[[si:lab:lab2]]



# Configurarea sistemului

Table of Contents
Configurarea sistemului
1. Introducere
2. Iniţializarea sistemului
2.1. Bootloader, încărcare kernel
2.2 Init
2.3. Exerciţii
3. Anexă
Link-uri simbolice
Montare
Vizualizarea unui fişier mare
Editarea unui fişier cu vi

## 1. Introducere

Vor fi prezentate în continuare aspecte diverse legate de configurarea sistemelor încorporate bazate pe Linux, în special NGW100. Se va descrie procesul de încărcare a sistemului de operare, punctându-se fişierele ce pot fi modificate.

## 2. Iniţializarea sistemului

### 2.1. Bootloader, încărcare kernel

La punerea sub tensiune, procesorul AT32AP7000 începe să execute cod de la adresa 0, unde este conectată memoria Flash. În consecinţă, la începutul memoriei se află un bootloader, un mic program a cărui sarcină principală este să încarce sistemul de operare.

La fabricarea unui sistem precum NGW100 memoria Flash este în principiu goală (numai când se produc foarte multe unităţi putem cere furnizorului memorii preprogramate). Scrierea bootloaderului în memoria Flash este prima etapă după fabricarea hardware-ului şi se realizează cu un dispozitiv special de testare şi depanare. Acest dispozitiv se conectează la procesor printr-un port JTAG (Joint Test Action Group) şi îi comandă acestuia să scrie în memorie.

Odată încărcat în Flash şi rulat, un bootloader avansat va permite încărcarea prin metode tradiţionale a imaginii sistemului de operare în memoria RAM, fără a mai necesita instrumentul JTAG: prin conexiune serială, reţea, card MMC/SD etc. Se poate de asemenea scrie imaginea sistemului în Flash pentru a nu mai fi necesară încărcarea manuală la pornirea viitoare. Sistemul NGW100 foloseşte bootloader-ul U-Boot, care permite toate aceste operaţii. Comunicaţia cu U-Boot se face prin portul serial al sistemului, interfaţa fiind o consolă simplă.

U-Boot foloseşte o zonă din Flash pentru a stoca informaţii de configurare, aşa-numitul environment. Aceste informaţii pot fi afişate cu comanda U-Boot ***printenv***, sau din Linux cu comanda ***fw_printenv***:

```
~# fw_printenv
baudrate=115200
ethaddr=00:04:25:1C:90:7E
bootdelay=1
ethact=macb0
serverip=192.168.0.106
tftpip=192.168.0.106
eth1addr=00:04:25:1C:90:7F
bootargs=console=ttyS0 root=/dev/mtdblock1 rootfstype=jffs2
stdin=serial
stdout=serial
stderr=serial
bootcmd=fsload /boot/uImage;bootm
```

Parametrii pot fi modificaţi din U-Boot cu comanda askenv <nume_parametru> şi salvaţi în Flash cu comanda saveenv. NU se vor modifica aceşti parametri.

La pornire, U-Boot aşteaptă o secundă (***bootdelay=1***) primirea unui caracter Space, prin apăsarea tastei respective într-o consolă serială. În acest caz va afişa un prompt şi va executa comenzile date. Altfel, va executa automat comenzile date de parametrul bootcmd: va încărca din Flash în RAM kernel-ul, apoi îi va ceda controlul, cu argumentele bootargs: va folosi portul serial drept consolă şi va monta rădăcina sistemului de fişiere din Flash.

Pentru a încărca de pe un card MMC kernel-ul, se vor da, spre exemplu, următoarele comenzi:

```
Uboot> askenv bootcmd
Please enter 'bootcmd':mmcinit; ext2load mmc 0:1 0x10400000 /boot/uImage; bootm
Uboot> set bootargs 'console=ttyS0 root=/dev/mmcblk0p1 rootwait'
Uboot> boot
```

Se observă posibilitatea U-Boot de a încărca un kernel (/boot/uImage) de pe un sistem de fişiere standard. ***fsload*** încarcă de pe un sistem de fişiere jffs2 aflat în Flash, iar ***ext2load*** încarcă de pe un sistem de fişiere ext2.

### 2.2 Init

După ce este pornit de bootloader, kernel-ul va iniţializa dispozitivele esenţiale şi va monta sistemul de fişiere rădăcină (***root, /***). Dispozitivul pe care se află acest sistem de fişiere este dat din bootloader în argumentul ***root***. Astfel, ***root=/dev/mtdblock1*** montează blocul 1 al memory technology devices, definit în codul sursă al kernel-ului pentru platforma pe care se va executa.

În cazul NGW100 blocul 1 se află în memoria Flash principală, după U-Boot. Blocul 0 conţine U-Boot, blocul 2 conţine environment-ul U-Boot, iar blocul 3 se află în memoria Flash secundară şi conţine sistemul de fişiere ***/usr***. În cazul folosirii unui card MMC pentru boot-are, ***root=/dev/mmcblk0p1*** montează prima partiţie a cardului.

Argumentul ***single*** poate fi trimis kernel-ului din U-Boot pentru a activa imediat o consolă cu un shell (aşa-numitul mod single-user). Acest mod previne rularea etapelor ulterioare ale procesului de iniţializare, fiind folosit pentru depanare sau reconfigurare.

Dacă nu se specifică single, se încarcă în continuare executabilul ***/sbin/init*** de pe sistemul de fişiere abia montat. Init va consulta fişierul ***/etc/inittab*** pentru a stabili ce va face în continuare.

În cazul NGW100, liniile relevante sunt:

***::sysinit:/etc/init.d/rcS*** - va rula scriptul rcS aflat în directorul /etc/init.d/

***::shutdown:/etc/init.d/rcK*** - înainte de oprirea sistemului va rula scriptul rcK

***ttyS0::respawn:-/bin/sh*** - după rularea rcS, pe consola serială va da un shell, care va fi repornit în caz că se termină, pentru a nu bloca accesul utilizatorului la sistem. În cazul altor sisteme, se va rula în loc de shell un program numit getty, care se va ocupa de login, utilizatorul primind un shell numai după ce se autentifică.

Să analizăm scriptul rcS:

```
~ # cat /etc/init.d/rcS
#!/bin/sh
for s in /etc/init.d/S*; do
if [ x
$s ]; then
$s start
fi
done
echo
echo "NGW100 ready"
echo
```

După cum se poate vedea, sunt rulate toate fişierele executabile din directorul ***/etc/init.d*** care încep cu **S**, cu parametrul start. (Un fişier este executabil în Linux dacă are atributul x setat; un executabil nu trebuie să conţină neapărat cod maşină, ci poate fi un script, interpretat de un shell.)

Similar, scriptul rcK va rula fişierele ce încep cu K la oprire, cu parametrul stop. Parametrii start şi stop sunt daţi deoarece, pe sistemele Linux mari, convenţia este ca o funcţionalitate să fie implementată de un singur script, la care se fac două link-uri, unul cu S* şi unul cu K*.

Se observă că scripturile de start conţin un număr după litera S, pentru a fi executate într-o ordine bine stabilită - serviciile de nivel înalt depind de servicii de nivel mai jos:

```
~ # ls /etc/init.d/S*
/etc/init.d/S00mountvirtfs    /etc/init.d/S21dnsmasq
/etc/init.d/S01hotplug        /etc/init.d/S22iptables
/etc/init.d/S02hostname       /etc/init.d/S41inetd
/etc/init.d/S05avahisetup.sh  /etc/init.d/S42httpd
/etc/init.d/S08syslog         /etc/init.d/S43ntpdate
/etc/init.d/S09klog           /etc/init.d/S49netfs
/etc/init.d/S10modulesinit    /etc/init.d/S49ntp
/etc/init.d/S13portmap        /etc/init.d/S50dropbear
/etc/init.d/S15localfs        /etc/init.d/S91smb
/etc/init.d/S20network
```

Fiecare script va efectua o anumită etapă de iniţializare şi de obicei va afişa un mesaj de succes/eşec, după care se lansează shell-ul (interpretorul de comenzi). Pe calculatoarele dotate cu display, unul din scripturi încarcă sistemul grafic.

| **Script** | **Funcţie** |
|---|---|
| **mountvirtfs** | montează sistemele de fişiere virtuale (care nu au un dispozitiv fizic ca suport): /proc, /sys, /config - structura de fişiere şi directoare este generată de kernel pentru a expune structuri de date interne - procese, dispozitive hardware, configurarea anumitor subsisteme etc. /dev - conţine fişiere speciale bloc şi caracter, ce permit prezentarea dispozitivelor hardware sau virtuale sub formă de fişiere, accesibile cu funcţiile clasice de lucru cu fişierele (open, read, write, ioctl). Accesul unui fişier special de către un program trimite apelul la driver-ul specific din kernel. Legătura dintre fişierul special şi driver-ul din kernel se face prin două atribute comune fişierului şi driver-ului, numite major number şi minor number (nu prin nume, dar numele sunt stabilite prin convenţie). /dev/pts - conţine fişiere speciale caracter pentru pseudo-terminale - terminale ce nu au un suport fizic cum ar fi o linie serială sau o consolă text. Se folosesc de exemplu pentru rularea unui shell prin ssh sau într-o fereastră în sistemul grafic. /tmp - pentru fişiere temporare, memorate în RAM |
| **hostname** | setează numele maşinii, citindu-l din fişierul /etc/hostname. |
| **modules-init** | încarcă în kernel modulele specificate în fişierul /etc/modules, apelând modprobe pentru fiecare linie |
| **localfs** | montează sistemele de fişiere locale, apelând mount a. Mount va consulta fişierul /etc/fstab, unde sunt listate dispozitivele ce trebuie montate. În cazul NGW100, se va monta /dev/mtd3 (memora Flash secundară) în directorul /usr. |
| **network** | Iniţializează interfeţele de reţea, apelând ifup. Ifup consultă fişierul /etc/network/interfaces pentru a stabili configurarea interfeţelor. Conţinutul acestui fişier este următorul:<br>`~ # cat /etc/network/interfaces`<br>`# Configure Loopback`<br>`auto lo`<br>`iface lo inet loopback`<br>`# Configure Ethernet 0`<br>`auto eth0`<br>`iface eth0 inet dhcp`<br>`# Configure Ethernet 1`<br>`auto eth1`<br>`iface eth1 inet static`<br>`address 10.0.0.1`<br>`netmask 255.255.255.0`<br>`network 10.0.0.0`<br>`broadcast 10.0.0.255`<br>Observaţi interfaţa de loopback, interfaţa WAN (eth0) ce se autoconfigurează prin DHCP şi interfaţa LAN (eth1) ce are adresa statică 10.0.0.1. |
| **dnsmasq** | Porneşte serviciul dnsmasq, serverul de DHCP şi DNS pentru reţeaua LAN. Ca şi network, doar apelează executabilul respectiv. |

| | |
|---|---|
| **iptables** | Rulează comenzi iptables pentru a configura reţeaua (filtrare de pachete şi NAT). |
| **httpd** | Porneşte serverul HTTP |
| **ntpdate** | Actualizează data şi ora sistemului prin protocolul Network Time Protocol. |
| **dropbear** | Porneşte serverul SSH |
| **smb** | Porneşte serverul Samba (pentru File Sharing) |

După ce rulează aceste scripturi, **rcS** afişază mesajul ***NGW100 Ready***, iar init rulează shell-ul pe interfaţa serială (***/bin/sh***), dând controlul utilizatorului.

## 2.3. Exerciţii

**1.** Editaţi fişierul ***/etc/modules*** pentru a încărca automat modulul g_serial, ce emulează o conexiune serială prin USB. Specificaţi argumentul ***use_acm=1***, pentru a folosi clasa USB CDCACM (Communications Device Class, Abstract Control Model). Scriptul de startup mountvirtfs a creat deja fişierul ***/dev/ttyGS0*** corespunzător, care funcţionează ca un port serial (asemenea ***/dev/ttyS0***).

Rulaţi ***/etc/init.d/S10modulesinit*** pentru a simula o reîncărcare a sistemului fără a mai aştepta rularea tutror celorlalte scripturi. Verificaţi cu ***lsmod*** şi ***dmesg|tail*** că modulul g_serial s-a încărcat.

**2.** Cuplaţi dispozitivul USB şi observaţi apariţia fişierului ***/dev/ttyACM0*** corespunzător (de exemplu cu ***ls /dev/ttyA****). Conectaţi-vă cu un terminal serial precum GtkTerm la respectivul port serial, la 9600 baud.

Pe NGW100 scrieţi ceva în ***/dev/ttyGS0*** şi observaţi cum apare în GtkTerm.

**3.** Închideţi GtkTerm. Editaţi fişierul ***/etc/inittab*** pentru a rula un shell şi pe terminalul ***/dev/ttyGS0***, nu numai pe ***/dev/ttyS0***. Aceasta presupune decomentarea unei linii date ca exemplu în fişier. Faceţi în prealabil un backup. Reporniţi NGW100 cu comanda **reboot**.

**4.** Conectaţi-vă cu GtkTerm pe ***/dev/ttyACM0*** (sau ACM1 dacă s-a schimbat cifra), la 115200 baud. Reconectaţi dispozitivul USB dacă este nevoie. Aşteptaţi obţinerea unui prompt de shell. Verificaţi funcţionarea corespunzătoare.

**5.** Conectaţi-vă prin ssh la NGW100. Scrieţi ceva în ***/dev/ttyGS0***, observând din nou apariţia textului în GtkTerm. Observaţi că textul ce apare în dreptul prompt-ului nu constituie o comandă pentru shell, ci a fost doar transmis pe acelaşi canal ca ieşirea shell-ului.

**6.** Încercaţi să citiţi din ***/dev/ttyGS0***, cu **cat**, observând că nu primiţi caracterele tastate în GtkTerm, deoarece acestea sunt preluate de shell. Daţi o comandă invalidă la shell, este posibil ca după aceea caracterele să fie preluate de cat. Închideţi **cat** cu ^C.

**7.** Listaţi conţinutul directorului ***/dev/pts***, ce conţine pseudo-terminalele. Unul din ele (probabil singurul) corespunde conexiunii prin ssh. Scrieţi (din GtkTerm) un text în ***/dev/pts/0*** şi observaţi cum apare în terminalul ssh.

**8.** Refaceţi fişierul ***/etc/modules*** (ar trebui să fie gol).

**9.** Scrieţi un script numit ***/student.sh*** care:

- încarcă modulul g_serial **use_acm=1**
- creează un director **/student** şi montează un sistem de fişiere de tipul **tmpfs** în el, **readwrite**
- creează un director **/student/readonly** şi montează un sistem **tmpfs read-only** în el (cu opţiunea **o ro**)

Un script trebuie să aibă pe prima linie numele programului ce îl interpretează, precedat de #!, adică **#!/bin/sh** . Efectuaţi **chmod +x** pe script pentru a-l face executabil. Creaţi un link simbolic numit ***/etc/init.d/S99student*** la fişierul **/student.sh.** Reporniţi sistemul şi observaţi că se comportă corect (înainte puteţi doar rula scriptul pentru a verifica funcţionarea, în scopul de a nu aştepta prea mult):

- funcţionează shell-ul prin USB
- comanda **mount** listează corect sistemele de fişiere **/student** şi **/student/readonly**
- puteţi crea fişiere în **/student** dar nu în **/student/readonly**.

**10.** Ştergeţi link-ul şi scriptul. Demontaţi sistemele de fişiere montate anterior şi ştergeţi directoarele (în ordinea corectă). Refaceţi ***/etc/inittab*** (comentaţi linia referitoare la **ttyGS0**).

**11.** Studiaţi scripturile de iniţializare şi fişierele de configurare până la expirarea timpului alocat desfăşurării laboratorului. Folosiţi resursele disponibile pe Web pentru explicaţii detaliate asupra aspectelor de interes.

# 3. Anexă

## Link-uri simbolice

O legătură simbolică (symlink) este un fişier cu rol de pointer la alt fişier. Fişierul ţintă este specificat prin calea şi numele său (spre deosebire de hard link, când e vorba de mai multe fişiere ce se leagă la aceleaşi date). Când link-ul este accesat de un program, citirea sau scrierea se face în mod transparent pe fişierul ţintă. Ştergerea unui fişier symlink şterge link-ul, nu ţinta. Crearea unui symlink:

```
ln s <target_file> <link_name>
```

## Montare

Montarea reprezintă cuplarea conţinutului unui sistem de fişiere într-un director existent. Directorul are ca suport, bineînţeles, un alt sistem de fişiere. Sistemul de fişiere rădăcină (root, /) este montat de kernel la pornirea sistemului. Celelalte sisteme se montează ulterior cu mount.

Sistemul de fişiere provine dintr-un dispozitiv, ce poate fi:

- indicat de un fişier dispozitiv din /dev
- virtual, dat de tipul sistemului de fişiere
- un dispozitiv loop, care permite montarea unei imagini a unui sistem de fişiere, stocată într-un fişier obişnuit, nu pe un dispozitiv (exemplu .iso pentru CD-uri).

Pentru a monta un sistem de fişiere de pe un dispozitiv într-un director:

```
mount t
<tip_sistem_fisiere> <dispozitiv> <director> -o <optiuni>
```

În cazul sistemelor virtuale (cum ar fi tmpfs), dispozitiv are doar rol de nume şi poate fi orice şir.

## Vizualizarea unui fişier mare

```
less <filename>
/<string> - caută <string>
q - ieşire
```

## Editarea unui fişier cu vi

```
vi <filename>
i, Insert - editare
Esc - ieşire din modul editare
:wq Enter - salvare şi ieşire
:q! Enter - ieşire fără salvare
```