Napredno korištenje operacijskog sustava Linux

1. Boot proces i datotečni sustavi

Leonard Volarić Horvat
Nositelj: Stjepan Groš

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

04.03.2017

Sadržaj

Od uključivanja do /bin/bash

- BIOS/UEFI
- MBR/GPT
- Bootloader i odabir OS-a
- Učitavanje initramdiska i kernela
- Init proces
- Bash

BIOS

Basic **I**nput and **O**utput **S**ystem

- Firmware na matičnoj ploči - spremljen u ROM čipu
- Prvi softver koji se pokreće
- Izvršava Power-On Self-Test (POST) nad matičnom pločom i u nju spojenim uređajima (S.M.A.R.T. test diskova, Extension ROM dodatnih uređaja itd.)
- Konfigurira se preko ugrađenog firmwarea ili hardverski (jumperi, DIP switchevi, ...)
- Omogućuje odabir uređaja s kojeg se pokreće operacijski sustav (*boot sequence*)

UEFI

Unified **E**xtensible **F**irmware **I**nterface

- Specifikacija sučelja između OS-a i *firmwarea*
- Nadgradnja BIOS-a
- Nudi sve funkcionalnosti BIOS-a
- Puno naprednije mogućnosti
  - Servisi, aplikacije, driveri ...
  - EFI system partition (ESP)
- Najveći doprinos specifikacije - **GPT**, nasljednik MBR-a

UEFI

- Boot manager je dio UEFI specifikacije
- Traži i izvršava bootloader s neke od particija
- Detekcija UEFI bootloadera na sustavu
- *Compatibility support module (CSM)*
  - Omogućuje *backward compatibility* s klasičnim bootloaderima

Particije

- Hard disk sadrži particije formatirane određenim datotečnim sustavima
- Particije se koriste za funkcijsko i logičko odjeljivanje podataka
- Za pokretanje sustava s diska nužno je na njemu imati boot particiju
- Informacije o particijama sadržane su u
  - **M**aster **B**oot **R**ecord (MBR)
  - **G**UID **P**artition **T**able (GPT)

MBR

- Zapisan u prvom sektoru diska (512 B / 2 KiB)
- Sadrži redom:
  - **Bootstrap code** - minimalni podaci potrebni za nastavak boot procesa (446B)
  - *Partition table* - informacije o particijama (4*16B)
  - *Magic number* - zaštitna suma (2B)
- Može adresirati najviše 2 TB (4B za adresiranje; 512B sektori)
- Podržava (samo) 4 *primarne* particije
- Za veći broj particija koriste se *logičke* particije koje se upisuju u jednu primarnu particiju tipa *extended*
- U pravilu se MBR koristi s BIOS-om
  - UEFI-eva kompatibilnost unatrag omogućuje korištenje MBR-a i s UEFI-em
  - Nije idealno i ne koristi sve mogućnosti UEFI specifikacije

GPT

- Služi istoj svrsi kao i MBR, ali je napredniji i nudi više mogućnosti
  - Može adresirati 9.4 zetabajta (9.4*10^9 TB)
  - Nema realnog ograničenja na broj particija
  - Replikacija GPT-a na više mjesta na disku $\longrightarrow$ robusnost
- Sadrži i "protective MBR" na nultoj logičkoj adresi
  - Zaštita od "nesporazuma" s alatima koji očekuju MBR disk
  - *backward compatibility* s BIOS-om
- Za bootanje se koristi isključivo s UEFI-em
  - BIOS može koristiti GPT diskove za podatke - ali samo do 2TB!

Bootloader

- Izvršava se nakon POST-a
- Pokreće operacijski sustav - učitava kernel

- Prva faza bootloadera zapisana u tzv. *bootstrap code* MBR diskova
- Često se pokreće druga faza (npr. GRUB) s neke od particija

- UEFI&GPT diskovi imaju suštinski sličan, ali drugačije implementiran mehanizam
- EFI System Partition sadrži:
  - Bootloadere
  - Informacije o kernelima instaliranih OS-eva
  - Informacije o driverima
  - ...

Popularni bootloaderi

GRand **U**nified **B**ootloader

- GNU projekt
- Konfigurabilan za više operacijskih sustava na jednom računalu (*multiboot*)
- **GRUB Legacy**
  - Starija verzija
  - Više se ne razvija
- **GRUB 2**
  - Drugačije konfiguracijske datoteke
  - Modularan, puno mogućnosti
  - Zauzima puno više prostora od GRUB Legacy

Syslinux

- *Lightweight* bootloader
- Jednostavna konfiguracija

initrd/initramfs

- Kernel prepoznaje sve uređaje na sustavu i učitava **init**ial **r**am**d**isk
- Riječ je o pomoćnom datotečnom sustavu učitanom direktno u RAM
- Omogućuje kernelu učitavanje osnovnih modula kako bi došao do *root* filesystema
  - moduli
  - enkripcija
  - RAID / LVM
  - ...
- initrd se po završetku briše iz memorije
- Pokreće se init

Zašto initrd?

- initrd sadrži osnovni i vrlo mali *filesystem* koji zna baratati diskom na kojem se nalazi kernel
- Nužan za učitavanje (inherentno većeg) kernela

- Važan dio koncepcije otvorenih računarskih sustava je **modularnost**
- Od otvorenog se sustava očekuje podrška za lagano proširivanje funkcionalnosti kroz **module**
- initrd upravlja učitavanjem samo potrebnih modula pri pokretanju sustava
- Ostali se moduli mogu uključivati po potrebi (korištenjem naredbe *modprobe*)

init

- init je sustav zadužen za inicijalizaciju OS-a, a pokreće se odmah nakon pokretanja kernela
- System V (jedna od prvih komercijalnih Unix distribucija) uvodi koncept *runlevela*
- runlevel je način rada sustava

- U Linuxu definirano nekoliko runlevela

  0 - Sustav se isključuje
  1 / S - Single-user mode
  2 - 5 - Multi-user mode, nekoliko inačica
  6 - Ponovno pokretanje

- Stvarni *startup* level je određen u datoteci `/etc/inittab`

init

- Kod ulaska u svaki runlevel pokreću se skripte

  `/etc/rc.d/rc?.d`
- Poredak izvođenja skripti je određen imenom skripte u direktoriju:

  K01-K99 - skripta gasi servis pri ulasku u runlevel
  S01-S99 - skripta pokreće servis pri ulasku u runlevel
- Nakon izvođenja svih skripti u direktoriju pokreće se i `/etc/rc.local`
- Sve što se nalazi u direktorijima `/etc/rc.d` je obično samo symlink na skripte u `/etc/init.d`

systemd

- Moderni sustav upravljanja procesima i cijelim sustavom
- Istiskuje iz uporabe monolitni i glomazni init
- systemd koristi svaka popularna moderna distribucija

- Konfiguracija zasnovana na *unit* datotekama
- Neki tipovi *unit* datoteka

  daemon označava pozadinske, neinteraktivne procese (provjeravanje IO uređaja, logovi, mreža...)

  target grupira unite u skupine

  service, socket, ...

systemd

- Upravljanje systemd sustavom: naredba *systemctl*
  - *systemctl* - ispis svih jedinica
  - *systemctl status ime* - ispis trenutnog stanja jedinice *ime*
  - *systemctl enable httpd* - pokretanje httpd daemona pri pokretanju sustava
  - *systemctl start httpd* - instantno pokretanje httpd daemona
- Upravljanje logovima systemd sustava: naredba *journalctl*
  - systemd logove ne zapisuje tekstualno nego u binarnom zapisu
  - journalctl omogućuje čitanje tih zapisa
- Analiza vremena trajanja systemd sustava pri pokretanju OS-a: naredba *systemd-analyze*
  - *systemd-analyze blame* - distribucija trajanja po procesima pri pokretanju

Primjer dodavanja systemd servirsa

- Servisi se dodaju u direktorij /etc/systemd/system/
- Primjer: /etc/systemd/system/moj-program.service
- Na sljedećem je slajdu primjer systemd unit filea

```
[Unit]
Description=moj super kul program
After=network.target \# kao ovisi o mrezi pa kazes da se prvo digne mreza

[Service]
ExecStart=/opt/program.sh
\# tu se mogu dodati stvari poput
\# ExecStop=/opt/program.sh --stop \# ako postoji neki poseban nacin za zau
\# ExecReload=/opt/program.sh --reload \# ako postoji neki poseban nacin za
\# systemd ima defaultan nacin za gasenje programa (signali) tako da nije p
\# i po defaultu restart smatra gasenjem i paljenjem

\# jos tu mozes stavljati fileove koji specificiraju environment varijable
\# kul je ovaj restart
Restart=always
\# systemd ce sada pratiti je li program ziv i ako nije, restartati ce ga

[Install]
WantedBy=default.target \# u kojoj grupi se pokrece servis
```

Kernel moduli

"Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system."

- Moduli su spremljeni na lokaciji

  `/usr/lib/modules/kernel_release`

- `lsmod` - trenutno učitani moduli
- `modprobe module_name parameter_name=parameter_value` - učitavanje modula
- `modprobe -r module_name` - *unload* modula

  `/etc/modprobe.d/`

  ```
  options module_name parameter_name=parameter_value
  ```

Kernel domesticus

Kompajliranje kernela (CentOS)

```
yum -y groupinstall 'Development tools'
yum -y install ncurses-devel rpm-build qt-devel

cd /usr/src/kernels
wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.12.6.tar.bz2
tar xvjf linux-3.12.6.tar.bz2
cd /usr/src/kernels/linux-3.12.6
```

Izmijeniti konfiguraciju (promjeniti verziju u Makefile):

```
cp /boot/config`-uname -r` .config
make menuconfig \# ili make xconfig
```

Spremiti .config i kopajlirati

```
make; make modules_install; make install;
vi /boot/grub/grub.conf \# (default=1 to default=0)
```

- Dakle, pokretanje sustava ukratko izgleda ovako:
  - Procesor iz firmwarea čita i pokreće BIOS/UEFI
  - BIOS/UEFI saznaju lokaciju bootloadera na disku iz MBR-a/GPT-a
  - S diska se pokreće bootloader
  - Bootloader pokreće odabrani OS
  - OS prvo učitava initrd/initramfs da može učitati ostatak kernela
  - Kada se učita kernel, on pokreće init proces s PID-om 1 (SysVinit ili sve češće systemd)
  - init proces pokreće sve potrebne pozadinske procese
  - Korisnik pali terminal i u svojem omiljenom *shellu* pokreće naredbu *sl* i zadovoljan je

Literatura

https://www.happyassassin.net/2014/01/25/uefi-boot-how-does-that-actually-work-then/

https://wiki.archlinux.org/index.php/GRUB
http://wiki.gentoo.org/wiki/GRUB2_Quick_Start
https://wiki.archlinux.org/index.php/Syslinux

https://wiki.archlinux.org/index.php/Kernel_parameters
https://wiki.archlinux.org/index.php/Unified_Extensible_Firmware_Interface

http://0pointer.de/blog/projects/the-biggest-myths.html

http://users.cecs.anu.edu.au/~okeefe/p2b/power2bash/power2bash.html