Linux w systemach wbudowanych – Laboratorium 3

Tymon Felski

6 maja 2017

1 Treść zadania

Podczas trzeciego laboratorium należało wykonać następujące polecenia:

1. Przygotować "administracyjny" system Linux:
   - pracujący w initramfs,
   - zawierajacy narzędzia niezbędne do zarządzania kartą SD.
2. Przygotować "użytkowy" system Linux pracujący z systemem plików e2(3,4)fs na drugiej partycji, zawierający serwer WWW, sterowany przez interfejs WWW, który:
   - będzie udostępniać pliki z partycji 3 na karcie SD,
   - będzie umożliwiać wgrywanie nowych plików na tę partycję po uwierzytelnieniu użytkownika.
3. Przygotować bootloader, umożliwiający określenie, który system (administracyjny czy użytkowy) ma zostać załadowany.

2 Odtwarzanie projektu z załączonego archiwum

Dostarczone archiwum należy umieścić w dowolnym miejscu na dysku i rozpakować. W środku znajdują się pliki niezbędne do odtworzenia konfiguracji obrazów systemów administracyjnego i użytkowego oraz nieskompilowany skrypt bootloadera. Uruchomienie skryptów `install.sh`, znajdujących się w środku katalogów `admin/` i `user/` w archiwum, spowoduje odtworzenie konfiguracji odpowiednich środowisk z laboratorium. Jako parametr w kazdym z nich należy podać ścieżkę do katalogu z rozpakowanymi plikami Buildroota.

Skrypty utworzą katalogi `felskit-lab3-admin/` oraz `felskit-lab3-user/` równoległe do katalogów przekazanych jako argumenty. Następnie zostaną przekopiowane odpowiednie pliki i katalogi. Ponadto, oba skrypty zastosują domyślną konfigurację dla płytki Raspberry Pi i właściwie spatchują pliki konfiguracyjne Buildroota. Na koniec każdy ze skryptów zapyta czy rozpocząć kompilację jądra systemu.

3 Opis rozwiązania

3.1 Przygotowanie

W celu przygotowania obu środowisk Buildroot, ustawiono następujące opcje:

1. *System configuration* → *Port to run a getty (login prompt) on* na *ttyAMA0*
2. *Build options* → *Mirrors and Download locations* → *Primary download site* na *http://192.168.137.24/dl*
3. *Target options* → *Target ABI* na *EABI*
4. *Toolchain* → *Toolchain type* na *External toolchain*
5. *Toolchain* → *Toolchain* na *Sourcery CodeBench ARM 2014.05*

Dla obu obrazów włączono ich kompresję ustawiając opcję

Filesystem images → *tar the root filesystem, Compression method (gzip)*

a systemowi administracyjnemu dodatkowo zaznaczono opcję

Filesystem images → *initial RAM filesystem linked into linux kernel*

3.2 Zadania

3.2.1 Skrypt bootloadera

Jednym z celów ćwiczenia było przygotowanie skryptu bootloadera `U-Boot`, który umożliwiałby wybór obrazu systemu do załadowania. System ratunkowy jest dalej dostępny poprzez przerwanie procedury bootwania i wpisanie polecenia `run rescue`. Jeżeli natomiast nie zostanie wciśnięty żaden klawisz, nastąpi wykonanie poniższego skryptu.

```
gpio set 18
sleep 1
gpio clear 18

fatload mmc 0:1 ${fdt_addr_r} bcm2708-rpi-b.dtb
if gpio input 10 || gpio input 22 || gpio input 27; then
       gpio set 24
       fatload mmc 0:1 ${kernel_addr_r} zImage-admin
       setenv bootargs "${bootargs_orig} console=ttyAMA0"
else
       gpio set 23
       fatload mmc 0:1 ${kernel_addr_r} zImage-user
       setenv bootargs "${bootargs_orig} console=ttyAMA0 root=/dev/mmcblk0p2 rootwait"
fi
bootz ${kernel_addr_r} - ${fdt_addr_r}
```

Listing 1: Zawartość pliku boot.script

Skrypt zapala białą diodę na płytce, korzystając z interfejsu `GPIO`. Po upływie jednej sekundy jest ona gaszona i następuje sczytanie stanów przycisków. Jeżeli przynajmniej jeden z nich był w tym momencie wciśnięty, uruchomi się system administracyjny. W przeciwnym przypadku zostanie uruchomiony system użytkowy. Po wybraniu systemu zostanie zapalona odpowiednia dioda, czerwona lub zielona, sygnalizująca, który obraz został uruchomiony. Użycie `rootwait` pozwala uniknąć tzw. `kernel panic` przed wczytaniem systemu plików z innej partycji.

3.2.2 System administracyjny

System administracyjny ma pełnić rolę podobną do systemu ratunkowego, który poznaliśmy wcześniej. Został on wyposażony w narzędzia pozwalające na partycjonowanie dysku i zarządzanie nim, takie jak `parted`, `dosfstools` oraz `e2fsprogs`. Dodanie programu `parted` wymaga zaznaczenia opcji

Target packages → *Hardware handling* → *parted*

Aby wybrać zestaw narzędzi `dosfstools` zaznaczono opcję

Target packages → *Filesystem and flash utilities* → *dosfstools*

a także wszystkie trzy zawarte pod nią, czyli *fatlabel*, *fsck.fat* oraz *mkfs.fat*. Dodanie `e2fsprogs` ogranicza się do wybrania opcji

Target packages → *Filesystem and flash utilities* → *e2fsprogs*

Poza wybranymi domyślnie opcjami potomnymi została dodana jedynie *resize2fs*. Ponadto, system musi mieć możliwość transferu danych przy pomocy SSH. W tym celu potrzebny jest `dropbear`, który można znaleźć pod opcją

Target packages → *Networking applications* → *dropbear*

W celu ułatwienia korzystania z powyższego mechanizmu, ustawiono hasło do konta `root` wpisując ciąg znaków `Lab3` w pole

System configuration → *Root password*

Obraz tego systemu został umieszczony na pierwszej partycji karty pamięci, obok obrazu systemu ratunkowego, pod nazwą `zImage-admin`.

3.2.3 System użytkowy

System użytkowy, którego obraz jądra na pierwszej partycji karty pamięci nazywał się `zImage-user`, jest domyślnie wybierany przez skrypt bootloadera. W przeciwieństwie do systemu administracyjnego, system plików nie jest tutaj ładowany w całości do pamięci RAM (nie pracuje w `initramfs`). Znajduje się on na drugiej partycji, z której zostaje załadowany przez wcześniej opisywany skrypt.

Na potrzeby zadania do obrazu zostały dodane niezbędne paczki. Interpreter języka Python w wersji 3 można znaleźć pod opcją

> *Target packages* → *Interpreter languages and scripting* → *python3*

Ponadto obraz został wyposażony w środowisko Tornado poprzez zaznaczenie opcji

> *Target packages* → *Interpreter languages and scripting* → *External python modules* → *python-tornado*

Do poprawnego działania wykorzystywanego filemanagera wymagana była między innymi paczka `file`, którą można dodać wybierając opcję

> *Target packages* → *Shell and utilities* → *Utilities* → *file*

oraz samodzelnie dodany pakiet `python-magic`. Do katalogu `package/` dodano `python-magic/`, w którym utworzono dwa pliki: `Config.in` oraz `python-magic.mk`.

Pierwszy z nich ma za zadanie zdefiniowanie pakietu.

```
config BR2_PACKAGE_PYTHON_MAGIC
       bool "python-magic"
       depends on BR2_PACKAGE_PYTHON3
       help
         Python interface to the libmagic
         file type identification library

comment "python-magic needs python3"
       depends on !BR2_PACKAGE_PYTHON3
```

Listing 2: Zawartość pliku Config.in

Drugi plik odpowiada za definicję metody instalacji źródeł.

```
################################################################################
#
# python-magic
#
################################################################################

PYTHON_MAGIC_VERSION = 0.4.13
PYTHON_MAGIC_SITE = https://github.com/ahupp/python-magic.git
PYTHON_MAGIC_SITE_METHOD = git
PYTHON_MAGIC_LICENSE = MIT
PYTHON_MAGIC_SETUP_TYPE = setuptools

$(eval $(python-package))
```

Listing 3: Zawartość pliku python-magic.mk

Pliki źródłowe paczki pobierane są z repozytorium udostępnionego na GitHubie, do którego adres został podany w `PYTHON_MAGIC_SITE`. W pliku `package/Config.in` w sekcji *External python modules* należy dodać linię

```
source "package/python-magic/Config.in"
```

Wówczas możliwe będzie dodanie naszej paczki poprzez zaznaczenie opcji

> *Target packages* → *Interpreter languages and scripting* → *External python modules* → *python-magic*

Powyższe paczki są niezbędne do poprawnego działania opisanego niżej filemanagera.

3.2.4 Filemanager

Użyty filemanager jest modyfikacją filemanagera z repozytorium udostępnionego na GitHubie. Jest napisany w języku Python, wykorzystuje środowisko Tornado oraz Java Script i jQuery do obsługi interfejsu webowego.

Został on umieszczony w systemie użytkowym poprzez wykorzystanie mechanizmu Overlay. W opcji

System configuration → *Root filesystem overlay directories*

podano ścieżkę do katalogu `overlay/`, do którego zostały skopiowane wszystkie pliki źródłowe.

Filemanager udostępnia jedynie pliki znajdujące się na trzeciej partycji karty pamięci. Jest ona montowana w punkcie `/files` podczas startu systemu. Takie zachowanie zostało osiągnięte poprzez dodanie pliku `overlay/etc/fstab` rozszerzonego o linię

```
\dev\mmcblk0p3 \files ext4 defaults 0 2
```

Został on utworzony na podstawie domyślnego pliku `fstab` wygenerowanego podczas kompilacji obrazu systemu. Przed końcem kompilacji domyślna wersja tego pliku zostanie nadpisana tą z katalogu `overlay/`.

Aby serwer Tornado startował razem z systemem, co było przedmiotem dodatkowej części zadania, dodano plik `overlay/etc/init.d/S50filemanager`. Odpowiada on za uruchomienie serwera poprzez wykonanie skryptu `run.sh` przy starcie systemu.

```
#!/bin/sh
cd /filemanager
case $1 in
    start) ./run.sh > /var/log/filemanager.log 2>&1 & ;;
    stop) killall python3 ;;
esac
```

Listing 4: Zawartość pliku S50filemanager

Przeglądać katalogi może każda osoba, natomiast upload plików jest zabiezpieczony przez dekorator `@tornado.web.authenticated` i wymaga autoryzacji. Uwierzytelnianie jest zrealizowane przy pomocy mechanizmu `secure cookies`. Po kliknięciu przycisku `Login` na górnym pasku należy wpisać nazwę użytkownika `admin` oraz hasło `admin`.

3.2.5 Partycjonowanie i konfiguracja

Skrypt bootloadera `boot.script` został umieszczony na głównej partycji karty pamięci płytki, obok wszystkich obrazów systemów. Jego skompilowana wersja `boot.scr`, rozszerzona o sumę kontrolną, została wygenerowana przy pomocy polecenia

```
# mkimage -T script -C none -n "boot.script" -d boot.script boot.scr
```

Jako pierwszy uruchomiono system administracyjny, po czym skonfigurowano dodatkowe partycje karty pamięci oraz system użytkowy. Pierwszym krokiem było utworzenie dwóch partycji poleceniami

```
# mkfs.ext4 /dev/mmcblk0p2
# mkfs.ext4 /dev/mmcblk0p3
```

Dzięki programowi `dropbear` możliwe było skopiowanie przy pomocy SSH pliku `rootfs.tar.gz`, zawierającego skompresowany system plików systemu użytkowego. Po przesłaniu pliku można było przystąpić do instalacji systemu plików na drugiej partycji. Wypakowanie wymagało wykonania następującego polecenia

```
# gunzip -c rootfs.tar.gz | tar -xf -
```

Ostatnim krokiem konfiguracji było utworzenie katalogu `/files`, aby podczas uruchamiania systemu użytkowego mogła tam zostać podmontowana trzecia, wykorzystywana przez filemanagera, partycja karty pamięci.