Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Odsek za računarsku tehniku i računarske komunikacije

Linuks u namenskim sistemima

Uvod i razvojno okruženje

Uvod u linuks u namenskim sistemima

LINUKS U NAMENSKIM SISTEMIMA

Nastanak besplatnog softvera

- 1983, Richard Stallman, GNU projekat i koncept besplatnog softvera. Početak razvoja gcc, gdb, glibc i drugih važnih alata
- 1991, Linus Torvalds, Linuks kernel projekat, kernel Unix-like operativnog sistema. Zajedno sa GNU softverom i mongim drugim komponentama otvorenog koda: potpuno besplatan operativni sistem, GNU/Linuks
- 1995, Linuks je sve popularniji na sistemima sa poslužiocem
- 2000, Linuks je sve popularniji u ugrađenim (namenskim) sistemima
- 2008, Linuks je sve popularniji u prenosivim uređajima
- 2010, Linuks je sve popularniji na telefonima.

Besplatan softver?

- Program se smatra **besplatnim** kada njegova licenca nudi svim korisnicima sledeće **4** mogućnosti:
  - Slobodu da se softver koristi (pokreće)u bilo koju svrhu
  - Slobodu da se softver izučava i menja
  - Slobodu da se redistribuiraju kopije
  - Slobodu da se distribuiraju izmenjene verzije
- Ove slobode su date za kako komercijalnu tako i nekomercijalnu upotrebu
- Podrazumevaju postojanje izvornog koda, te da softver može biti modifikovan i distribuiran mušterijama
- **Veoma pogodno za namenske sisteme**

Šta je Linuks u namenskim sistemima?

- Linuks u namenskim sistemima je upotreba Linuks jezgra (kernela) i različitih komponenti otvorenog koda u namenskim sistemima

Zašto linuks u namenskim sistemima?

Prednosti Linuksa i otvorenog koda u namenskim sistemima

UVOD U LINUKS U NAMENSKIM SISTEMIMA

Ponovna upotreba komponenti

- Glavna prednost Linuksa i otvorenog koda u namenskim sistemima je **mogućnost** ponovnog korišćenja komponenti
- Ekosistem otvorenog koda već nudi mnoge komponente za uobičajene zahteve, od hardverske podrške do mrežnih protokola, multimedije, grafike, kriptografskih biblioteka, itd.
- Čim se dovoljno raširi upotreba hardverskog uređaja, protokola ili karakteristike, postoji velika šansa da je komponenta otvorenog koda koja je podržava već dostupna
- Omogućava brz dizajn i razvoj komplikovanih komponenti.
- Niko ne bi trebalo da ponovo razvija još jedan operativni sistem, TCP/IP stek, USB stek ili druge biblioteke iz grafičke grupe alata.
- Dozvoljava fokusiranje novu vrednost produkta.

Mala cena

- Besplatan softver može da se umnoži na željenom broju uređaja, potpuno besplatno
- Ukoliko vaš namenski sistem koristi isključivo besplatan softver, možete svesti i cenu softverskih licenci na 0. Čak su i alati za razvoj besplatni, osim ukoliko odaberete komercijalni Linuks za namenske sisteme
- Dozvoljava postojanje višeg budžeta za hardver, odnosno povećanje veština i znanja u kompaniji.

Potpuna kontrola

- Sa otvorenim kodom, imate izvorni kod svih komponenti u sistemu
- Dozvoljava neograničene modifikacije, promene, podešavanja, debagovanje, optimizacije, u neograničenom vremenskom periodu
- Bez zaključavanja ili zavisnosti od "third-party" prodavaca
  - Da bi se ovo ispoštovalo, komponente koje ne pripadaju otvorenom kodu moraju da se izbegnu prilikom dizajna i razvoja
- **Dozvoljava punu kontrolu nad softverom u vašem sistemu**

Kvalitet

- Mnoge komponente otvorenog koda imaju široku upotrebu, na milionima sistema
- Često imaju viši kvalitet od komponenti koje se mogu napraviti unutar kompanije, pa čak i od izvornog prodavca
- Naravno, nisu sve komponente otvorenog koda kvalitetne, ali one sa širom upotrebom jesu.
- Omogućava dizajniranje vašeg sistema sa visokokvalitetnim komponentama od samog početka

Olakšava testiranje novih funkcionalnosti

- S obzirom da je otvoreni kod dosupan besplatno, lako je doći do nekog softvera i isprobati ga
- Omogućava jednostavno proučavanje više opcija dok pravite neki odabir
- Mnogo jednostavnije od kupovine i demonstracionih procedra potrebnih za većinu izvornih produkata
- **Omogućava jednostavni pregled novih mogućnosti i rešenja**

Podrška u zajednici

- Programske komponente otvorenog koda razvija zajednica developera i korisnika
- Ova zajednica nudi podršku visokog kvaliteta: možete direktno da kontaktirate glavne developere komponenti koje koristite. Pritom, verovatnoća da ćete dobiti odgovor ne zavisi od toga za koju kopaniju radite.
- Često je ova podrška bolja od tradicionalne, ali zahteva razumevanje kako zajednica funkcioniše da bi se mogučnosti podrške zajednice adekvatno iskoristile
- **Omogućava ubrzavanje rešavanja problema kada razvijate vaš sistem**

Preuzimanje uloge u zajednici

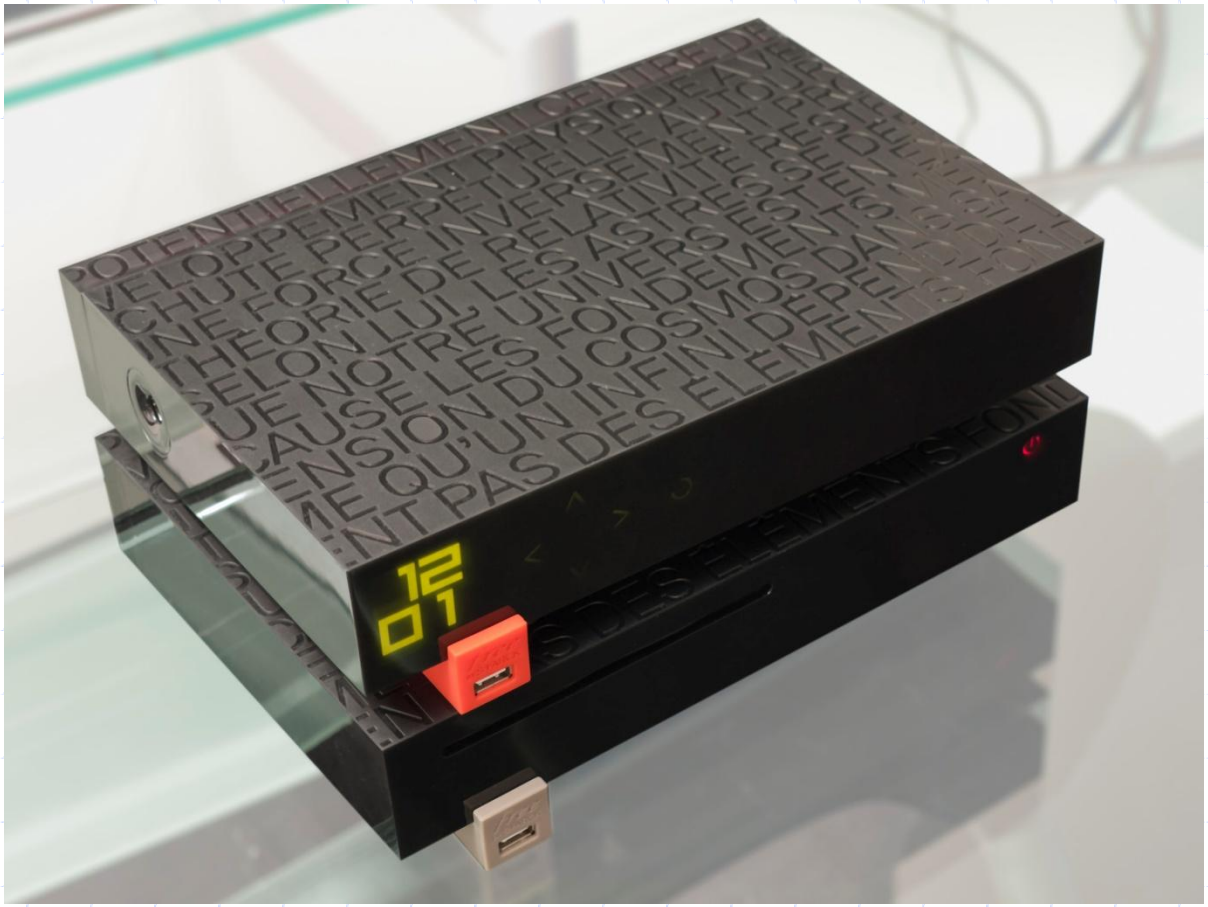
- Postoji mogućnost preuzimanja uloge u zajednici razvoja za neku komponentu korišćenu u namenskim sistemima: prijava greške, testiranje novih verzija ili svojstva, zakrpa koje popravljaju greške ili dodaju nova svojstva, itd.
- Najčešće komponente otvorenog koda nisu glavna vrednost proizvoda, već je to interes svakoga da uzvrati doprinos.
- Za *inženjere* to predstavlja: veoma **motivišući** način da budu prepoznati izvan kompanije, komunikaciju sa drugima u datoj oblasti, **otvaranje novih mogućnosti**, itd.
- Za *menadžere*: **motivišući faktor** za inženjere, omogućava kompaniji da bude prepoznata u zajednici otvorenog koda i stoga lakše dođe do podrške i da bude **atraktivnija** developerima otvorenog koda.

Sistemi sa Linuksom

Nekoliko primera namenskih sistema na kojima se nalazi Linuks

UVOD U LINUKS U NAMENSKIM SISTEMIMA

Personalni ruteri



Televizori

POS terminali

Laserska mašina za sečenje

Mašine za vinogradarstvo

Namenski hardver za Linuks sisteme

UVOD U LINUKS U NAMENSKIM SISTEMIMA

Procesori i arhitekture (1/2)

- Linuks kernel i većina drugih komponenti koje zavise od arhitekture podržavaju širok spektar 32-bitnih i 64-bitnih arhitektura
  - x86 i x86-64, kako na PC platformama, tako i u namenskim sistemima (multimedijalnim, industrijskim)
  - ARM, sa stotinama različitih SoC (multimedijalnim, industrijskim)
  - PowerPC (uglavnom u realnom vremenu, industrijska primena)
  - MIPS (uglavnom mrežna primena)
  - SuperH (uglavnom set top box i multimedijalna primena)

Procesori i arhitekture (2/2)

- Blackfin (DSP architecture)
- Microblaze (softverski procesor za Xilinx FPGA)
- Coldfire, SCore, Tile, Xtensa, Cris, FRV, AVR32, M32R
- Podržane su arhitekture sa i bez MMU, iako arhitekture bez MMU imaju određena ograničenja
- Linuks nije napravljen za male mikrokontrolere.
- Osim alata za prevođenje, bootloader-a i kernela, sve ostale komponente su generalno nezavisne od arhitekture

Ram i prostor za skladištenje

- **RAM**: osnovna verzija Linuks sistema može da radi sa svega 8 MB RAM, ali malo realističniji sistem obično zahteva bar 32 MB RAM memorije. Zavisno od tipa i veličine aplikacija.
- **Prostor za skladištenje**: osnovna verzija Linuks sistema može da radi sa svega 4 MB prostora, ali je obično potrebno više.
  - Podržan je Flash tip, uključujući NAND i NOR flash, sa specifičnim sistemima datoteka
  - Podržano je blokovsko skladištenje uključujući SD/MMC kartice i eMMC
- Nije neophodno biti previše restriktivan po pitanju količine RAM memorije ili memorije za skladištenje: na ovom nivou imamo fleksibilnost da iskoristimo sve dostupne komponente.

Komunikacija

- Linuks kernel ima podršku za mnoge ustaljene komunikacione magistrale
  - I2C
  - SPI
  - CAN
  - 1-wire
  - SDIO
  - USB
- Takođe i punu podršku za mrežu:
  - Ethernet, Wifi, Bluetooth, CAN, itd.
  - IPv4, IPv6, TCP, UDP, itd.
  - Firewall, napredno rutiranje, multikast

Tipovi hardverskih platformi

- **Platforme za evaluaciju** od proizvođača SoC-a. Najčešće su skupe, ali poseduju mnoge ugrađene periferije. Generalno su neprikladne za krajnje proizvode.
- **Komponenta u vidu modula**, pločica samo sa CPU/RAM/flash i još ponekom komponentom i konektorima za pristup svim drugim periferijama. Može se koristiti za pravljenje krajnjih proizvoda u malim ili srednjim serijama.
- **Platforme za razvoj u zajednici**, popularizuju određeni SoC i čine ga lako dostupnim. Odmah su spremni za upotrebu uz dobru podršku i nisku cenu, ali često imaju manje periferija od platformi za evaluaciju. Donekle mogu biti korišćene i za krajnje proizvode.
- **Platforme po meri**. Šematski prikazi i platforme za evaluaciju ili razvojne platforme su sve češće besplatno dostupni, što olakšava razvoj platformi po meri.

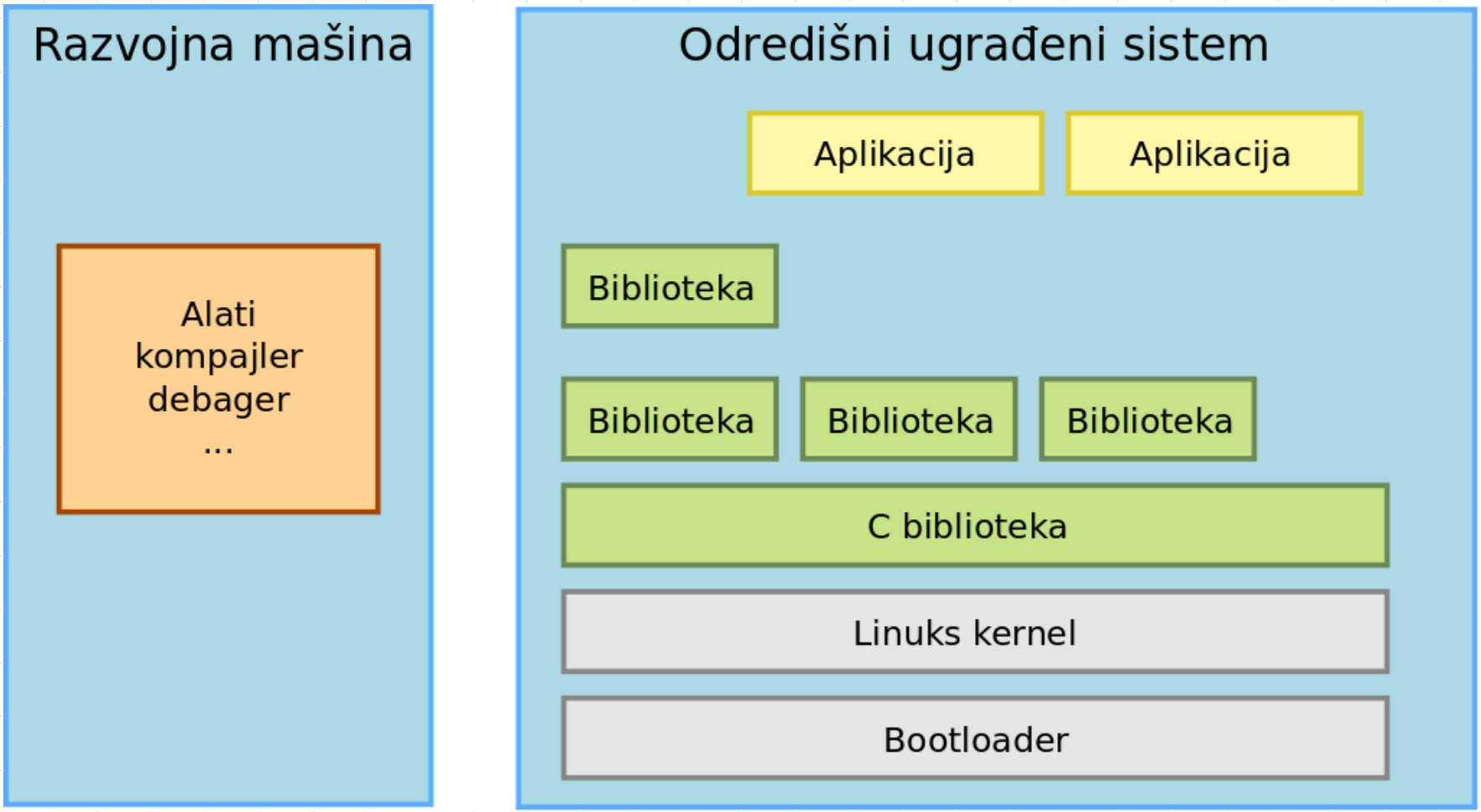
Kriterijumi za odabir hardvera

- Uverite se da je hardver koji planirate da koristite već podržan od strane Linuks kernela, da ima bootloader sa otvorenim kodom, pre svega SoC koji ciljate.
- Postojanje podrške u zvaničnim verzijama projekata (kernel, bootloader) je daleko bolje: kavalitetnije i dostupne su nove verzije.
- Neki proizvođači SoC i/ili ploča ne doprinose svojim izmenama glavnom Linuks kernelu. Pitajte ih da to učine ili koristite drugi hardver ako možete. Dobra mera je proveriti razliku između njihovog i zvaničnog kernela.
- **Postojaće velika razlika, u vremenu i ceni razvoja, između hardvera sa korektnom podrškom u zvaničnom Linuks kernelu i hardvera sa slabom podrškom**

Arhitekture za namenske Linuks sisteme

UVOD U LINUKS U NAMENSKIM SISTEMIMA

Globalna arhitektura



Programske komponente

- Alat za prevođenje za drugu platformu (unakrsno prevođenje, Cross-compilation)
  - Kompajler koji se pokreće na razvojnoj mašini, ali generiše kod za odredišnu
- Bootloader
  - Pokreće ga hardver, odgovoran je za izvršavanje kernela
- Linuks Kernel
  - Sadrži proces menadžment i memorijski menadžment, mrežni stek, rukovaoce uređajima i nudi servise za aplikacije u korisničkom prostoru
- C biblioteka
  - Sprega između kernela i aplikacija u korisničkom prostoru
- Biblioteke i aplikacije
  - Dobijene od drugih ili napravljene “u kući”

Rad na Linuksu za namenske sisteme

- Nekoliko jasnih zadataka je potrebno za razvoj Linuksa za određeni proizvod:
  - **Razvoj Board Support Package**
    - BSP sadrži bootloader i kernel sa odgovarajućim rukovaocima uređajima za ciljni hardver
    - Ovo je ujedno i cilj ovog kursa
  - **Integracija sistema**
    - Integracija svih komponenti u sistem: bootloader-a, kernela, tuđih biblioteka i aplikacija kao i onih razvijenih "u kući"
    - Ovo je takođe cilj ovog kursa
  - **Razvoj aplikacija**
    - Regularne Linuks aplikacije, ali koriste posebno odabrane biblioteke

Razvojno okruženje Linuksa u namenskim sistemima

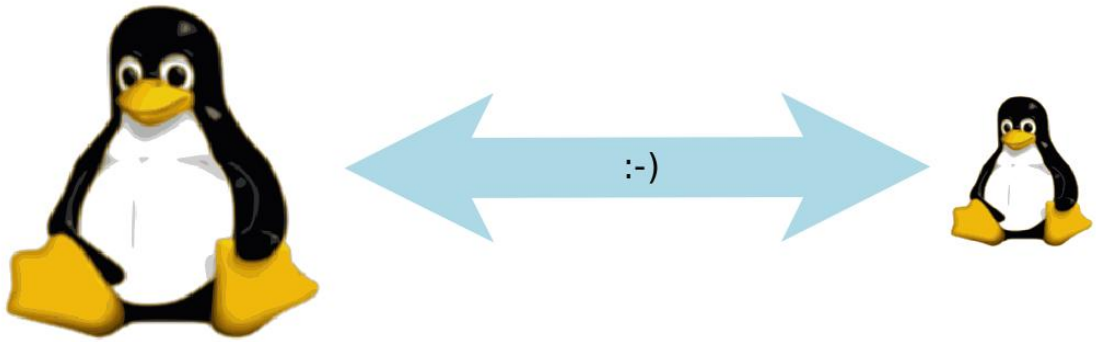
LINUKS U NAMENSKIM SISTEMIMA

Rešenja Linuksa u namenskim sistemima

- Dva načina za prebacivanje na Linuks za namenske sisteme
  - Korišćenje **rešenja i podrške od proizvođača** kao što su MontaVista, Wind River ili TimeSys. Ova rešenja dolaze sa razvojnim alatima i okruženjem. Predstavljaju kombinaciju komponenti otvorenog koda i vlasničkih alata.
  - Korišćenje **rešenja zajednice**. Potpuno su otvorena i podržana od strane zajednice.
- Treba pribegavati korišćenju rešenja zajednice otvorenog koda.
  - Znajući koncepte, ni prelazak na rešenja proizvođača neće biti problem

OS za razvoj Linuksa

- Preporučuje se upotreba Linuksa kao desktop operativnog sistema za razvoj Linuksa za namenske sisteme iz više razloga.
- Svi alati zajednice su razvijeni i dizajnirani za pokretanje na Linuksu. Pokušaj korišćenja istih na drugom OS (Windows, Mac, OS X) bi doveo do problema, i njihova upotreba na ovim sistemima u principu nije podržana od strane zajednice.
- Kako je Linuks i na namenskom uređaju, svo znanje stečeno korišćenjem Linuksa na desktop računaru se može slično primeniti na namenski uređaj.



Desktop Linuks distribucija

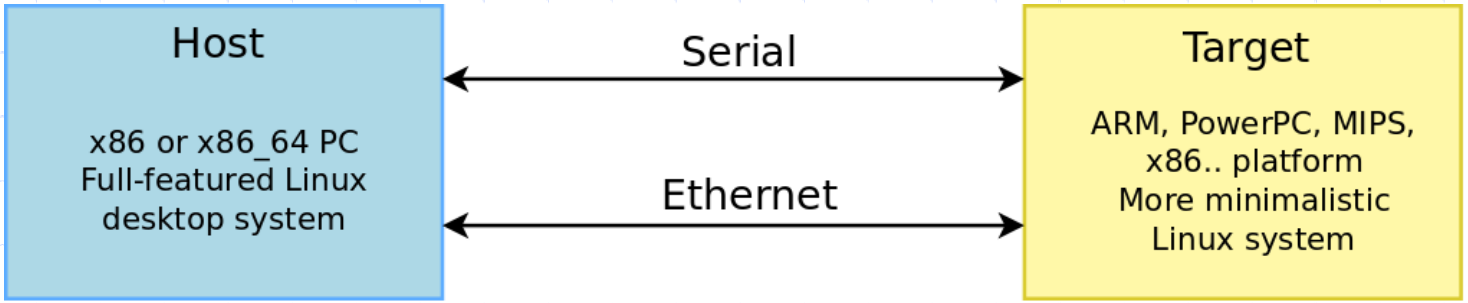
- **Bilo koja dobra i dovoljno sveža Linuks desktop distribucija** se može koristiti na razvojnoj radnoj stanici
  - Ubuntu, Debian, Fedora, openSUSE, Red Hat, itd.
- Na kursu se koristi Ubuntu, kao **često upotrebljavana i jednostavna** desktop Linuks distribucija
- Ubuntu podešavanja na virtuelnim mašinama su namerno ostavljena nedirnuta nakon uobičajenog instalacionog procesa. Učenje o Linuksu za namenske sisteme takođe obuhvata i učenje o alatima potrebnim na razvojnoj radnoj stanici

Linuks korenski i ostali korisnici

- Linuks je operativni sistem sa više korisnika
  - **Korenski korisnik je administrator** i može da sprovodi privilegovane operacije kao što su: mauntovanje sistema datoteka, konfigurisanje mreže, stvaranje datoteka uređaja, promena konfiguracije sistema, instaliranje ili uklanjanje paketa, itd.
  - Svi **ostali su neprivilegovani** korisnici i ne mogu da sprovode pomenute operacije predviđene za administratora
- Na Ubuntu sistemu nije moguće logovanje kao `root` korisnik već samo kao regularan korisnik.
- Sistem je konfigurisan tako da je prvom stvorenom nalogu omogućeno da pokreće privilegovane operacije kroz program pod nazivom `sudo`.
  - Primer: `sudo mount /dev/sda2 /mnt/disk`

Razvojna i odredišna platforma

- Prilikom razvoja namenskog uređaja, uvek postoji podela između
  - *host*, razvojne radne stanice, koja je tipično moćan PC
  - *target*, koji je namenski sistem koji se razvija
- Mogu biti povezani različitim sredstvima:
  - Skoro uvek serijskom linijom u svrhe debagovanja
  - Često Eternet vezom
  - Ponekad JTAG spregom za debagovanje na niskom nivou



Program za komunikaciju preko serijske linije

- Izuzetno važan alat za razvoj namenskih uređaja je program za komunikaciju preko serijske linije, poput program HyperTerminal u OS Windows.
- Postoje razne opcije u Linuksu: Minicom, Picocom, Gtkterm, Putty, itd.
- Na ovom kursu se predlaže picocom
  - Instalacija pomoću `sudo apt-get install picocom`
  - Pokretanje komandom `picocom -b BAUD_RATE /dev/SERIAL_DEVICE`
  - Prekid rada komandom `Control-A Control-X`
- `SERIAL_DEVICE` je tipično
  - `ttyUSBx` za USB na serijsku vezu
  - `ttySx` za fizičke serijske portove

Saveti za komandnu liniju

- Korišćenje komandne linije je mandatorno zbog mnogih operacija potrebnih za razvoj Linuksa za namenske uređaje
- Veoma je moćan način za interakciju sa sistemom, te može da uštedi dosta vremena.
- Možete koristiti nekoliko tabova u Gnome Terminalu
- Setite se da imate relativnu putanju (npr: `../../linux`) in addition to absolute paths (npr: /home/user)
- U šelu, pritisnite [Control] [r], i tada će se komanda pretraživati u istoriji komandi. Pritisnite ponovo [Control] [r] da biste pretražili istoriju unazad.
- Možete da prenesete putanje direktno sa drugih menadžera u terminal pomoću drag-and-drop

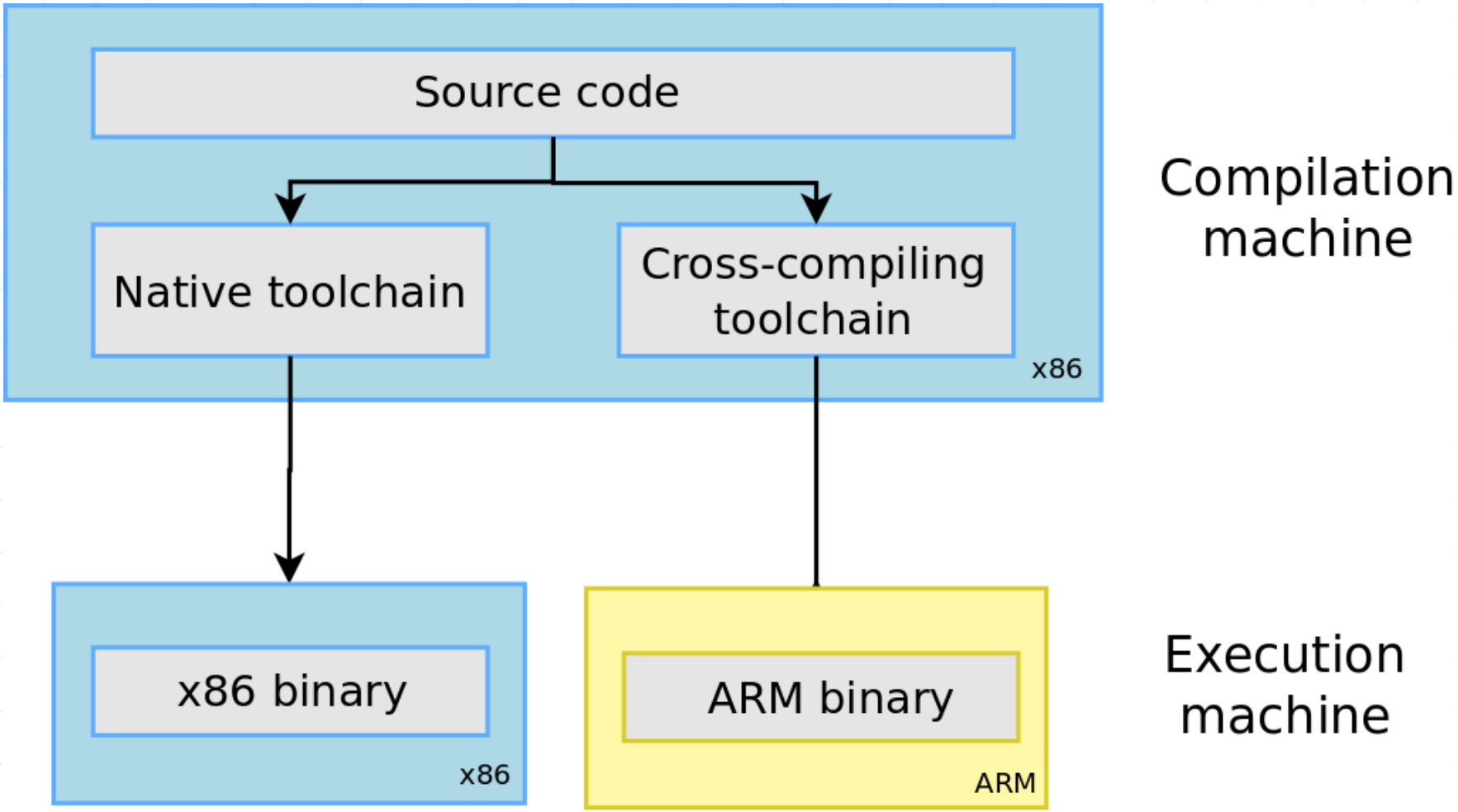
Alati za unakrsno prevođenje - definicija i komponente

RAZVOJNO OKRUŽENJE LINUKSA U NAMENSKIM SISTEMIMA

Definicija (1/2)

- Uobičajeni alati za razvoj dostupni na GNU/Linuks radnoj stanici su **ugrađeni alati za prevođenje**
- Ovi alati za prevođenje se pokreću na radnoj stanici i generišu kod za radnu stanicu, najčešće je to x86.
- Za razvoj namenskih sistema, najčešće je nemoguće ili neinteresantno korišćenje ugrađenih alata za prevođenje
  - Odredišna platforma ima preveliku restrikciju po pitanju prostora za skladištenje i/ili memorije.
  - Odredišna platforma je spora u poređenju sa radnom stanicom
  - Verovatno ne želite da instalirate sve razvojne alate na odredišnu platformu.
- Stoga su **alati za unakrsno prevođenje** najčešće korišćeni. Oni se pokreću na radnoj stanici, ali generišu kod za odredišnu platformu.

Definicija (2/2)



Mašine u bild procedurama

- Tri mašine se moraju razdvojiti kada je reč o pravljenju opisanog alata za prevođenje
  - **bild** mašina, gde se pravi alat za unakrsno prevođenje.
  - **host** mašina, gde se izvršava alat za unakrsno prevođenje i pravi binarnu, izvršnu datoteku za odredišnu platformu.
  - **target** mašina, gde se izvršavaju binarne datoteke napravljene od strane alata za unakrsno prevođenje.
- Četiri uobičajena bild tipa postoje za alate za prevođenje

Različite bild procedure alata za prevođenje

Native build

used to build the normal gcc of a workstation

Cross build

used to build a toolchain that runs on your workstation but generates binaries for the target

The most common case in embedded development

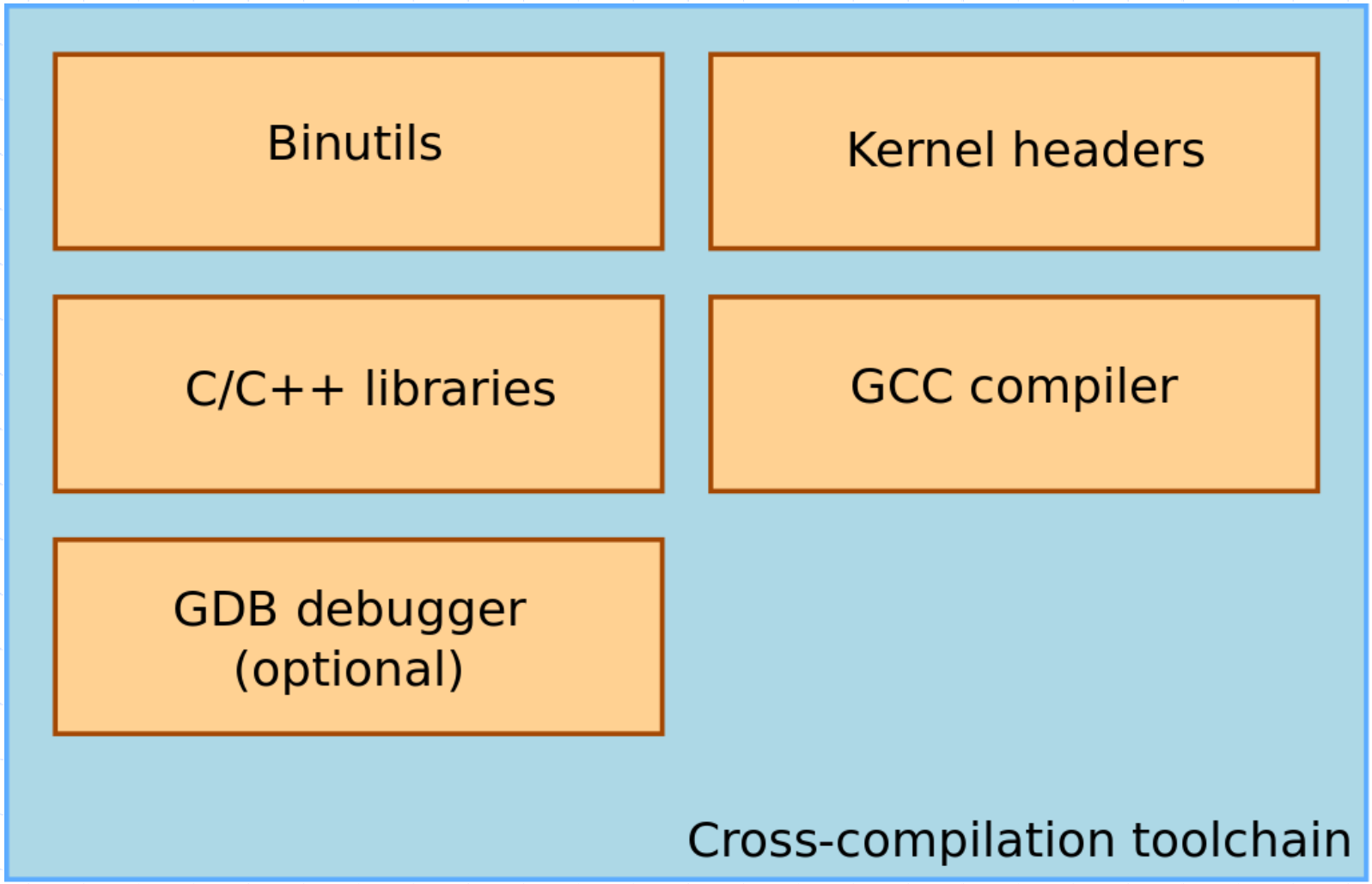
Cross-native build

used to build a toolchain that runs on your target and generates binaries for the target

Canadian build

used to build on architecture A a toolchain that runs on architecture B and generates binaries for architecture C

Komponente

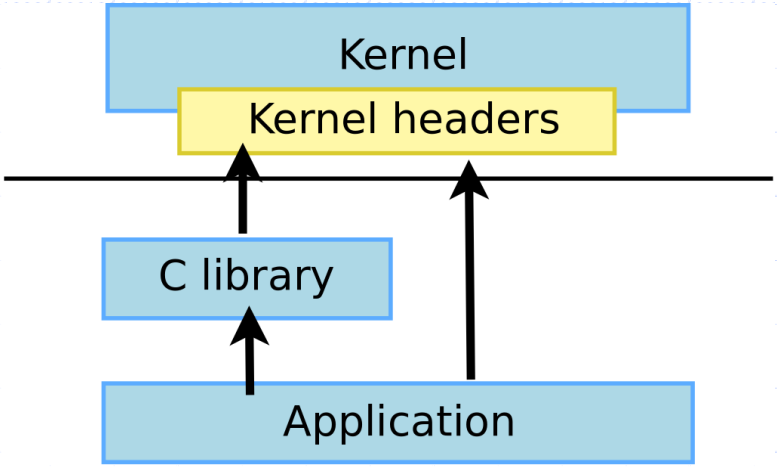

Binutils
Kernel headers
C/C++ libraries
GCC compiler
GDB debugger
(optional)
Cross-compilation toolchain

Binutils

- **Binutils** je skup alata za generisanje i manipulisanje binarnim datotekama za zadatu CPU arhitekturu
  - `as`, asembler, koji generiše binarnu datoteku na osnovu asemblerskog izvornog koda
  - `ld`, povezivač
  - `ar`, `ranlib`, za generisanje `.a` arhiva, korišćenih za biblioteke
  - `objdump`, `readelf`, `size`, `nm`, `strings`, za inspekciju binarnih datoteka. Veoma koristan alat za analizu!
  - `Strip`, za odbacivanje dela binarnih datoteka koje se koriste samo za debagovanje (umanjujući im veličinu).
- http://www.gnu.org/software/binutils/
- GPL licenca

Kernel zaglavlja (1/3)

- C biblioteka i prevedeni programi zahtevaju interakciju sa kernelom
  - Dostupni sistemski pozivi i njihovi brojevi
  - Definicije konstanti
  - Strukture podataka, itd.
- Stoga, prevođenje C biblioteke zahteva zaglavlja kernela i mnoge aplikacije ih takođe zahtevaju.
- Dostupno u `<linux/...>` i `<asm/...>` i još nekim drugim direktorijumima koji odgovaraju onima vidljivim u `include/` u izvornom kodu kernela



Kernel zaglavlja (2/3)

- Brojevi sistemskih poziva u <asm/unistd.h>

```
#define __NR_exit          1
#define __NR_fork          2
#define __NR_read          3
```

- definicije konstanti, u <asm-generic/fcntl.h>, uključeno sa putanje <asm/fcntl.h>, uključeno sa putanje <linux/fcntl.h>

```
#define O_RDWR 00000002
```

- Strukture podataka, u <asm/stat.h>

```
struct stat {
        unsigned long st_dev;
        unsigned long st_ino;
        [...]
};
```

Kernel zaglavlja (3/3)

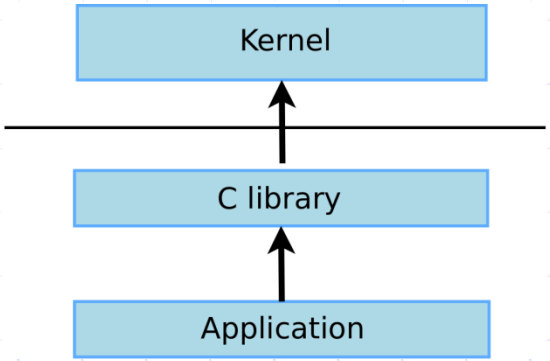
- Sprega (ABI) kernela i korisničkog prostora je **kompatibilna sa starijim verzijama**
  - Binarne datoteke generisane alatom za prevođenje uz korišćenje kernel zaglavlja starijih od pokrenutog kernela će raditi bez problema, ali neće moći da koriste nove sistemske pozive, stručture podataka, itd.
  - Binarne datoteke generisane alatom za prevođenje uz korišćenje kernel zaglavlja novijih od pokrenutog kernela mogu raditi ukoliko ne koriste skorašnje funkcionalnosti, inače bi se neuspšno izvršile
  - Korišćenje poslednjih zaglavlja Linuks kernela nije neophodno osim ako je pristup novim kernel funkcionalnostima potreban.
- Kernel zaglavlja se mogu raspakovati iz kernel izvornog koda, korišćenjem `headers_install` kernel Makefile ciljem.

GCC

- GNU Compiler Collection, poznati besplatan kompajler
- Prevodi C, C++, Ada, Fortran, Java, Objective-C, Objective-C++, i generiše kod za velik broj procesorskih arhitektura, uključujući ARM, AVR, Blackfin, CRIS, FRV, M32, MIPS, MN10300, PowerPC, SH, v850, i386, x86_64, IA64, Xtensa, itd.
- http://gcc.gnu.org/
- Dostupan pod GPL licencom, biblioteke pod LGPL licencom.

C biblioteka

- C biblioteka je važna komponenta Linuks sistema
  - Sprega između aplikacija i kernela
  - Obezbeđuje dobro poznatu standardnu C spregu za olakšani razvoj aplikacija
- Nekoliko C biblioteka je dostupno: *glibc*, *uClibc*, *musl*, *dietlibc*, *newlib*, itd.
- Izbor C biblioteke mora biti napravljen u vreme generisanja alata za unakrsno prevođenje, jer se GCC kompajler prevodi nasuprot specifičnoj C biblioteci.



glibc

- Licenca: LGPL
- C biblioteka sa GNU projekta
- Dizajnirana za performansu, usklađenost sa standardima i portabilnost
- Nalazi se na svim GNU / Linuks host sistemima
- Naravno, aktivno se održava
- Podrazumevano, prilično je velika za male namenske sisteme: približno 2.5 MB na ARM (version 2.9 - `libc`: 1.5 MB, `libm`: 750 KB)
- Ipak, neke funkcionalnosti koje nisu potrebne u namenskim sistemima se mogu isključiti u konfiguraciji (uvučeni su iz starog *eglibc* projekta).
- http://www.gnu.org/software/libc/

uClibc-ng (1/2)

- http://uclibc-ng.org/
- Nastavak starog uClibc projekta
- Licenca: LGPL
- Mala C biblioteka za male namenske sisteme
  - Visoka konfigurabilnost: mnoge funkcionalnosti se mogu uključiti/isključiti kroz menuconfig spregu
  - Podržava većinu namenskih arhitektura
  - Podržava arhitekture bez MMU (ARM Cortex-M, Blackfin, itd.)
  - Ne garantuje binarnu kompatibilnost. Može da zahteva rekompajliranje aplikacija kada se promeni konfiguracija biblioteke.
  - Fokusira se na veličinu biblioteke pre nego na performansu
  - Malo vreme prevođenja

uClibc-ng (2/2)

- Većina aplikacija se prevodi sa uClibc-ng. Ovo se odnosi na sve aplikacije u namenskim sistemima.
- Veličina (arm): 3.5x manja od glibc!
  - uClibc-ng 1.0.14: približno 716kB (libuClibc: 282kB, libm: 73kB)
  - glibc 2.22: približno 2.5 MB
- Neke funkcionalnosti nisu dostupne ili su ograničene: npr. nasleđivanje prioriteta mutex-a.
- Koristi se na velikom broj namenskih proizvoda koji su u produkciji, uključujući uređaje iz grupe potrošačke elektronike.

Poređenje uClibc-ng i glibc

- Veličina izvršne datoteke poređena na ARM, testirana sa *glibc* 2.22 i *uClibc-ng* 1.0.14

C program	Prevedeno dinamički sa glibc	Prevedeno dinamički sa uClibc	Prevedeno statički sa glibc	Prevedeno statički sa uClibc
Običan "hello world"	2.7 KB	2.5 KB	479 KB	33.4 KB
Busybox	503 KB	664 KB	1206 KB	818 KB

musl C biblioteka

- http://www.musl-libc.org/
  - Veoma mala, brza i jednostavna biblioteka za namenske sisteme
  - Napravljena dok je razvoj uClibc bio obustavljen
  - U suštini, odlična je za pravljenje malih izvršnih datoteka sa statičkim uvezivanjem
  - Permissive license (MIT)
  - Poređenje sa drugim C bibliotekama: http://www.etalabs.net/compare_libcs.html
  - Podržana od strane bild sistema kao što je Buildroot

Ostale manje C biblioteke

- Razvijeno je još nekoliko manjih C biblioteka, ali nijedna od njih nema za cilj omogućavanje prevođenja velikih postojećih aplikacija
- Mogu da se koriste za relativno jednostane program, tipično za pravljenje veoma malih izvršnih datoteka sa statičkim uvezivanjem i pokretanje istih u veoma malim korenskim sistemima datoteka.
- Mogućnosti:
  - Dietlibc, http://fefe.de/dietlibc/. Približno 70 KB.
  - Newlib, http://sourceware.org/newlib/
  - Klibc, http://www.kernel.org/pub/linux/libs/klibc/, dizajnirana za *initramfs* ili *initrd* za vreme boot-a.

Opcije alata za prevođenje

RAZVOJNO OKRUŽENJE LINUKSA U NAMENSKIM SISTEMIMA

ABI

- Kada se pravi alat za prevođenje, ABI, koji se koristi za generisanje binarnih datoteka, mora biti definisan
- ABI, *Application Binary Interface*, definiše pozivne konvencije (kako se prosleđuju argumenti funkcija, kako se prosleđuje povratna vrednost i kako se prave sistemski pozivi) i organizacije struktura (poravnanja i sl)
- Sve binarne datoteke u sistemu moraju biti prevedene korišćenjem istog ABI, i kernel mora da ume da rukuje istim ABI
- Na ARM arhitekturi, postoje dva glavna ABI: *OABI* i *EABI*
  - U novije vreme se koristi *EABI*
- Na MIPS arhitekturi, postoji nekoliko ABI: *o32*, *o64*, *n32*, *n64*
- http://en.wikipedia.org/wiki/Application_Binary_Interface

Podrška za račun u pokretnom zarezu

- Neki procesori imaju jedinicu za račun u pokretnom (FPU) zarezu, dok drugi nemaju
  - Npr, mnogi ARMv4 i ARMv5 procesori nemaju FPU. Od ARMv7, VFP jedinica je postala mandatorna.
- Za procesore koji imaju FPU, alat za prevođenje mora da generiše *hard float* kod, da bi koristio instrukcije u pokretnom zarezu direktno.
- Za procesore koji nemaju FPU, postoje dva rešenja:
  - Generisati *hard float* kod i osloniti se na kernel koji će emulirati instrukcije u pokretnom zarezu. Ovo je veoma sporo.
  - Generisati *soft float* kod, tako da se umesto generisanja instrukcija u pokretnom zarezu, generišu pozivi biblioteka iz korisničkog prostora
- Odluka mora biti doneta u vreme konfiguracije alata za prevođenje
- Moguće je konfigurisati koja će FPU biti korišćena

CPU optimizacione zastavice

- Skup alata za unakrsno prevođenje je specifičan za svaku CPU arhitekturu (ARM, x86, MIPS, PowerPC)
- Međutim, uz `-march=`, `-mcpu=`, `-mtune=` opcije, moguće je preciznije odabrati tip ciljnog procesora
  - Npr, `-march=armv7 -mcpu=cortex-a8`
- U vreme prevođenja alata za prevođenje, mogu se odabrati vrednosti. One se koriste:
  - Kao podrazumevane vrednosti za alate za unakrsno prevođenje, kada druge vrenosti za `-march`, `-mcpu`, `-mtune` opcije nisu prosleđen
  - Za prevođenje C biblioteke
- Čak i ako je C biblioteka prevedena za armv5t, to ne sprečava prevođenje drugih programa za armv7

Obezbeđivanje alata za prevođenje

RAZVOJNO OKRUŽENJE LINUKSA U NAMENSKIM SISTEMIMA

Pravljenje alata za prevođenje

- Pravljenje alata za unakrsno prevođenje je težak i naporan zadatak! Može da traje danima ili nedeljama!
  - Treba naučiti dosta detalja: mnogo komponenti treba prevoditi, komplikovane konfiguracije
  - Mnogo odluka je potrebno napraviti (vezija C biblioteke, ABI, floating point mehanizmi, verzije komponenti, itd)
  - Potrebna su kernel zaglavlja i izvorni kod C biblioteka
  - Potrebno je poznavanje trenutnih `gcc` problema i zakrpa na ciljnoj platformi
  - Korisno je za upoznavanje sa alatima za konfiguraciju i prevođenje
  - Pogledajte *Crosstool-NG* `docs/` direktorijum za detalje o tome kako se pravi alat za prevođenje.

Preuzmite preveden alat za prevođenje

- Mnogi biraju ovo rešenje
  - Prednost: najjednostavnije i najzgodnije rešenje
  - Mane: ne postoji mogućnost za fina podešavanja alata prema potrebama
- Uverite se da alat koji ste pronašli odgovara zahtevima: CPU, endianness, C biblioteka, verzije komponenti, ABI, softverska ili harverska jedinica u pokretnom zarezu, itd.
- Mogući izbor:
  - Alati sadržani u paketima vaše distribucije. Ubuntu primer:

```
sudo apt-get install gcc-arm-linux-gnueabi
sudo apt-get install gcc-arm-linux-gnueabihf
```

  - Sourcery CodeBench alati sada podržavaju samo MIPS, NIOS-II, AMD64 i Hexagon. Stare verzije sa podrškom za ARM su i dalje dostupni kroz bild sisteme (Buildroot...)
  - Alat obezbeđen od strane proizvođača hardvera.

Uslužni programi za pravljenje alata za prevođenje (1/3)

- Drugo rešenje je korišćenje usložnog programa za **automatizaciju procesa pravljenja alata za prevođenje**
  - Ista prednost kao i kod korišćenja prevedenog alata: ne morate da se zamarate oko detalja u procesu prevođenja.
  - Ali takođe nudi veću fleksibilnost po pitanju konfiguracije alata za prevođenje, odabira verzije neke komponente, itd.
  - Obično sadrže i nekoliko zakrpa za popravljanje poznatih problema sa raznim komponentama na nekoj arhitekturi.
  - Više alata po istom principu: šel skripte ili Makefile automatski dobavljaju arhive, raspakuju ih, konfigurišu, prevode i instaliraju različite komponente

Uslužni programi za pravljenje alata za prevođenje (2/3)

- Crosstool-ng
  - Zamena za stariji Crosstool program, sa konfiguracionim sistemom nalik na menuconfig
  - Razne funkcionalnosti: podržava uClibc, glibc, musl, hard i soft float, mnoge arhitekture
  - Aktivno se održava
  - http://crosstool-ng.org/

Uslužni programi za pravljenje alata za prevođenje (3/3)

- Mnogi sistemi za pravljenje korenskog sistema datoteka dozvoljavaju i pravljenje alata za prevođenje
  - **Buildroot**
    - Makefile-baziran. Može da napravi (e)glibc, uClibc i musl bazirane alate za širok spektar arhitektura.
    - http://www.buildroot.net
  - **PTXdist**
    - Makefile-bazirani, uClibc ili glibc, održavan pre svega od strane *Pengutronix*
    - http://pengutronix.de/software/ptxdist/
  - **OpenEmbedded** / **Yocto**
    - Komplikovaniji bild sitem pun opcija
    - http://www.openembedded.org/
    - https://www.yoctoproject.org/

Crosstool-NG: installation and usage

- Instalacija Crosstool-NG se može uraditi na sistemu ili samo lokalno u direktorijumu gde je izvorni kod. Za lokalnu instalaciju:

```
./configure --enable-local
make
make install
```

- Neki primeri konfiguracija za različite arhitekture su dostupni među primerima, mogu se izlistati pomoću:

```
./ct-ng list-samples
```

- Za učitavanje primera konfiguracije:

```
./ct-ng <sample-name>
```

- Za promenu konfiguracije:

```
./ct-ng menuconfig
```

- Za prevođenje alata:

```
./ct-ng build
```

Sadržaj alata za prevođenje

- Izvršne datoteke alata za unakrsno prevođenje, u `bin/`
  - Ovaj direktorijum treba dodati u `PATH` varijablu za lakše korišćenje alata za unakrsno prevođenje
- Jedan ili više *sysroot*, svaki sadrži
  - C biblioteku i povezane biblioteke, prevedene za odredište
  - Zaglavlja C biblioteke i kernela
- Postoji po jedan *sysroot* za svaku varijantu: alat može biti *multilib* ukoliko postoji nekoliko kopija C biblioteke sa različitim konfiguracijama (npr. ARMv4T, ARMv5T, itd)
  - Stari CodeSourcery ARM alati su bili multilib, sysroot-ovi u:
    `arm-none-linux-gnueabi/libc/`,
    `arm-none-linux-gnueabi/libc/armv4t`,
    `arm-none-linux-gnueabi/libc/thumb2`
  - Crosstool-NG alati mogu takođe biti multilib (još u eksperimentalnoj fazi), u suprotnom sysroot je u
    `arm-unknown-linux-uclibcgnueabi/sysroot`

Sekvenca pokretanja (boot)

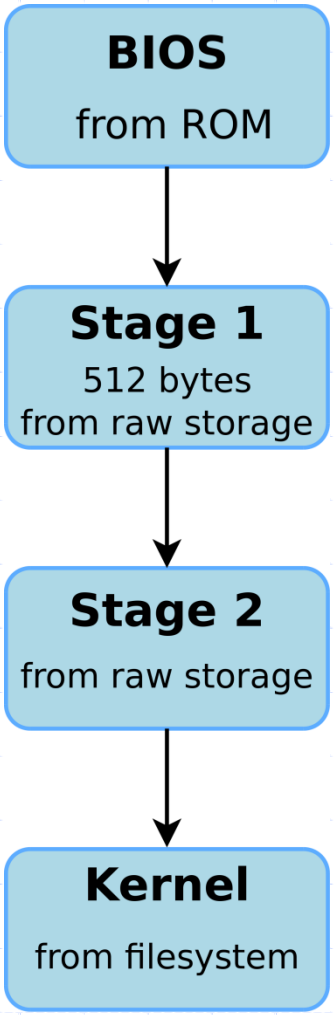
RAZVOJNO OKRUŽENJE LINUKSA U NAMENSKIM SISTEMIMA

Bootloader-i

- Bootloader je deo koda odgovoran za
  - Osnovnu inicijalizaciju hardvera
  - Učitavanje binarne datoteke aplikacije, najčešće kernela operativnog sistema, iz flash memorije, sa mreže ili sa drugog tipa memorije.
  - Može da uključi i dekompresiju binarne datoteke aplikacije.
  - Izvršavanje aplikacije
- Osim ovih osnovnoih funkcionalnosti, većina bootloader-a nudi šel sa različitim komandama koji implementira različite operacije.
  - Učitavanje podataka iz skladištene memorije ili sa mreže, provera memorije, hardverska dijagnostika, testiranje, itd.

Bootloader-i na BIOS-baziranoj x86 arhitekturi (1/2)

- x86 procesori su tipično upakovani u ploču sa trajnom memorijom koja sadrži program pod nazivom BIOS.
- Na starim BIOS-baziranim x86 platformama: BIOS je odgovoran za inicijalizaciju osnovnog hardvera i učitavanje veoma malog dela koda iz trajne memorije.
- Ovaj deo koda je tipično bootloader prve faze koji će učitati potpun bootloader.
- Tipično ima podršku za različite formate sistema datoteka tako da može da učita sliku kernela direktno sa uobičajenog sistema datoteka.
- Ova sekvenca se razlikuje na modernijim, EFI-baziranim sistemima.



Bootloader-i na BIOS-baziranoj x86 arhitekturi (2/2)

- GRUB, Grand Unified Bootloader, najmoćniji bootlader.
  http://www.gnu.org/software/grub/
  - Može da čita mnoge formate sistema datoteka, a učitava slike kernela i konfiguracije, nudi moćan šel sa različitim komandama, može da učita sliku kernela preko mreže, itd.
  - Za detalje pogledajte prezentaciju:
    http://free-electrons.com/docs/grub/
- Syslinux, za boot-ovanje preko mreže ili prenosnog medijuma (USB key, CD-ROM)
  http://www.kernel.org/pub/linux/utils/boot/syslinux/

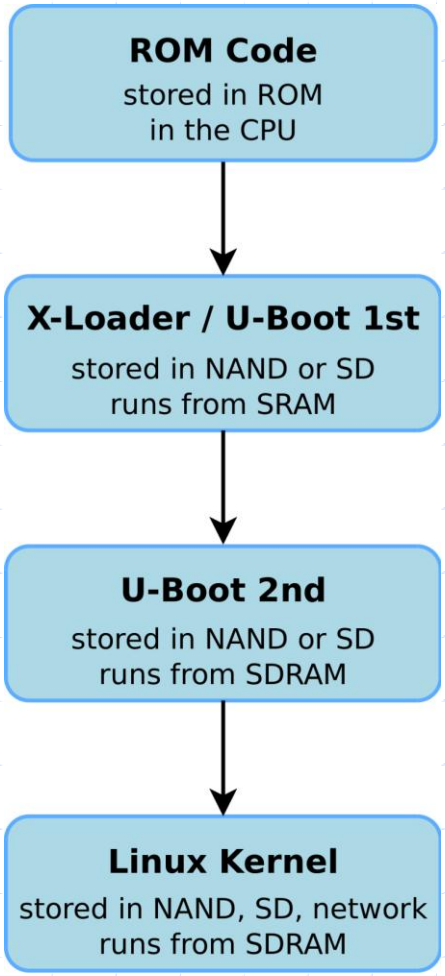
Boot-ovanje na namenskim procesorima: slučaj 1

- Kada se uključi, CPU počne izvršavanje koda na određenoj, fiksnoj adresi
- Ne postoji drugi mehanizam boot-ovanja omogućen od CPU
- Hardver mora biti dizajniran tako da osigura da NOR flash čip bude povezan tako da je dostupan na adresi na kojoj CPU započinje izvršenje instrukcija
- Bootloader prve faze mora da se nalazi na ovoj adresi u NOR memoriji
- NOR je mandatoran jer omogućava slučajan pristup, koji NAND nedozvoljava
- **Nije više toliko uobičajen** (nepraktičan, zahteva NOR flash)

Boot-ovanje na namenskim procesorima: slučaj 2

- CPU ima integrisan kod za boot-ovanje u ROM-u
  - BootROM na AT91 CPU, "ROM code" na OMAP, itd.
  - Detalji zavise od CPU
- Ovaj kod za boot-ovanje može da pokrene bootloader prve faze iz skladišta u SRAM (DRAM još nije inicijalizovan u to vreme)
  - Skladište može biti tipa: MMC, NAND, SPI flash, UART (prenos podataka preko serijske linije), itd.
- Bootloader prve faze je
  - Ograničene veličine zbog hardverskih ograničenja (veličina SRAM)
  - Obezbeđen od strane proizvođača CPU ili kroz projekte zajednice
- Bootloader prve faze mora da inicijalizuje DRAM i ostale hardverske uređaje i učita bootloader druge faze u RAM

Boot-ovanje na ARM TI OMAP3



- **ROM Code**: pokušava da pronađe validnu bootstrap sliku na različitim lokacijama i učitava je u SRAM ili RAM (RAM može biti inicijalizovancan ROM Code-om kroz konfiguraciona zaglavlja). Veličina je ograničena na <64 KB. Nije moguća interakcija sa korisnikom.
- **X-Loader** ili **U-Boot**: pokreće se iz SRAM. Inicijalizuje DRAM, NAND ili MMC kontroler i učitava u RAM bootloader druge faze i pokreće ga. Nije moguća interakcija sa korisnikom. Datoteka pod nazivom `MLO`.
- **U-Boot**: pokreće se iz RAM-a. Inicijalizuje još neke uređaje (mreža, USB, itd). Učitava sliku kernela iz trajne memorije ili preko mreže u RAM i pokređe ga. Dostupan je šel sa komandama. Datoteka pod nazivom `u-boot.bin` ili `u-boot.img`.
- **Linux Kernel**: pokreće se iz RAM-a. Preuzima kompletan sistem (bootloader-i više ne postoje).

Generički bootloader-i za namenske CPU

- Fokusiraćemo se na generički deo, glavni bootloader, koji nudi najvažnije funkcionalnosti.
- Postoji nekoliko generičkih bootloader-a otvorenog koda.
- Najpopularniji:
  - **U-Boot**, univerzalni bootloader razvijen od Denx Najkorišćeniji na ARM arhitekturama, ali i na PPC, MIPS, x86, m68k, NIOS, itd. To je danas de-facto standard. Izučavaćemo ga detaljno.

    http://www.denx.de/wiki/U-Boot
  - **Barebox**, novi bootloader nezavisan od arhitekture, pisan kao naslednik U-Boot-a. Bolji dizajn, bolji kod, aktivni razvoj, ali i dalje nema toliko harverske podrške koliko U-Boot.

    http://www.barebox.org}
- Postoje još mnogo drugih namenskih bootloaders otvorenog koda, često zavisnih od arhitekture
  - RedBoot, Yaboot, PMON, itd.

U-boot

RAZVOJNO OKRUŽENJE LINUKSA U NAMENSKIM SISTEMIMA

U-Boot

- U-Boot je tipičan projekat besplatnog softvera
  - Licenca: GPLv2 (kao i Linuks)
  - Besplatno dostupno na http://www.denx.de/wiki/U-Boot
  - Dokumentacija dostupna na http://www.denx.de/wiki/U-Boot/Documentation
  - Poslednji razvojni izvorni kod se nalazi na Git repozitorijumu: http://git.denx.de/?p=u-boot.git;a=summary
  - Razvoj i diskusija se dešavaju kroz otvorenu mejling listu http://lists.denx.de/pipermail/u-boot/
  - Od kraja 2008, prati fiksan intervala izbacivanja novih verzija.
  - Svaka tri meseca se izbacuje nova verzija. Verzije su imenovane po konvenciji u formatu `YYYY.MM`

U-Boot konfigurisanje

- Preuzmite izvorni kod sa sajta i raspakujte ga
- `configs/` direktorijum sadrži po jednu konfiguracionu datoteku za svaku podržanu ploču.
  - Ona definiše tip CPU, periferije i njihove konfiguracije, memorijsku mapu, U-Boot funkcionalnosti koje treba prevesti i sl.
- Napomena: U-Boot migrira sa konfiguracionih datoteka ploče u zaglavljima na putanji (`include/configs/`) na *defconfig* kao što je i u Linux kernelu (`configs/`)
  - Nisu sve ploče prebačene na novi konfiguracioni sistem.
  - Starije verzije U-Boot-a ponuđene od strane proizvođača možda još ne koristi ovaj konfiguracioni sistem.

U-Boot konfiguraciona datoteka

- CHIP_defconfig

```
CONFIG_ARM=y
CONFIG_ARCH_SUNXI=y
CONFIG_MACH_SUN5I=y
CONFIG_DRAM_TIMINGS_DDR3_800E_1066G_1333J=y
# CONFIG_MMC is not set
CONFIG_USB0_VBUS_PIN="PB10"
CONFIG_VIDEO_COMPOSITE=y
CONFIG_DEFAULT_DEVICE_TREE="sun5i-r8-chip"
CONFIG_SPL=y
CONFIG_SYS_EXTRA_OPTIONS="CONS_INDEX=2"
# CONFIG_CMD_IMLS is not set
CONFIG_CMD_DFU=y
CONFIG_CMD_USB_MASS_STORAGE=y
CONFIG_AXP_ALDO3_VOLT=3300
CONFIG_AXP_ALDO4_VOLT=3300
CONFIG_USB_MUSB_GADGET=y
CONFIG_USB_GADGET=y
CONFIG_USB_GADGET_DOWNLOAD=y
CONFIG_G_DNL_MANUFACTURER="Allwinner Technology"
CONFIG_G_DNL_VENDOR_NUM=0x1f3a
CONFIG_G_DNL_PRODUCT_NUM=0x1010
CONFIG_USB_EHCI_HCD=y
```

Konfiguracija i prevođenje U-Boot-a

- U-Boot mora biti konfigurisan pre prevođenja
  - `make BOARDNAME_defconfig`
  - Gde je `BOARDNAME` ime konfiguracije, kako se vidi u `configs` direktorijumu.
  - Nakon toga možete pokrenuti `make menuconfig` za dalju definiciju U-Boot konfiguracije!
- Uverite se da je alat za unakrsno prevođenje dostupan u `PATH`
- Prevedite U-Boot, navođenjem prefiksa za unakrsno prevođenje.
- Npr, ukoliko je izvršna datoteka vašeg unakrsnog prevodioca `arm-linux-gcc`: `make CROSS_COMPILE=arm-linux-`
- Glavni rezultat prevođenja je u datoteci `u-boot.bin`, koja predstavlja sliku U-Boot-a. Zavisno od vaše specifične platforme, mogu postojati i druge slike: `u-boot.img`, `u-boot.kwb`, `MLO`, itd.

Instalacija U-Boot-a

- U-Boot najčešće mora biti instaliran u flash memoriju da bi ga hardver mogao pokrenuti. Zavisno od hardvera, instalacija U-Boot se radi na različite načine:
  - CPU nudi neku vrstu posebnog boot monitora sa kojim se može komunicirati kroz serijski port ili USB korišćenjem specifičnog protokola.
  - CPU boot-uje prvi spoljašnji medijum (MMC) pre boot-ovanja sa fiksnog medijuma (NAND). U ovom slučaju, boot-uje se sa MMC-a da bi se upisala nova verzija
  - U-Boot je već instaliran i može se koristiti za upis nove verzije U-Boot-a. Međutim, treba biti oprezan: Ukoliko nova verzija U-Boot-a ne radi, ploča je neupotrebljiva
  - Ploča nudi JTAG spregu koja omogućava upis u flash memoriju, bez ikakvog sistema pokrenutog na ploči. Takođe omogućava i spašavanje ploče u slučaju da bootloader ne radi.

U-boot prompt

- Povežite odredišnu platformu sa razvojnom stanicom serijskim kablom, odnosno serijskom konzolom.
- Uključite ploču. Na serijskoj konzoli ćete videti nešto poput:

```
U-Boot 2016.05 (May 17 2016 - 12:41:15 -0400)

CPU: SAMA5D36
Crystal frequency:       12 MHz
CPU clock        :      528 MHz
Master clock     :      132 MHz
DRAM:  256 MiB
NAND:  256 MiB
MMC:   mci: 0

In:    serial
Out:   serial
Err:   serial
Net:   gmac0

Hit any key to stop autoboot:  0
=>
```

- U-Boot šel nudi skup komandi. Izučićemo one najvažnije, a za potpunu referencu pogledajte dokumentaciju ili `help` komandu.

Komande sa informacijama

- Flash informacije (NOR i SPI flash)

```
U-Boot> flinfo
DataFlash:AT45DB021
Nb pages: 1024
Page Size: 264
Size= 270336 bytes
Logical address: 0xC0000000
Area 0: C0000000 to C0001FFF (RO) Bootstrap
Area 1: C0002000 to C0003FFF Environment
Area 2: C0004000 to C0041FFF (RO) U-Boot
```

- NAND flash informacije

```
U-Boot> nand info
Device 0: nand0, sector size 128 KiB
  Page size      2048 b
  OOB size         64 b
  Erase size    131072 b
```

- Detalji verzije

```
U-Boot> version
U-Boot 2016.05 (May 17 2016 - 12:41:15 -0400)
```

Važne komande (1/2)

- Tačan skup komandi zavisi od U-Boot konfiguracije
- `help` i `help command`
- `ext2load`, učitava datoteku sa ext2 sistema datoteka u RAM
  - Takođe `ext2ls` lista datoteke, `ext2info` daje dodatne informacije o datoteci
- `fatload`, učitava datoteku sa FAT sistema datoteka u RAM
  - Takođe `fatls` lista datoteke, `fatinfo` daje dodatne informacije o datoteci
- `tftp`, učitava datoteku sa mreže u RAM (sa TFTP servera)
- `ping`, testira mrežu
- `boot`, pokreće podrazumevanu boot komandu, sačuvanu u promenljivoj `bootcmd`
- bootz `<address>`, pokreće sliku kernela učitanu na datu adresu u RAM memoriji

Važne komande (2/2)

- `loadb`, `loads`, `loady`, učitava datoteku sa serijske linije u RAM
- `usb`, inicijalizuje i kontroliše USB podsistem,
- uglavnom se koristi za USB uređaje za skladištenje podataka, kao što je USB flash
- `mmc`, za inicijalizaciju i kontrolu MMC podsistema, koristi se za SD i microSD kartice
- `nand`, za brisanje, čitanje i pisanje sadržaja u NAND flash
- `erase`, `protect`, `cp`, za brisanje, izmenu, zaštitu i upis u NOR flash
- `md`, prikazuje sadržaj memorije. Može biti korisno za proveru sadržaja učitanog u memoriju ili za pregled hardverskih registara.
- mm, modifikuje sadržaj memorije. Može biti korisno za direktnu modifikaciju hardverskih registara u cilju testiranja.

Varijable okruženja: principi

- U-Boot može da se konfiguriše kroz varijable U-Boot okruženja
  - Neke specifične varijable utiču na ponašanje različitih komandi
  - Dodatne varijable se mogu dodati i koristiti u skriptama
- Varijable okruženja se učitavaju iz flash memorije u RAM u toku pokretanja U-Boot-a, mogu biti modifikovane i sačuvane trajno na flash.
- Postoji namenska lokacija u flash-u (ili u MMC) za čuvanje U-Boot okruženja, definisano u konfiguracionoj datoteci ploče

Komande varijabli okruženja

- Komande za manipulisanje varijablama:
  - `printenv`

    Prikazuje sve varijable i njihove vrednosti
  - `printenv <variable-name>`

    Prikazuje vrednost navedene varijable
  - `setenv <variable-name> <variable-value>`

    Postavlja novu vrednost varijable, samo u RAM
  - `editenv <variable-name>`

    Menja vrednost varijable, samo u RAM
  - `saveenv`

    Čuva trenutno stanje okruženja na flash

Komande varijabli okruženja - Primer

```
u-boot # printenv
baudrate=19200
ethaddr=00:40:95:36:35:33
netmask=255.255.255.0
ipaddr=10.0.0.11
serverip=10.0.0.1
stdin=serial
stdout=serial
stderr=serial
u-boot # printenv serverip
serverip=10.0.0.1
u-boot # setenv serverip 10.0.0.100
u-boot # saveenv
```

Važne U-Boot promenljive okruženja

- `bootcmd`, sadrži komandu koju će U-Boot automatski izvršiti u vreme boot-a nakon isteka konfigurisanog vremena `bootdelay`, ukoliko proces nije prekinut
- `bootargs`, sadrži argumente koji se prosleđuju Linuks kernelu
- `serverip`, IP adresa servera na koji će se U-Boot spojiti prilikom izvršavanja komandi vezanih za mrežu
- `ipaddr`, IP adresa koju će U-Boot koristiti
- `netmask`, mrežna maska za povezivanje sa serverom
- `ethaddr`, MAC adresa, može se postaviti samo jednom
- `autostart`, ukoliko je postavljeno na yes, U-Boot automatski pokreće sliku koja je učitana u memoriju
- `filesize`, veličina datoteke poslednje kopirane u memoriju (kroz `tftp`, `fatload`, `nand read`...)

Skripte i promenljive okruženja

- Promenljive okruženja mogu da sadrže manje skripte, da izvršavaju nekoliko komandi i testiraju rezultat komandi.
  - Korisno za automatizaciju booting ili upgrade procesa
  - Nekoliko komandi može biti ulančano korišćenjem operatora `;`
  - Testovi se sprovode konstrukcijom

    `if command ; then ... ; else ... ; fi`
  - Skripte se izvršavaju korišćenjem `run <variable-name>`
  - Možete referencirati druge promenljive korišćenjem `${variable-name}`
- Primer
  - `setenv mmc-boot 'if fatload mmc 0 80000000 boot.ini; then source; else if fatload mmc 0 80000000 zImage; then run mmc-do-boot; fi; fi'`

Prenos podataka do odredišne platforme

- U-Boot je najčešće korišćen za učitavanje, prenos i pokretanje slike kernela, ali takođe omogućava i izmenu slike kernela i korenskog sistema datoteka
- Datoteke moraju biti razmenjene između odredišne platforme i razvojne radne stanice. Ovo je moguće:
  - Kroz mrežu ukoliko odredišna platfoma ima Eternet konekciju i U-Boot sadrži rukovalac za Eternet čip. Ovo je najbrže i najefikasnije rešenje.
  - Kroz USB flash, ako U-Boot podržava USB kontroler vaše platforme
  - Kroz SD ili microSD karticu, ukoliko U-Boot podržava MMC kontroler na vašoj platformi
  - Kroz serijski port

TFTP

- Mrežni prenos od razvojne stanice do U-Boot-a na ciljnoj platformi se odvija kroz TFTP
  - *Trivial File Transfer Protocol*
  - Donekle sličan FTP-u, ali bez autentifikacije preko UDP-a
- TFTP server je potreban na razvojnoj stanici
  - sudo apt-get install tftpd-hpa
  - Sve datoteke u `/var/lib/tftpboot` su vidljive kroz TFTP
  - TFTP klijent je dostupan u `tftp-hpa` paketu, za testiranje
- TFTP klijent je integrisan u U-Boot
  - Konfiguriši `ipaddr` i `serverip` promenljive okruženja
  - Koristi `tftp <address> <filename>` za učitavanje datoteke