Warsaw, 21.03.2021

Linux for Embedded Systems - Laboratory ex. 1

Emilia Wróblewska 291674

Problem Description

1. Virt 64 should automatically connect to the network, using DHCP to obtain the network parameters. Disconnection of the cable should shut down the network connection. Connection of the cable should reconnect the network.

2. The system host name should be set to: “firstname_lastname” of the student

3. When the Internet connection is established, the system clock should be synchronized with an NTP server (preferably using the „NTP server pool”).

4. Except for the „root” user, there should be a plain user (with arbitrarily chosen name) account created. Both users should have set the passwords.

5. Individual part: Add the Lua interpreter with the script that after the start of the system tries to download from certain URL *update.tar.gz* file and unpacks it overwriting the filesystem

Description of the Solution

Before I can start solving the problems I need to download Buildroot from https://buildroot.org/downloads. The adviced version is 2021.02, so I am using this exact version. After downloading and unpacking the files, I can open a terminal window, navigate to the unpacked directory and use the default configuration for Virt64.

```
$make qemu_aarch64_virt_defconfig
```

After that to add some of my changes I can do:

$make menuconfig

which starts a graphical interface for configuration.
Now I modify configurations as recommended in the laboratory guide:

Toolchain → Toolchain type → External toolchain
Toolchain → Toolchain → "Arm AArch64 2020.11" (automatically set)
Build options → Enable compiler cache
Filesystem images → initial RAM filesystem linked into linux kernel

1. For Virt 64 to connect and disconnect to internet automatically when plugging in and out a network cable:

Target packages → Show packages that are also provided by busybox
Target packages → Networking applications → netplug

2. To set the system host name:

System configuration → System hostname → "emilia_wroblewska"

3. For clock synchronization:

Target packages → Networking applications → ntp → ntpd
Target packages → Networking applications → ntp → ntpdate

I also added a short script that synchronizes the clock every time the Internet connection is established. It is placed in the "overlay/etc/network" directory created outside the buildroot directory.

System configuration → Root filesystem overlay directories → "../overlay"
(overlay/etc/network/synchronize) ntpdate -u 0.pl.pool.ntp.org
(overlay/etc/network/interfaces) added post-up case:

```
….
post-up /etc/network/synchronize
wait-delay 15
….
```

4. In order to set a password for root:

System configuration → Root password → "test03"

For adding a new user I need to create a file describing a table of users. As described in the buildroot manual:
https://buildroot.org/downloads/manual/manual.html#customize-users
https://buildroot.org/downloads/manual/manual.html#makeuser-syntax
My file is named “users_table” and is located in the buildroot directory. It needs to have executable permissions so I set it with: $chmod +x users_table. Then I can set:

System configuration → Path to the users tables → “users_table”

5. For running a Lua script I have to add some target packages:

System configuration → Root filesystem overlay directories → “../overlay”
Target packages → Interpreter languages and scripting → lua
Target packages → Interpreter languages and scripting → Lua libraries/modules → lua-http
Target packages → Interpreter languages and scripting → Lua libraries/modules → luasocket
Target packages → Interpreter languages and scripting → Lua libraries/modules → luasec
Target packages → Compressors and decompressors → gzip
Target packages → System tools → tar

Then after creating my dummy file *update.tar.gz* I place it in the “server” directory outside the buildroot directory. Then I can make it available for other local virtual machines to be downloaded by creating an http server in this directory:

$cd server
$python3 -m http.server

The next step is to create the lua script in overlay/etc/init.d/S97script.lua . The script sends an http request to my local server and can download the *update.tar.gz* file from url: http://10.0.2.2:8000/update.tar.gz . The file is put in “/etc/update.d” - an empty directory created only to store my dummy file.

```
#!usr/bin lua

print("Downloading update.tar.gz...")
local http = require("socket.http")
local socket = require("socket")
local ltn12 = require("ltn12")
local host, port = "10.0.2.2", 8000
```

```
local file = ltn12.sink.file(io.open('/etc/update.d/update.tar.gz',
'w+b'))
local body, code = http.request{
    url = "http://10.0.2.2:8000/update.tar.gz",
    sink = file,
}
if not body then
    print("No update found")
else
    print(code)
    print("Update found")

    os.execute("tar --overwrite -xvf /etc/update.d/update.tar.gz -C
/etc/update.d")
    local f = io.open('/etc/update.d/update.txt', 'rb')
    local content = f:read("*all")
    print(content)
    f:close()
end
```

And finally, the updated rcS file:

```
#!/bin/sh


# Start all init scripts in /etc/init.d
# executing them in numerical order.
#
for i in /etc/init.d/S??* ;do

     # Ignore dangling symlinks (if any).
     [ ! -f "$i" ] && continue

     case "$i" in
    *.sh)
        # Source shell script for speed.
        (
        trap - INT QUIT TSTP
```

```
        set start
        . $i
        )
        ;;
    *.lua)
        (
        lua .$i
        )
        ;;
    *)
        # No sh extension, so fork subprocess.
        $i start
        ;;
    esac
done
```

Procedure to recreate the design from the archive

After unpacking the archive one should execute the *build.sh* script, which will build download, unpack and prepare buildroot configuration. Now after compilation one should execute: $./run_before to start hosting the *update.tar.gz* file. Then, in new terminal (without closing the one with server) one should start the virtual machine with the command: $./runme