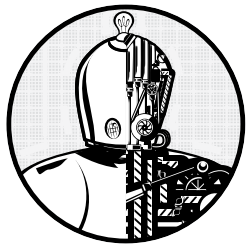
1

OPŠTA SLIKA



Na prvi pogled, jedan savremeni operativni sistem kao što je Linux izgleda vrlo komplikovano, s vrtoglavim brojem sastavnih delova koji rade istovremeno i međusobno komuniciraju. Na primer, veb server može „pričati“ sa serverom baza podataka, koji pak koristi deljenu biblioteku koju koriste i mnogi drugi programi. Ali kako sve to radi zajedno?

Kako radi operativni sistem najlakše ćete razumeti pomoću *apstrakcije* – što je slikovit način da se kaže da mnoge detalje možete zanemariti. Na primer, kada vozite auto, obično nema potrebe da razmišljate o detaljima kao što su zavrtnji koji drže motor unutar karoserije, ili ljudima koji grade i održavaju put po kojem vozite. Ako ste saputnik u vozilu, treba da znate samo šta vozilo radi (prevozi vas nekud) i ono osnovno o upotrebi tog vozila (kako se otvaraju vrata i vezuje sigurnosni pojas).

Ali ukoliko ste vozač auta, potrebno je da znate više. Treba da naučite kako se koriste njegove komande (kao što je volan i pedala gasa) i šta ćete uraditi ako nešto krene naopako.

Na primer, recimo da se tokom vožnje vozilo trese. Tada apstrakciju „vozilo se kreće po putu“ možete podeliti na tri dela: vozilo, put i način na koji vozite. To vam pomaže da ustanovite razlog problema: ako je put neravan, krivicu ne možete prebaciti na vozilo, niti na način na koji vozite. Umesto toga, možda ćete poželeti da saznate zašto je put u lošem stanju ili, ako je put nov, zašto su njegovi graditelji loše uradili svoj posao.

Kada grade operativni sistem i aplikacije za njega, projektanti softvera koriste apstrakciju kao alatku. Postoji više naziva za apstrahovani sastavni deo računarskog softvera, kao što su *podsistem*, *modul* i *paket* – ali u ovom poglavlju koristićemo reč *komponenta* zato što je jednostavna. Kada prave neku softversku komponentu, projektanti obično ne razmišljaju mnogo o internoj strukturi drugih komponenata, ali vode računa o tome koje druge komponente mogu da iskoriste i kako.

Ovo poglavlje pruža opšti uvid u komponente koje čine jedan Linux sistem. Mada se svaka sastoji od ogromnog broja tehničkih detalja koji opisuju njenu internu strukturu, te detalje ćemo zanemariti i usredsrediti se na to šta komponente rade i u kakvoj je to vezi s celim sistemom.

1.1 Nivoi i slojevi apstrakcije Linux sistema

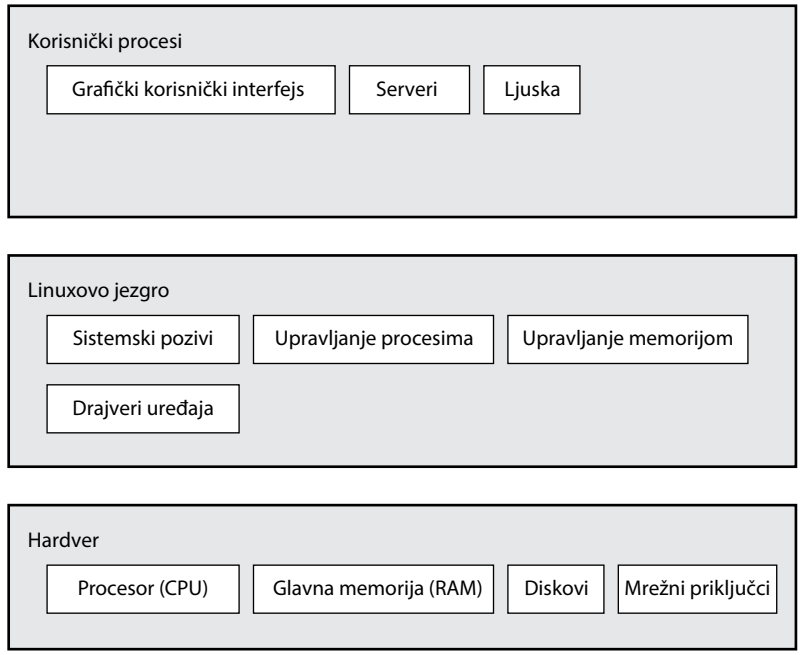
Upotreba apstrakcije radi deljenja računarskih sistema na komponente olakšava razumevanje stvari, ali je za to neophodna organizacija. Komponente razvrstavamo u slojeve ili nivoe. *Sloj* ili *nivo* se dobija klasifikacijom (tj. grupisanjem) date komponente na osnovu mesta gde se komponenta nalazi između korisnika i hardvera. Čitači veba (engl. *web browsers*), igre i slične aplikacije nalaze se u najvišem sloju; u najnižem sloju imamo memoriju u računarskom hardveru – nule i jedinice. Operativni sistem zauzima više slojeva između pomenuta dva.

Linux sistem ima tri glavna nivoa. Slika 1-1 prikazuje te nivoe i neke od komponenata unutar svakog nivoa. Osnovni nivo čini *hardver*. Hardver se sastoji od memorije i jedne ili više centralnih procesorskih jedinica (engl. *central processing unit*, CPU) koje obavljaju računske operacije i učitavaju podatke iz memorije, odnosno upisuju podatke u memoriju. Uređaji kao što su diskovi i mrežni interfejsi takođe su sastavni delovi hardvera.

Sledeći nivo je *jezgro* (engl. *kernel*) što je „srce“ operativnog sistema. Jezgro je softver koji se učitava u memoriju i govori procesoru šta treba da uradi. Jezgro upravlja upotrebom hardvera i prevashodno igra ulogu interfejsa između hardvera i svakog programa koji se izvršava.

Procesi – programi koji se izvršavaju i kojima upravlja jezgro – zajedno čine najviši nivo sistema, koji se zove *korisnički prostor* (engl. *user space*). (Precizniji izraz za proces je *korisnički proces*, bez obzira na to da li korisnik neposredno komunicira s procesom. Na primer, svi veb serveri rade kao korisnički procesi.)

Postoji ključna razlika između načina na koji rade jezgro i korisnički procesi: jezgro radi u *režimu jezgra* (engl. *kernel mode*), dok korisnički procesi rade u *korisničkom režimu* (engl. *user mode*). Programski kôd koji radi u režimu jezgra ima neograničen pristup procesoru i glavnoj memoriji. To je moćna, ali i opasna privilegija jer procesu koji radi u režimu jezgra omogućava da lako sruši ceo sistem. Oblast kojoj može da pristupa samo jezgro, zove se *prostor jezgra* (engl. *kernel space*).



Slika 1-1: Opšta organizacija Linux sistema

S druge strane, korisnički režim omogućava pristup samo u (najčešće prilično manji) podskup memorije i izvršavanje bezbednih CPU operacija. *Korisnički prostor* predstavlja delove glavne memorije u koje je odobren pristup korisničkim procesima. Ako se u procesu javi greška i on se sruši, posledice su ograničene, a jezgro može da „sredi" stanje posle njih. To znači sledeće: ako se vaš veb server sruši, to verovatno neće prekinuti obavljanje naučnih proračuna koje se već danima odvija u pozadini.

Teorijski gledano, korisnički proces koji se oteo kontroli ne može načiniti ozbiljnu štetu u preostalom delu sistema. U stvarnosti, to zavisi od onoga što podrazumevate pod „ozbiljnom štetom" i od konkretnih privilegija tog procesa, budući da je nekim procesima dozvoljeno da rade više stvari nego drugima. Na primer, da li neki korisnički proces može potpuno uništiti podatke na disku? Da, ako ima odgovarajuća ovlašćenja – a to možete smatrati prilično opasnim. Međutim, postoje bezbednosni mehanizmi da se to spreči, a većini procesa se jednostavno ne dozvoljava da čine štetu na taj način.

1.2 Hardver: glavna memorija

Od celokupnog hardvera koji čini računarski sistem, *glavna memorija* je možda njegov najvažniji deo. U svom najjednostavnijem obliku, glavna memorija je samo veliko mesto za skladištenje gomile nula i jedinica. Svaka nula ili jedinica zove se *bit*. Memorija je mesto gde „borave" jezgro i procesi koji se u datom trenutku izvršavaju – to su samo velike zbirke bîta. Ceo unos sa perifernih uređaja i sve što se njima prosleđuje protiče kroz glavnu memoriju,

takođe u obliku gomile bîta. CPU je samo operater koji radi s memorijom; učitava instrukcije i podatke iz memorije i upisuje rezultujuće podatke nazad u memoriju.

U vezi s memorijom, procesima, jezgrom i drugim delovima računarskog sistema često ćete čuti reč *stanje* (engl. *state*). Strogo govoreći, stanje je konkretna kombinacija bîta. Na primer, ako u memoriji imate četiri bita, 0110, 0001 i 1011 predstavljaju tri različita stanja.

Kada uzmete u obzir da jedan proces lako može biti sačinjen od više miliona bîta u memoriji, često je lakše da kada govorite o stanjima, koristite apstraktne izraze. Umesto da stanje opišete pomoću bîta, opisujete šta je urađeno ili šta se radi u datom trenutku. Na primer, možete reći „proces čeka unošenje podataka“ ili „proces izvršava korak 2 postupka svog pokretanja“.

NAPOMENA *Pošto je uobičajeno da se stanje opisuje pomoću apstraktnih termina umesto pomoću stvarnih bîta, izraz* slika *(engl.* image*) predstavlja određenu fizičku kombinaciju bîta.*

1.3 Jezgro

Zbog čega govorimo o glavnoj memoriji i stanjima? Gotovo sve što jezgro radi tiče se glavne memorije. Jedan od zadataka jezgra jeste da deli memoriju na više odeljaka i da sve vreme održava određene informacije o stanju tih odeljaka. Svaki proces dobija svoj deo memorije, a jezgro mora da se stara o tome da svaki proces ima pristup samo u svoj deo.

Jezgro je odgovorno za upravljanje poslovima u sledeće četiri opšte sistemske oblasti:

Procesi Jezgro je odgovorno za određivanje kojim je procesima dozvoljeno da koriste CPU.

Memorija Jezgro mora da nadzire upotrebu celokupne memorije – šta je u datom trenutku dodeljeno konkretnom procesu, šta bi moglo da se deli između više procesa i šta je slobodno.

Drajveri uređaja Jezgro igra ulogu interfejsa između hardvera (kao što je disk) i procesa. Direktan rad s hardverom najčešće je posao jezgra.

Sistemski pozivi i podrška U normalnim okolnostima procesi komuniciraju s jezgrom pomoću sistemskih poziva.

Sada ćemo ukratko istražiti svaku od pomenutih oblasti.

NAPOMENA *Ako vas zanimaju detalji o tome kako radi jezgro, dva dobra priručnika su* Operating System Concepts*, 9. izdanje, čiji su autori Abraham Silberschatz, Peter B. Galvin i Greg Gagne (izdavač je Wiley, 2012) i* Modern Operating Systems*, 4. izdanje, čiji su autori Andrew S. Tanenbaum i Herbert Bos (izdavač je Prentice Hall, 2014).*

1.3.1 Upravljanje procesima

Upravljanje procesima obuhvata pokretanje, pauziranje, nastavljanje i okončavanje procesa. Koncepti pokretanja i okončavanja procesa prilično su lako razumljivi, ali je teže opisati kako proces koristi CPU tokom svog uobičajenog rada.

U svakom savremenom operativnom sistemu, mnogi procesi rade „istovremeno“ Na primer, na svom stonom računaru možete imati otvorene u isto vreme čitač veba i radni list. Međutim, stvari se ne odvijaju onako kako izgleda: procesi iza tih aplikacija uglavnom se ne odvijaju *baš* u isto vreme.

Razmotrite sistem čiji CPU ima samo jedno jezgro. Više procesa *može* koristiti taj CPU, ali u svakom datom trenutku samo jedan proces zaista koristi taj CPU. U praksi, svaki proces koristi CPU tokom kratkog dela sekunde, a zatim pauzira; u tom trenutku drugi proces dobija CPU tokom drugog delića sekunde; zatim neki treći proces preuzima CPU itd. Događaj kada jedan proces prepušta CPU drugom procesu zove se *promena konteksta* (engl. *context switch*).

Svaki takav delić vremena – koji se zove *isečak vremena* (engl. *time slice*) – dovoljno je dugačak da proces može obaviti značajnu količinu proračuna (često se događa da proces do kraja obavi ceo svoj tekući posao u samo jednom isečku). Međutim, pošto su ti isečci tako kratki, ljudi ih ne mogu primetiti, pa im deluje da sistem izvršava više procesa u isto vreme (to se zove *višeprogramski rad* /engl. *multitasking/*).

Za promene konteksta odgovorno je jezgro. Da biste razumeli kako to funkcioniše, zamislite situaciju u kojoj neki proces radi u korisničkom režimu i njegov isečak vremena je upravo istekao. Evo šta se događa:

1. CPU (sam hardver) prekida tekući proces na osnovu internog tajmera, prelazi u režim jezgra i predaje kontrolu jezgru.
2. Jezgro beleži tekuće stanje CPU-a i memorije, što će kasnije biti ključno za nastavljanje procesa koji je upravo prekinut.
3. Jezgro obavlja poslove koji su možda bili pokrenuti tokom prethodnog isečka vremena (kao što je prikupljanje podataka koje je korisnik zadao ili podataka za ispisivanje, tj. ulazno/izlazne operacije).
4. Sada je jezgro spremno da omogući drugom procesu da se pokrene. Jezgro analizira listu procesa koji su spremni za pokretanje i iz nje bira jedan proces.
5. Jezgro priprema memoriju za taj novi proces, a onda priprema i CPU.
6. Jezgro obaveštava CPU koliko će trajati isečak vremena za novi proces.
7. Jezgro prebacuje CPU u korisnički režim i predaje CPU procesu.

Promena konteksta odgovara na važno pitanje *kada* jezgro radi. Ogovor je da radi tokom promena konteksta, u vremenskim isečcima *između* isečaka dodeljenih procesima.

U slučaju sistema s više CPU jedinica, stvari postaju nešto složenije zato što jezgro ne mora da prepusti svoj tekući CPU kako bi omogućilo datom procesu da se pokrene na drugoj CPU jedinici. Međutim, da bi maksimiralo upotrebu svih raspoloživih CPU jedinica, jezgro najčešće ipak baš to radi (a može primenjivati i određene trikove da bi za sebe prigrabilo malo više CPU vremena).