Uvod u drajvere u Linux OS-u

dr Predrag Teodorovic

Fakultet Tehničkih Nauka, Novi Sad

November 4, 2016

Uvod

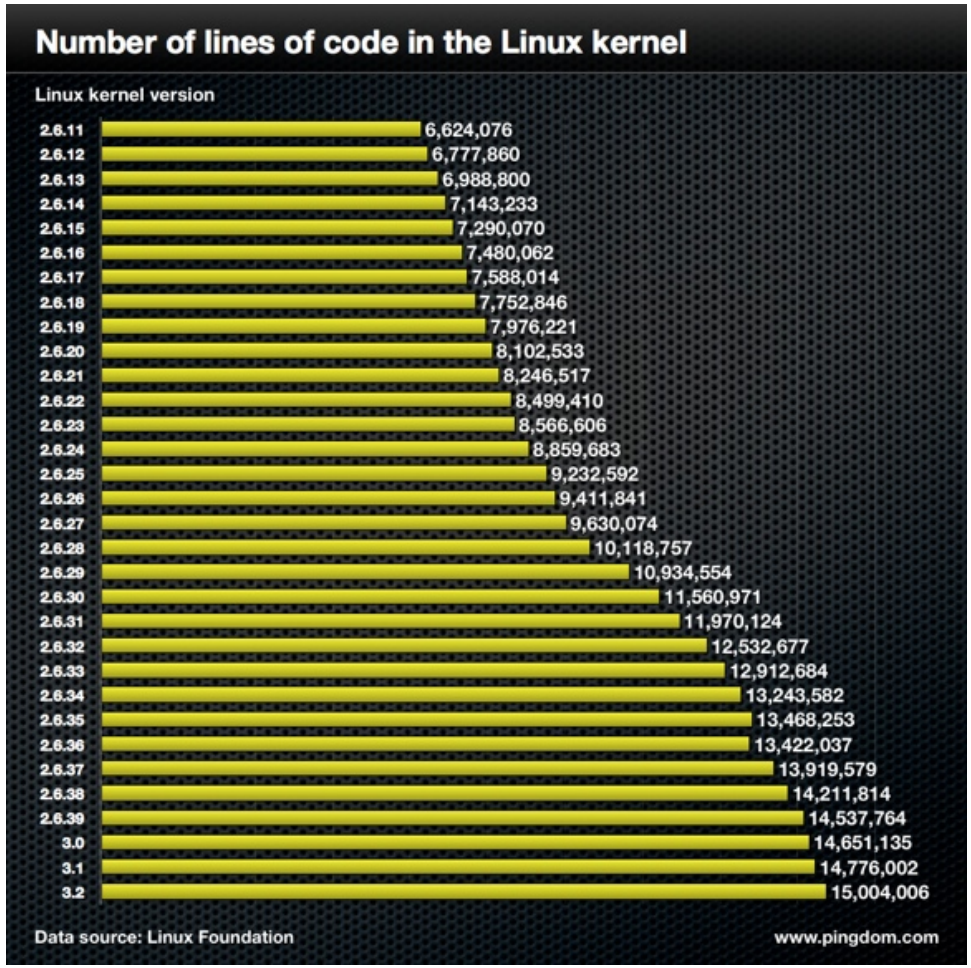
- Operativni sistemi se razlikuju od jednog do drugog, ne postoji generalizacija ni jedinstven opis i pravila
- Ukoliko želimo da se upoznamo sa detaljima, moramo odabrati jedan i na njemu raditi
- Mi ćemo u tu svrhu da koristimo Linux OS

Zašto Linux

- OS sa najvećom ekspanzijom trenutno (pogotovo u embedded svetu)
- OS otvorenog koda (Open-source OS)
- Omogućava korišćenje već postojećih ili modifikaciju/dodavanje novih softverskih modula
- Modularnost koja omogućava jednostavno „proširenje“ kernel-a u toku njegovog izvršavanja, nasuprot neophodnom prekompajliranju kernel koda i programiranju platforme na kojoj se izvršava

Drajveri

- Omogućavaju ulaznu tačku preko koje se pristupa kernel kodu, bez posledica činjenice da je kernel izuzetno kompleksan i velik (4.3 prva sa preko 20mil)



Drajveri

- Drajveri su „crne kutije“ koje u potpunosti kriju detalje kako uređaj funkcioniše
- Korisnikove aktivnosti se izvode setom standardnih poziva koji su nezavisni u odnosu na određeni drajver, a samo mapiranje tih poziva na operacije specifične za dati uređaj su uloga drajvera
- Modularnost podrazumeva programski interfejs koji omogućava razvoj drajvera nezavisno od ostatka hardvera i „uključivanje u akciju“ kada je to potrebno

Koja je motivacija?

- Tempo kojim se pojavljuje novi hardver, a zastareva postojeći, garantuje tvorcima drajvera priličnu uposlenost
- Proizvođači hardvera razvijajući drajvere dodaju veliku i rastuću Linux-ovu zajedničku korisničku bazu na svoje potencijalno tržište
- Open source priroda Linuxa podrazumeva da tvorac drajvera, ako želi, može svoj izvorni kod da podeli sa milionima korisnika

Uloga drajvera

- ***Vreme potrebno za razvoj*** VS ***fleksibilnost drajvera***
- Razdvajanje „mehanizma“ od načina njegovog izvršavanja je jedna od najboljih ideja koja stoji iza Linux (UNIX-like) sistema Odnosi se na podelu:
  - Koje mogućnosti pruža drajver (mehanizam)
  - Kako te mogućnosti da se koriste (polisa, eng. policy)

Primer

- Linux-ovo upravljanje grafičkim displejem je podeljeno između:
  - tzv. X-servera koji poznaje hardver i nudi unificirani interfejs korisničkim programima i
  - Prozorskog menadžera i menadžera sesije koji implementiraju određenu polisu bez ikakvog znanja o hw
- Isti prozorski menadžer može da se koristi na različitom hardveru i različiti korisnici mogu imati različite konfiguracije na istoj radnoj stanici. Čak i KDE i GNOME (različita desktop okruženja) mogu da koegzistiraju na istom sistemu

Još jedan primer

- Slojevita struktura TCP/IP umrežavanja
- OS nudi odvajanje priključka koji ne primenjuje polisu na podatke koji se prenose od servera koji upravljaju servisima i za njih vezanim polisama
- FTPD pruža mehanizam za prenos podataka dok korisnici mogu da koriste klijenta koji im najviše odgovara:
  - Grafički klijenti
  - Klijenti iz komandne linije
  - Svako može da napiše svoj novi korisnički interfejs za prenos datoteka

A kod drajvera

- Slično ovome, i kod drajvera se vrši razdvajanje mehanizma od polise
- Drajver za HD ima ulogu da prikaže disk kao blokove sa podacima, a viši nivoi sistema obezbeđuju polise kao na primer ko sme da pristupi podacima, da li se pristupa direktno ili preko fajl sistema i da li korisnici uopšte mogu postaviti fajl sistem na disk
- Programer koji piše drajver treba da vodi računa o ovome: **pisati kod kernela koji pristupa hardveru ali ne forsirati određene polise ka korisniku**

Šta drajver onda radi?

- Drajver se nosi sa funkcionalnošću hardvera ostavljajući sve detalje načina korišćenja hardvera aplikacijama
- Drajver je fleksibilan ako omogućava pristup mogućnostima hardvera bez uvođenja ograničenja
- Drajver je softverski sloj koji leži između aplikacija i uređaja
- Ova privilegovana uloga drajvera omogućava programeru da odabere kako će uređaj biti prikazan (npr. memorijsko mapiranje ili korisnička biblioteka)

Drajver bez polise

- Podrška kako za sinhrone tako i za asihnrone operacije
- Mogućnost da budu otvoreni više puta
- Mogućnost da u celosti koriste mogućnosti hardvera
- Manje softverskih slojeva koji „pojednostavljuju“ stvari koliko je to moguće
- Generalno se bolje ponašaju krajnjim korisnicima, lakši su za pisanje i održavanje

Razdvajanje kernela

- U Linux sistemu nekoliko konkurentnih procesa obavlja odgovarajuće operacije
- Svaki proces traži deo sistemskih resursa, bilo da je to procesorska moć, memorija, mrežna povezanost ili neki drugi resurs
- Kernel u sebe uključuje blok za: upravljanje procesima, upravljanje memorijom, fajl sistemom, uređajima, mrežom, raspoloživim modulima

Upravljanje procesima

- Kernel je zadužen za stvaranje i uništavanje procesa kao i za njihovo povezivanje sa spoljašnjim svetom
- Komunikacija između procesa je značajna za sveukupnu funkcionalnost sistema i takođe je upravljana od strane kernela
- Planer procesa kontroliše podelu resursa procesora među više procesa

Upravljanje memorijom

- Memorija je izuzetno važan resurs i polisa zadužena za nju je od kritičnog značaja za kvalitet funkcionisanja sistema
- Različiti delovi jezgra interaguju sa podsistemom za upravljanje memorijom kroz set funkcijskih poziva, počev od jednostavnog malloc/free para pa sve do mnogo složenijih funkcija

Upravljanje fajl sistemom

- Linux je striktno baziran na konceptu fajl sistema; gotovo sve u Linux-u se može tretirati kao fajl
- Kernel gradi strukturirani fajl sistem iznad nestrukturiranog hardvera i rezultujuća apstrakcija fajlova se u velikoj meri koristi kroz ceo sistem
- Linux podržava više tipova fajl sistema, tj. različite načine organizovanja podataka i njihovog smeštanja na fizički medij.
- Na primer, diskovi se mogu formatirati standardnim Linux ext3(4) fajl sistemom, često korišćenim FAT fajl sistemom kao i mnogim drugim sistemima

Upravljanje uređajima

- Skoro svaka sistemska operacija se mapira na fizički uređaj
- Sa izuzetkom procesora, memorije i vrlo malo drugih delova, sve ostale operacije za upravljanje uređajima se obavljaju preko koda koji je specifičan za svaki uređaj ponaosob
- Taj kod se zove **drajver**
- Kernel mora imati u sebi ugrađene drajvere za sve periferije koje su prisutne u sistemu, od hard diska do tastature

Umrežavanje

- Umrežavanjem treba da upravlja operativni sistem, zbog toga što većina mrežnih operacija nije karakteristična za proces, a primljeni paketi imaju karakter asinhronih događaja
- Paketi se moraju sakupiti, identifikovati i proslediti pre nego što ih proces preuzme
- Sistem upravlja isporukom paketa preko programskih i mrežnih interfejsa
- Svi elementi vezani za rutiranje i određivanje adresa implementirani su u kernelu

Raspoloživi moduli

- Jedna od dobrih osobina Linux-a je da se karakteristike koje nudi kernel mogu proširivati „u hodu“
- Jezgru se može pridodati funkcionalnost (ali mu se može i oduzeti) dok je sistem podignut i funkcioniše
- Svaki deo koda koji se može dodati kernelu dok je sistem podignut i radi se naziva modul
- Program *insmod* dodaje module kernelu, dok *rmmod* uklanja module iz kernela

Klase uređaja i modula

- Razlikujemo tri osnovna tipa uređaja:
  1. Sekvencijalni (eng. Character devices)
  2. Blok (eng. Block devices)
  3. Mrežni (eng. Network devices)
- Ova podela nije stroga, programer može da izgradi ogromne module u sklopu jedinstvenog dela koda
- Dobra praksa je da se ipak krera poseban modul za svaku novu funkcionalnost koja se implementira

Sekvencijalni (karakter) uređaji

- Uređaji kojima je moguće pristupiti kao nizu bajtova (kao u okviru datoteke)
- Takav drajver obično implementira bar osnovne sistemske pozive kao što su open, close, read i write
- Konzola i serijski portovi su primeri ovakvih uređaja, jer su dobro predstavljeni nizom karaktera
- Pristupa im se preko delova fajl sistema (dev/ttyS1 za serijski port)

Sekvencijalni (karakter) uređaji VS datoteke

- Razlika u odnosu na tekstualne datoteke je što se podacima može pristupati samo sekvencijalno (bez kretanja napred nazad)
- Bez obzira na to, postoje karakter uređaji koji imaju izgled zona podataka i u njima se može kretati napred-nazad
- Ovo se, na primer, obično odnosi na tzv frame buffer, gde aplikacija može da pristupi celoj preuzetoj slici korišćenjem komandi mmap ili lseek

Blok uređaji

- Kao i karakter uređajima, blok uređajima se može pristupiti preko delova fajl sistema u /dev direktorijumu
- Blok uređaj je uređaj u kome se može biti smešten fajl sistem
- Za razliku od Unix sistema gde je moguć prenos samo blokova određene veličine (npr. 512 bajtova), kod Linuxa se mogu prenositi blokovi proizvoljnog broja bajtova

Blok uređaji

- Blok i karakter urežaji se razlikuju samo u načinu na koji se upravlja podacima interno preko kernela, a samim tim i u softverskom interfejsu između kernela i drajvera
- Blok drajveri imaju potpuno drugačiji interfejs sa kernelom nego što je to slučaj kod drajvera sa sekvencijalne uređaje

Mrežni interfejs

- Mrežni uređaji su obično dizajnirani samo za slanje i prijem paketa
- Mrežni drajver ne zna ništa o poredinim konekcijama (TCP, UDP,...), on samo obrađuje pakete
- Ako uređaj nije usmerenog toka, mrežni interfejs nije lako mapirati u deo fajl sistema kao u slučaju /dev/ttyS1
- Pristup ovim interfejsima je još uvek baziran na dodeli jedinstvenih imena (npr. eth1) koja nemaju odgovarajuću instancu u okviru fajl sistema

Problem sa podelom

- Postoje i druge podele drajverskih modula, ortogonalne u odnosu na gore navedenu
- Neki drajveri podržavaju funkcije za dati tip uređaja
- Na primer, svaki USB uređaj je kontrolisan od strane USB modula koji radi sa USB podsistemom (klasom uređaja)
- Uređaj se u sistemu pojavljuje kao karakter uređaj (USB serijski port na primer) ili kao blok uređaj (USB fleš memorija) ili kao mrežni (USB Ethernet uređaj)

Druge klase drajvera

- U novije vreme, kernelu su dodate druge klase drajvera (FireWire, I2C, itd)
- Na sličan način kao što su se nosili sa USB ili SCSI drajverima, skupljene su zajedničke osobine tih klasa i implementirani drajveri u cilju izbegavanja dupliranja posla i grešaka, kao i pojednostavljenja procesa pisanja takvih drajvera

Druge klase drajvera

- Osim drajvera za uređaje, postoje dodatne funkcionalnosti modularizovane u jezgru (primer toga su fajl sistemi)
- Tip fajl sistema određuje način organizovanja informacija na blok uređaju u cilju predstavljanja stabla direktorijuma i fajlova
- Takav entitet nije drajver, više je „softverski” drajver jer mapira strukture podataka nižeg nivoa na one sa višeg (imena fajlova i putanja na strukture podataka smeštene u blokovima) i potpuno je nezavisan od stvarnog prenosa podataka (što radi blok uređaj)

Sigurnosne mere

- Sigurnost je tema kojoj se u sadašnjosti pridaje sve veća važnost
- Ima nekoliko generalnih koncepata koje treba spomenuti na ovu temu
- Bilo koja sigurnosna provera sistema potiče od koda kernela
- Ako kernel ima sigurnosne rupe, tada sistem u globalu ima rupe

Sigurnosne mere

- U zvaničnoj distribuciji kernela, samo autorizovani korisnik može da učitava module: sistemski poziv init_module proverava da li trenutni proces ima pravo učitavanja modula u kernel
- Stoga, kada se koristi zvanični kernel, samo superkorisnik ili uljez koji je uspeo da postane privilegovan, može da iskoristi snagu privilegovanog koda
- Kad god je to moguće, tvorci drajvera bi trebalo da izbegnu ubacivanje sugurnosne polise u svoj kod. Sigurnost je tema koja se najbolje obrađuje na najvišim nivoima u kernelu, pod kontrolom sistem administratora
- Ipak, izuzeci su uvek prisutni

Kojih se pravila treba pridržavati?

- Svaki ulaz primljen od strane korisničkih procesa treba tretirati sa velikom sumnjičavošću
- Nikad ne treba imati poverenja u nesto osim ako to može da se proveri
- Treba biti pažljiv sa neinicijalizovanom memorijom: svaka memorija dobijena od kernela mora biti anulirana ili na drugi način inicijalizovana pre nego što postane dostupna korisniku procesa ili uređaja
- Inače, curenje informacija (otvaranje podataka, lozinki, itd.) se može dogoditi

Kojih se pravila treba pridržavati?

- Ako vaš uređaj interpretira podatke koji su mu poslati, osigurajte da korisnik ne može da pošalje nešto što može da ugrozi sistem
- Na kraju, razmislite o mogućim efektima operacija nad uređajem - ako postoje neke posebne operacije (npr. nadgradnja firmvera na ploči adaptera ili formatiranje diska) koje mogu da utiču na sistem, te operacije treba uvek ograničiti na privilegovane korisnike

Uslovi licenciranja

- Linux je licenciran pod verzijom 2 GNU General Public Licence (GPL), dokumentom izdatim za GNU projekat od strane Free Software Foundation
- GPL dozvoljava svakom da redistribuira ili čak da prodaje proizvod koji se pod tom licencom nalazi, dokle god prijemna strana ima pristup izvornom kodu i može da prakticira ista ta prava
- Dodatno, svaki softverski proizvod izveden iz proizvoda koji je pod GPL mora, ako se uopšte distribuira, izdati takođe pod GPL licencom

Koji je cilj ovakve licence?

- Glavni cilj takve licence je da dozvoli rast znanja dozvoljavajući svakome da menja programe po svom nahođenju
- U isto vreme, ljudi koji prodaju softver to mogu da nastave da rade
- Uprkos ovom jednostavnom cilju, diskusiji u vezi sa GPL i njenom upotrebom nema kraja
- Ako želite da pročitate licencu, možete je naći na nekoliko mesta u sistemu, uključujući i gornji direktorijum stabla gde se nalazi pod nazivom COPYING