Linux w systemach wbudowanych – Laboratorium 1

Tymon Felski

1 kwietnia 2017

1 Treść zadania

Podczas pierwszego laboratorium należało wykonać następujące czynności:

1. Uruchomić Raspberry Pi w trybie ratunkowym i zapoznać się z możliwościami tego trybu.
2. Przygotować obrazu systemu Linux, używającego `initramfs` jako głównego systemu plików:
   (a) Raspberry Pi powinno automatycznie podłączyć się do sieci, używając DHCP do otrzymania parametrów sieci. Jeśli kabel sieciowy nie jest podłączony, połączenie powinno zostać zestawione automatycznie po podłączeniu kabla. Odłączenie kabla powinno powodować wyłączenie połączenia sieciowego, a jego ponowne podłączenie – ponowne zestawienie.
   (b) System powinien mieć ustaloną nazwę jako `imie_nazwisko` autora.
   (c) Przy starcie systemu lub po podłączeniu kabla sieciowego, jeśli nie był on podłączony podczas startu, zegar systemowy powinien być automatycznie synchronizowany z jednym z serwerów NTP.
   (d) Obraz systemu powinien zawierać interpreter Pythona z prostym skryptem pokazującym jego działanie.
   (e) Oprócz użytkownika `root` w systemie powinno zostać stworzone konto użytkownika domyślnego (o dowolnie wybranej nazwie). Hasła obu użytkowników powinny być ustalone.
3. Skrypt z podpunktu (d) powinien dodatkowo wysyłać do komputera laboratoryjnego informację o godzinie startu systemu wykorzystując protokół UDP.

2 Odtwarzanie projektu z załączonego archiwum

Dostarczone archiwum należy umieścić w dowolnym miejscu na dysku i rozpakować. Uruchomienie skryptu `install.sh`, znajdującego się w środku archiwum, spowoduje odtworzenie konfiguracji środowiska z laboratorium. Jako parametr należy podać ścieżkę do katalogu z rozpakowanymi plikami Buildroota.

Skrypt utworzy katalog `felskit-lab1/` równoległy do katalogu przekazanego jako argument. Następnie do obu z nich zostaną przekopiowane odpowiednie pliki i katalogi. Ponadto, skrypt zastosuje domyślną konfigurację dla płytki Raspberry Pi i właściwie spatchuje pliki konfiguracyjne Buildroota. Na koniec skrypt zapyta czy rozpocząć kompilację jądra systemu.

3 Opis rozwiązania

3.1 Przygotowanie

Na początku ćwiczenia płytka została uruchomiona w trybie ratunkowym poleceniem

```
$ run rescue
```

po wcześniejszym wciśnięciu klawisza w czasie oczekiwania przez bootloader U-boot. Po zapoznaniu się z możliwościami tego trybu takimi jak łączenie się przez SSH z komputerami laboratoryjnymi i wgrywania własnego obrazu systemu, przystąpiono do głównej części zadania.

Po pobraniu Buildroota w wersji 2016.11.2, zastosowano konfigurację domyślną dla płytki Raspberry Pi przy pomocy polecenia

```
$ make raspberrypi_defconfig
```

Następnie, w celu przygotowania środowiska 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*

Ponadto, zaznaczono opcję

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

oraz włączono kompresję obrazu ustawiając opcję

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

3.2 Zadania

3.2.1 Automatyczne konfigurowanie sieci

Pozyskiwanie parametrów sieci jest możliwe dzięki domyślnie wkompilowanemu programowi `udhcp`. Aby system automatycznie konfigurował interfejsy sieciowe przy jego pomocy, należy dołączyć paczkę zawierającą daemon `ifplugd`. Jest ona zależna od `BR2_PACKAGE_BUSYBOX_SHOW_OTHERS`, więc konieczne jest zaznaczenie opcji

Target packages → *BusyBox* → *Show packages that are also provided by busybox*

która spowoduje wyświetlenie dodatkowych paczek dostarczanych przez busybox. Wówczas możliwe jest zaznaczenie pakietu zawierającego interesujący nas daemon `ifplugd`

Target packages → *Networking applications* → *ifplugd*

Wraz z pakietem, do obrazu systemu dodawany jest skrypt, którego zadaniem jest uruchomienie daemona przy starcie systemu. Co więcej, `ifplugd` śledzi zmiany stanu połączenia sieciowego i odpowiednio konfiguruje interfejs sieciowy w przypadku odłączenia/podłączenia kabla.

3.2.2 Nazwa systemu

Zmiana nazwy hosta na `tymon_felski` polega na wpisaniu tego ciągu znaków w polu

System configuration → *System hostname*

3.2.3 Synchronizacja zegara

W celu umożliwienia automatycznej sychronizacji zegara systemowego przy pomocy protokołu NTP podczas startu systemu i odłączania/podłączania kabla sieciowego wykorzystano dołączony wcześniej do obrazu daemon `ifplugd` oraz program `ntpdate`, który można dodać zaznaczając opcję

Target packages → *Networking applications* → *ntp* → *ntpdate*

Do tego rozwiązania niezbędne jest także dodanie paczki `ifupdown`, co można zrobić ustawiając opcję

Target packages → *Networking applications* → *ifupdown*

Ponieważ daemon `ifplugd` śledzi zmiany stanu połączenia sieciowego, można go wykorzystać do wywołania polecenia

```
$ ntpdate pl.pool.ntp.org
```

zaraz po podłączeniu kabla, które zsynchronizuje zegar przy użyciu jednego z serwerów NTP z polskiej puli. Można także skonfigurować wywoływanie tego polecenia przy starcie systemu. Aby to osiągnąć, należy wykorzystać mechanizm Overlay. W opcji

System configuration → *Root filesystem overlay directories*

podano ścieżkę do katalogu `overlay/` z plikami, które zostaną dodane do obrazu pod koniec kompilacji i, jeżeli będzie taka konieczność, nadpiszą poprzednie. Wewnątrz utworzono katalog `etc/init.d/`, do którego dodano plik `S51ntpdate`, który spowoduje ustawienie odpowiedniego czasu przy starcie systemu.

```
#!/bin/sh
ntpdate pl.pool.ntp.org
exit 0
```

Listing 1: Zawartość pliku S51ntpdate

Plik o takiej samej zawartości został również dodany do `overlay/etc/network/if-up.d/`, co będzie skutkować wykonaniem go zaraz po podłączeniu kabla sieciowego.

3.2.4 Interpreter Pythona i prosty skrypt

Dodanie interpretera języka skryptowego Python do obrazu systemu polega na zaznaczeniu opcji

Target packages → *Interpreter languages and scripting* → *python*

Następnym krokiem jest przygotowanie paczki Buildroota, która wkompiluje do obrazu systemu nasz skrypt. Do katalogu `package/` dodano `python-script/`, w którym utworzono dwa pliki: `Config.in` oraz `python-script.mk`.

Pierwszy z nich ma za zadanie zdefiniowanie pakietu i jego zależności od innych (w tym wypadku od `BR2_PACKAGE_PYTHON`).

```
config BR2_PACKAGE_PYTHON_WELCOME
    bool "python-welcome"
    depends on BR2_PACKAGE_PYTHON
    help
      Python welcome script.

      https://www.python.org/

comment "python-welcome needs python"
    depends on !BR2_PACKAGE_PYTHON
```

Listing 2: Zawartość pliku Config.in

Drugi plik odpowiada za definicję metody instalacji źródeł, do których podano w nim ścieżkę.

```
################################################################################
#
# python-welcome
#
################################################################################

PYTHON_WELCOME_VERSION = 1.0
PYTHON_WELCOME_SITE = $(TOPDIR)/../felskit-lab1/python-welcome
PYTHON_WELCOME_SITE_METHOD = local
PYTHON_WELCOME_DEPENDENCIES = python
PYTHON_WELCOME_LICENSE = MIT
```

```
define PYTHON_WELCOME_INSTALL_TARGET_CMDS
    $(INSTALL) -D -m 0755 $(@D)/hello-world.py $(TARGET_DIR)/usr/bin
endef

$(eval $(generic-package))
```

Listing 3: Zawartość pliku python-welcome.mk

Tutaj plikiem źródłowym jest `hello-world.py`, będący pythonową wersją programu "Hello world!". Dodatkowa część zadania polegała na takim zmodyfikowaniu skryptu, aby wysyłał do komputera laboratoryjnego pakiet UDP z datą startu systemu. Wykorzystano tutaj moduł `socket` oraz możliwość pobierania aktualnej daty i godziny za pomocą modułu `datetime`.

```
#!/usr/bin/python
import socket
import datetime

print "Hello world!"

UDP_IP = "192.168.143.171"
UDP_PORT = 56000
MESSAGE = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

print "UDP target IP:", UDP_IP
print "UDP target port:", UDP_PORT
print "message:", MESSAGE

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

Listing 4: Zawartość pliku hello-world.py

W pliku `package/Config.in` w sekcji *External python modules* należy dodać linię

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

Wówczas możliwe będzie dodanie naszej paczki poprzez zaznaczenie opcji

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

Skrypt jest uruchamiany podczas startu systemu, ponieważ do wcześniej utworzonego katalogu `overlay/etc/init.d/` dodano plik `S50python-welcome`, który uruchamia nasz skrypt.

```
#!/bin/sh
hello-world.py
exit 0
```

Listing 5: Zawartość pliku S50python-welcome

3.2.5 Domyślny użytkownik i hasła

Hasło do konta `root` można ustawić przy pomocy opcji

> *System configuration* → *Root password*

Dodanie dodatkowego użytkownika jest zrealizowane za pośrednictwem tablicy użytkowników, do której ścieżkę należy podać w opcji

> *System configuration* → *Path to the users tables*

Poniższy wpis w tablicy użytkowników skutkuje dodaniem użytkownika o nazwie `felskit` i haśle `password`.

```
felskit -1 users -1 =password /home/felskit /bin/sh - Tymon Felski
```

Listing 6: Zawartość pliku users