Warszawa 24.05.2024

Linux w Systemach Wbudowanych – Laboratorium ćw. 3

Student: Jakub Kępka

Treść zadania:

1. Przygotowanie administracyjnego obrazu systemu Linux, pracującego z initramfs i zawierającego narzędzia do zarządzania kartą SD
2. Podział karty na 3 partycje:
    a. VFAT z systemem ratunkowym oraz administracyjnym i jądrem systemu użytkowego
    b. ext4 - z systemem plików systemu użytkowego.
    c. ext4 - z danymi systemu użytkowego.
3. Przygotowanie użytkowego systemu Linux pracującego z systemem plików ext4 na drugiej partycji, realizującego serwer WWW (aplikacja webowa w środowisku Flask), który udostępnia pliki z partycji 3 na karcie SD i pozwala na wybranie pliku do załadowania. Dodatkowo serwer umożliwia uwierzytelnionym użytkownikom wgrywanie nowych plików na tę partycję
4. Przygotowanie bootloader'a, umożliwiającego wybór systemu. Przygotowany bootloader powinien sygnalizować swoje działanie w następujący sposób:
    a. Żółta dioda powinna zasygnalizować, że za chwilę zostaną sprawdzone przyciski.
    b. Po sekundzie należy odczytać stan przycisków. Jeśli żaden przycisk nie zostanie wciśnięty powinien zostać załadowany system użytkowy.
    c. Po sprawdzeniu przycisków żółta dioda powinna zostać zgaszona. Jeśli został wybrany system użytkowy, powinna zostać zapalona dioda zielona, a jeśli został wybrany system administracyjny, powinna zostać zapalona dioda czerwona.

Procedura odtworzenia projektu z załączonego archiwum:

Aby odtworzyć projekt należy rozpakować archiwum i wykonać skrypt.

Opis rozwiązania:

BR dla Admina jest konfigurowany w folderze kepkajAdmin, a dla Usera - w kepkajUser

Admin: Początkowa konfiguracja została ustawiona za pomocą poradniku do laboratorium umieszczonego na Leonie. Dodatkowo w sekcji "Bootloaders" zaznaczono "U-Boot", a w pozycji "Board defconfig" wpisano "rpi_4" - to pozwala na uruchomienie U-Boot'a na Raspberry Pi 4.
Zaznaczono: "e2fs" - włączenie obsługi systemów plików ext2, ext3 i ext4 w U-Boot, "fsck", "gptfdisk" - manipulowanie partycjami na dyskach, "mkimage" - przetworzenie skryptu startowego na odpowiedni format.
Po zbudowaniu systemu w katalogu BR/output/build/uboot-2022.04/tools utworzyłem skrypt boot.txt a następnie za pomocą narzędzia mkimage i z pomocą polecenia:
./mkimage -T script -C none -n 'Start script' -d boot.txt boot.scr.

User: Korzystając z opcji "Load" wgrałem poprzednią konfigurację. Wyłączyłem opcję "initial RAM filesystem linked into linux kernel" oraz dodałem potrzebne pakiety python do postawienia serwera.

Następnie (po zamontowaniu) przesłałem u-boot.bin (z folderu kepkajAdmin/BR/output/images) na rpi i zmieniłem nazwę na Image; wgrałem boot.scr na rpi do folderu, gdzie został zamontowany system. W katalogu /mnt/user zostały stworzone 2 foldery: Admin i User, a następnie wgrałem obrazy kepkajAdmin/BR/output/images/Image do Admin, a kepkajUser/BR/output/images/Image do User.

Po reboot'cie i zalogowaniu się na Admina skorzystałem z narzędnia fdisk (fdisk /dev/mmcblk0) dla partycjonowania dysku. Następnie wgrałem obraz systemu plików (rootfs.ext4) z katalogu kepkajUser/BR/output/images na partycję drugą za pomocą poleceia:
wget -O - 192.168.145.xxx:8000/rootfs.ext4 | dd of=/dev/mmcblk0p2 bs=4096.

Dla stworzenia systemu plików dla danych systemu użytkowego na partycji 3 skorzystałem z polecenia:
mkfs.ext4 /dev/mmcblk0p3.

W sesji Usera w katalogu output/init.d umieściłem plik S99web_server, który uruchamia serwer przy starcie systemu. Aplikacje umieszczono, w folderze /root (/root/app.py), w tym samym miejscu w katalogu /template znajduje się plik index.html.
W aplikacji najpierw określiłem lokalizację, do której będą przesyłane pliki, następnie zdefiniowałem z trzy endpointy:
/upload - wgranie plików po wprowadzeniu hasła ("root"),
/download/<filename> - pobranie pliku.
/ - lista plików przesłanych na serwer;
/delete/<filename> usuwanie pliku