomocą przycisków podłączonych do płytki. Dodatkowo start odpowiedniego systemu powinien być zasygnalizowany przy pomocy diod.

Kolejnym zadaniem było stworzenie aplikacji serwującej zawartość wybranego folderu na płytce za pomocą interfejsu WWW.

2. Rozwiązanie

2.1. System administratora

System administratora będzie działał z systemem plików *initramfs*. System ten jest włączany do jądra systemu przy kompilacji, więc nie trzeba w tym przypadku podejmować dodatkowych działań związanych z montowaniem systemu plików. Wszelkie zmiany wprowadzone w tym systemie plików w trakcie działania systemu są wymazywane przy restarcie. Aby wprowadzić stałe zmiany do systemu należy zmienić konfigurację i przekompilować obraz systemu.

Oprócz podstawowej konfiguracji opisanej w poradniku do laboratorium do systemu zostały dodane paczki:

- **dropbear** - w celu umożliwienia przesyłania plików po SSH
- **dosfstools**, **e2fsprogs**, **resize2fs** - paczki pozwalające na manipulację konkretnym typami systemów plików

System został skompilowany i nazwa utworzonego obrazu "zImage" w BRPATH/output/images została zmieniona na "zImage_administrator". Następnym krokiem jest przesłanie tego obrazu na pierwszą partycję karty SD przy pomocy systemu ratunkowego postępując według instrukcji w poradniku do laboratorium.

2.2. System użytkownika

System użytkownika będzie działał z systemem plików *ext4*. Obraz tego systemu plików jest generowany oddzielnie wraz z obrazem jądra systemu i należy go oddzielnie wgrać na kartę SD. Zmiany wprowadzone do tego systemu plików w trakcie działania systemu są zachowywane przy restarcie systemu.

Aby skrócić czas kompilacji w tym podpunkcie będzie użyty zmodyfikowany system z wcześniejszych laboratoriów. W tym celu należy wprowadzić następujące zmiany do konfiguracji:

- w "Filesystem Images" odznaczyć **initial RAM filesystem linked into linux kernel** i upewnić się, że jest zaznaczone **ext2/3/4 root filesystem**; można również na wszelki wypadek zwiększyć wielkość systemu plików pod Exact size do 300 Mb
- dodać paczkę **NODE_JS** w celu umożliwienia uruchomienia aplikacji serwującej pliki

- w “System configuration/Root filesystem overlay directories” podaj ścieżkę “board/mini/sadowskim/overlay/ do przygotowanej struktury folderów **overlay**, która będzie skopiowana bezpośrednio do systemu plików przy starcie systemu; opis struktury i plików się w niej znajdujących jest podany niżej

Struktura overlay systemu plików jest następująca:

- mini/sadowskim/overlay/
  - etc
    - init.d
      - web-file-browser.sh
    - fstab
  - home
    - web-file-browser
      - pliki źródłowe aplikacji

W folderze **/etc/init.d** umieszczone są skrypty uruchamiane przy starcie systemu. Z tego powodu został tam umieszony skrypt web-file-browser.sh, który jest opisany w sekcji z opisem działania aplikacji. W folderze **/home** zostanie dodany folder web-file-browser zawierający pliki źródłowe aplikacji. W pliku **fstab** została dodana linijka:
`dev/mmcblk0p3 /media/ ext4 defaults 0 0`,
która spowoduje automatyczne zamontowanie trzeciej partycji karty SD przy starcie systemu, co ma umożliwić serwowanie z niej plików.

Żeby ułatwić przesłanie obrazu *rootfs* na kartę SD należy zaznaczyć opcję “Filesystem images —> cpio the root filesystem (for use as an initial RAM filesystem)”. Opis przesyłania znajduje się podpunkcie 2.5.

Po skompilowaniu systemu nazwa utworzonego obrazu “zImage” w BRPATH/output/images została zmieniona na “zImage_user”. Następnym krokiem było przesłanie tego obrazu na pierwszą partycję karty SD przy pomocy systemu ratunkowego postępując według instrukcji w poradniku do laboratorium. Wgranie systemu plików na drugą partycję karty SD jest opisane w późniejszych podpunktach.

2.3. Bootloader

Do startowania systemu używany jest program *U-Boot*. Aby mieć wpływ na to jaki obraz jest ładowany przy starcie systemu należy dokonać edycji pliku **boot.txt** znajdującego się na pierwszej partycji karty SD. Zawartość gotowego plik boot.txt:

```
fdt addr ${fdt_addr}
fdt get value orig_bootargs /chosen bootargs
echo $orig_bootargs
gpio set 18
sleep 1
if gpio input 27 ; then
```

```
 gpio set 24
 gpio clear 18
 fatload mmc 0:1 ${kernel_addr_r} zImage_administrator
 load mmc 0:1 ${fdt_addr_r} bcm2708-rpi-b.dtb
 setenv bootargs "${bootargs_orig} console=ttyAMA0"
 bootz ${kernel_addr_r} - ${fdt_addr_r}
else
 gpio set 23
 gpio clear 18
 fatload mmc 0:1 ${kernel_addr_r} zImage_user
 load mmc 0:1 ${fdt_addr_r} bcm2708-rpi-b.dtb
 setenv bootargs "${bootargs_orig} console=ttyAMA0 root=/dev/mmcblk0p2 rootwait”
 bootz ${kernel_addr_r} - ${fdt_addr_r}
fi
```

Wykorzystywana jest komenda **gpio**, która pozwala na odczyt oraz ustawianie wartości na wtykach GPIO. W tym przypadku podczas wykonania skryptu najpierw zapali się biała dioda LED sygnalizująca oczekiwanie na decyzję użytkownika przez 1 sekundę. Jeśli użytkownik przytrzyma przycisk numer 27, załaduje się system administratora i zapali się czerwona dioda, jeśli użytkownik tego nie zrobi, to załaduje się system użytkownika i zapali się zielona dioda.

Dodatkowo dla systemu użytkownika ustawiono ścieżkę do drugiej partycji karty SD /dev/mmcblk0p2 (do głównego systemu plików) w opcji **root**.

Opcja **rootwait** sprawia, że system nie załaduje się dopóki nie zostanie wczytany system plików z karty SD.

Po zapisaniu zawartości pliku boot.txt należy wykonać komendę:

```
mkimage -T script -C none -n 'Start script' -d boot.txt boot.scr
```

Jest to konieczne, aby założona logika się wykonała.

2.4. Przygotowanie karty SD

Obrazy jądra systemu oraz skrypt uruchomieniowy boot.txt przechowywany jest na partycji pierwszej karty SD. Była ona już przygotowana w formacie VFAT i nie było potrzeby jej modyfikowania.

Do wykonania ćwiczenia były potrzebne dwie dodatkowe partycje na karcie w formacie *ext4*. Do partycjonowania karty wykorzystane było narzędzie **fdisk** a do formatowania partycji narzędzie **mkfs.ext4**. Kolejne kroki jakie trzeba podjąć zostały opisane poniżej:

1. fdisk -u /dev/mmcblk0 - powoduje rozpoczęcie pracy z /dev/mmcblk0, od tego momenty można wprowadzać komendy z klawiatury
2. p - powoduje wylistowania wszystkich partycji na danym urządzeniu, w tym momencie powinny znajdować się tam dwie partycje, należy zapamiętać na jakim numerze sektora kończy się partycja pierwsza (END_1), usuniemy drugą z nich (na pierwszej znajdują się pliki uruchomieniowe)

3. d (delete) —> 2 (numer partycji) —> w (zapisz) - (wciśnięcie klawisza ‘d’, następnie ‘2’, a następnie ‘w’) - powoduje usunięcie drugiej partycji i zapisanie tabeli partycji
4. n (new) —> 2 (numer partycji) —> p (primary) —> {END_1+1}<Enter> (początek) —> +{SIZE}M<Enter> (koniec) - powoduje utworzenie nowej drugiej partycji primary zaczynającej się zaraz po partycji pierwszej na karcie SD o rozmiarze SIZE megabajtów
5. n (new) —> 3 (numer partycji) —> p (primary) —> {END_2+1}<Enter> (początek) —> +{SIZE}M<Enter> (koniec) - powoduje utworzenie nowej trzeciej partycji primary zaczynającej się zaraz po partycji drugiej na karcie SD o rozmiarze SIZE megabajtów
6. po zakończeniu pracy z fdisk należy wywołać: mkfs.ext4 /dev/mmcblk0p2 oraz mkfs.ext4 /dev/mmcblk0p3, co spowoduje sformatowanie obu partycji do ext4

2.5. Transfer systemu plików na kartę SD

Do przesłania *rootfs* na drugą partycję karty SD wykorzystane zostanie narzędzie **cpio**. Z serwisu Wikipedia.pl:

“Archiwum cpio jest strumieniem plików i katalogów w jednym pliku, którego nazwa często kończy się rozszerzeniem .cpio. Archiwum posiada nagłówek, który pozwala aplikacjom, takim jak GNU cpio, na wyodrębnienie plików i katalogów do systemu plików. Nagłówek zawiera także informacje o nazwach plików, znacznikach czasu, właścicielach i uprawnieniach.”

Po zalogowaniu na system administratora należy zamontować drugą partycję karty pod wybranym folderem:
`mkdir /tmp/d2, mount /dev/mmcblk0p2 /tmp/d2`
Następnie na maszynie roboczej należy wykonać polecenie:
`cat output/images/rootfs.cpio | ssh root@xxx.yyy.zzz.mmm 'cd /tmp/user; cpio -i -d’`,
gdzie xxx.yyy.zzz.mmm - adres ip płytki. System plików zostanie skopiowany na mmcblk0p2.

Następnie dokonana została zmiana wielkości systemu plików, aby zajmował całe dostępne miejsce na partycji:

1. `umount /dev/mmcblk0p2` - przez zmianami należy odmontować partycję
2. `e2fsck -f /dev/mmcblk0p2` - wywołanie resize2fs wymaga wcześniejszego sprawdzenia systemu plików pod kątem błędów
3. `resize2fs /dev/mmcblk0p2` - wywołana bez parametrów rozszerzy system plików na całą partycję

2.6. Aplikacja serwująca pliki

Aplikacja serwująca folder z trzeciej partycji karty SD została napisana w języku JavaScript przy użyciu prostego frameworka do tworzenia aplikacji WWW - Express. W celu stworzenia tej aplikacji korzystałem z poradników dostępnych w internecie. Odnośniki są załączone poniżej.

Aby aplikację można było uruchomić na płytce, do systemu użytkownika należało dodać paczkę NODE_JS wraz z pakietem NPM. Skrypt startujący aplikację przy uruchomieniu systemu:

```
#!/bin/sh
cd /home/web-file-browser;
```

```
npm install;
node server.js
```

W pliku źródłowym aplikacji podane jest na jakim porcie serwer ma nasłuchiwać na połączenia:

```
const PORT = 8000;
```

Po czym jest uruchamiany z tym parametrem:

```
server.listen(PORT);
```

Zmienna dir przechowuje ścieżkę do serwowanego folderu: (wcześniej trzeba zamontować trzecią partycję karty pod folderem /tmp/d3, jest to rozwiązane przy pomocy pliku fstab - opis w podpunkcie 2.2)

```
var dir = "/media/public";
```

Uwierzytelnienie użytkownika zostało rozwiązane za pomocą wtyczki **express-basic-auth**. Pozwala ona na proste uwierzytelnianie za pomocą nazwy użytkownika i hasła.

Dokładny opis działania aplikacji, jako niezwiązany z przedmiotem, został pominięty w tym sprawozdaniu. Pliki źródłowe są załączone wraz z tym dokumentem.

3. Odtworzenie projektu

Aby odtworzyć projekt należy w dużej mierze postępować według kroków opisanych powyżej. Obrazy systemów i system plików można wygenerować przy pomocy odpowiednich plików **.config** załączonych wraz z dokumentem. Struktura folderów **overlay** zawierająca **pliki źródłowe aplikacji**, plik **fstab** oraz **skrypt uruchomieniowy** również zostały załączone. Folder **mini** należy umieścić w folderze /board w katalogu głównym Buildroota. Załączony plik **boot.txt** należy wgrać na partycję uruchomieniową i utworzyć obraz za pomocą opisanej komendy.

4. Źrodła informacji

- https://en.wikipedia.org/wiki/Initial_ramdisk
- https://pl.wikipedia.org/wiki/Cpio
- https://expressjs.com/en/starter/static-files.html
- https://chawlasumit.wordpress.com/2014/08/04/how-to-create-a-web-based-file-browser-using-nodejs-express-and-jquery-datatables/
- http://adrianmejia.com/blog/2016/08/24/Building-a-Node-js-static-file-server-files-over-HTTP-using-ES6/
- https://stackoverflow.com/questions/34472993/using-http-basic-auth-for-certain-urls-only-with-express-framework
- https://cellux.github.io/articles/diy-linux-with-buildroot-part-1/
- https://unix.stackexchange.com/questions/295131/what-is-zimage-rootfs?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- https://elinux.org/images/b/b1/Filesystems-for-embedded-linux.pdf
- https://community.linuxmint.com/tutorial/view/1513
- https://www.tecmint.com/fdisk-commands-to-manage-linux-disk-partitions/
- https://geekpeek.net/resize-filesystem-fdisk-resize2fs/