
Operativni sistem - Free BSD

Autor **Marko Uskoković** maturski rad

FreeBSD je istaknuti operativni sistem za arhitekture računara koje su kompatibilne sa Intel x86, DEC Alpha i PC-98 arhitekturama (druge platforme su u planu razvoja).

Free u imenu znači da se može koristiti besplatno, čak i za komercijalne upotrebe, ali takodje i u smislu slobodno. Sa izvornim kodom FreeBSD-a korisnik može da radi bilo šta - da ga čita, menja, prepravlja, kopira, prodaje, doslovno sve **osim da tvrdi da ga je on napisao**.

Prva distribucija FreeBSD-a na CD-ROM-ovima (i naravno na netu) bila je FreeBSD 1.0, objavljena u decembru 1993. godine. Bila je zasnovana na Berkeley-evoj 4.3BSD-Lite ("Net/2") traci, a naravno sadržala je i komponente 386BSD-a i mnoge programe *Free Software Foundation* (fondacija besplatnog-slobodnog softvera).

Literatura koja je korisćena

Knjige

- **The Design and Implementation of the 4.4BSD Operating System** - Kusick, Bostic, Karels, Quarterrman
- **The FreeBSD Handbook** - The FreeBSD Documentation Project
- **Operating Systems: Internals and Design Principles** - William Stallings
- **Design elements of the FreeBSD VM system** - Matthew Dillon

Net

- www.freebsd.org/
- www.bsd.org.yu/
- venus.elfak.ni.ac.yu/unix/html/
- www.unix-systems.org/
- www.opengroup.org/
- www.vmunix.com/fbsd-book/book.phtml

Istorija Unix operativnog sistema

Razvoj Unix-a u Bell-ovim laboratorijama

Prvu verziju operativnog sistema **UNIX** razvio je 1969. godine ***Ken Thompson***, član AT&T-ove istraživačke grupe Bell-ovih Laboratorija (*The Research Group at Bell Laboratories*) na DEC-ovim

računarima PDP-7 sa ciljem da napravi pogodan timesharing sistem. (*Kompjuterski sistemi tih godina su bili potpuno nekompatibilni, čak i kad su u pitanju različite verzije sistema istog proizvodjača. Operativni sistemi su izvršavali samo ograničene zadatke i to samo na mašinama na kojima su bili napisani. Ako bi firma želela da unapredi svoje kompjutere, to bi značilo instaliranje novog sistema, ali i ponovno unošenje svih podataka!*) Njemu se uskoro pridružio ***Dennis Richie***, koji je ranije radio na projektu **MULTICS** (**Multiplexed Information and Computing Service**), koji ekonomski nije bio pogodan za dalje razvijanje. Od MULTICS-a je preuzeto mnogo toga: organizacija fajl sistema, ideja da se komandni interpreter (nazvan shell - ljuska) izvršava kao korisnički proces (a ne sistemski), korišćenje različitih procesa za svaku komandu, linijsko editovanje i druge. Naravno na UNIX su uticali i drugi operativni sistemi, pogotovo MIT-ov (*Massachusetts Institute of Technology*) **CTSS** (*Compatible Time-Sharing Service*), a sam UNIX je uspeo u onome što oni nisu - postao je snažni operativni sistem, neobične jednostavnosti i elegancije.

Druga verzija UNIX-a bila je portovana na računare DEC PDP-11. Ove verzije UNIX-a bile su programirane u asembleru. Ken Thompson je 1970. godine napisao programski jezik B (po ugledu na Martin Richards-ov BCPL), i njega je Dennis Ritchie 1973. godine proširio strukturama podataka, stvorivši programski jezik C u kom će sledeća verzija UNIX-a biti implementirana. Ova verzija je pokazala da postoje mnoge prednosti pisanja operativnog sistema u jeziku visokog nivoa, kakav je C (multiprogramiranje, deljenje koda, reusability...), tako da su danas sve verzije UNIX-a napisane u C-u. UNIX je ubrzo prešao na sisteme čiji je hardver podržavao multiprogramiranje (PDP-11/45) i tako dobio velike mogućnosti unapredjivanja.

Ken Thompson i Dennis Ritchie su 1973. godine na simpozijumu o principima operativnih sistema, održanom na Purdue univezitetu, predstavili svoj rad o UNIX operativnom sistemu, a 1974. godine prvi put je opisan u tehničkom žurnalu. Predavanja o UNIX-u koja je Ken Thompson držao na Berkeley univerzitetu (*University of California at Berkeley - UCB*) bila su od velikog značaja za povećanje popularnosti UNIX-a na fakultetima. 1976. je **Version 6 UNIX** bio distribuiran mnogim fakultetima pod akademskim licencama. Na Berkeley-ju je osnovana grupa za istraživanje - *Computer Science Research Group (CSRG)*, čiji je rukovodilac bio ***Robert Fabry***. Unapredjenja koja je *CSRG* doneo UNIX-u nazvana su *Berkeley Software Distribution (BSD)*. Sa dolaskom UNIX-a na Berkeley-u počeo je i rad na INGRES bazi podataka, koja je preteča jedne od najboljih baza podataka današnjice - Posgres baze podataka.

Sledeća verzija **Version 7 UNIX** je objavljena 1978. godine. Ovaj UNIX sistem radio je na PDP-11/70 i Interdata 8/32 računarima, a uskoro je portovan i na VAX, gde dobija oznaku **32V**. **Version 8** iz 1985. uključuje I/O stream sistem (sistem ulazno/izlaznog toka) kojim se omogućuje fleksibilnost kernela u medjuprocesnoj komunikaciji (IPC), i uključuje se RFS (*Remote File System* - daljinski fajl sistem) sličan SUN-ovom NFS-u (*Network File System*) koji danas predstavlja standardni način deljenja fajlova medju UNIX-ima. Sledeće verzije (**Version 9** (1986), **Version 10** (1989) i **Plan 9** (1991)), ostale su dostupne samo istraživčkoj grupi unutar Bell-ovih laboratorija.

Istorija Unix operativnog sistema

Razvoj Unix-a od strane USG

Zbog svojih osobina i mogućnosti (timesharing os, multitasking i multiuser os) UNIX je postao primamljiv i za komercijalnu upotrebu. Nakon Verzije 7 formirana je posebna grupa, *UNIX Support Group (USG)*, u okviru AT&T sa zadatkom da se brine o UNIX sistemu. Njihova prva distribucija bila je **System III** (1981. god.) i ona je imala ugradjene mogućnosti iz Version 7, 32V, real time **UNIX/RT** i Programmer's Work Bench (**PWB**) sistema (interne verzije u Bell-ovim laboratorijama).

System V

USG je 1983. god. objavio **System V**, System III derivat, zatim se preformirao u *UNIX System Development Laboratory (USDL)*, i 1984. god. objavio **UNIX System V Release 2 (V.2)**. Sledeća verzija - **UNIX System V Release 2, Version 4 (V.2.4)**, donela je implementaciju virtualne memorije sa paging-om tipa kopiraj-pri-upisu i deljenu memoriju. USDL se transformisao u *AT&T Information Systems (ATTIS)*, koji je 1987. god. distribuirao **System V Release 3 (V.3)**. SVR3 je adaptirao implementaciju I/O stream sistema iz V8 (nazvanu STREAMS) i uključio je RFS.

UNIX System V Release 4

ATTIS 1989. godine postaje *UNIX Software Organization (USO)* i razvija **UNIX System V Release 4 (SVR4)**, kombinujući mogućnosti iz SVR3, 4.3BSD, Microsoft Xenix i SunOS operativnih sistema. Predstavlja skoro potpuno prekodiran System V kernel i podržava procesovanje u realnom vremenu, process scheduling klase, signale i IPC, dinamično alociranje struktura podataka, upravljanje virtualnom memorijom, virtualni fajl sistem i kontrolu poslova (job control). SVR4 radi kako na 32-bitnim mikroprocesorima tako i na superkompjuterima i u potpunosti je usaglašen sa POSIX standardom. SVR4 je razvijan tako da postane jedinstvena platforma za komercijalni razvoj UNIX-a, a na njemu su radili kako komercijalni tako i akademski projektanti. Oni su u tome svakako i uspeli, tako da SVR4 predstavlja možda i najznačajniju varijantu UNIX-a, pošto je objedinio sve najznačajnije mogućnosti UNIX-a koje su razvijene.

Posle izdavanja SVR4 USO postaje *Unix System Laboratories (USL)*, koji postaje deo Novell-a 1993. godine.

Istorija Unix operativnog sistema

Unix na Berkeley univerzitetu

Najveći doprinos UNIX-u doneo je *University of California at Berkeley (UCB)*. Prvu zvaničnu distribuciju UNIX-a (**3BSD**) napravio je 1978. god. ***Bill Joy***, autor vi-a (što je impliciralo i drajvere za terminale), čovek koji je značajno unapredio tadašnji pascal interpreter, jedan od osnivača Sun Microsystems-a, autor programskog jezika JAVA i jedan od najvećih UNIX hakera svih vremena. Ta distribucija predstavljala je VAX UNIX 32V, kome su dodati virtuelna memorija, straničenje na zahtev (demand paging) i zamena stranica (page replacement). Virtuelna memorija je omogućila da se UNIX koristi za razvoj velikih programa, kao što je Berkeley-ev Franz Lisp. Paralelno sa razvojem na VAX računarima, BSD se razvija i na računarima PDP-11, ali je manja cena VAX računara doprinela širenju BSD i u drugim akademskim okruženjima.

4BSD je nastao DARPA-inim (**Defense Advanced Research Projects Agency**) finansiranjem Berkeley-a za razvijanje standardnog UNIX sistema za vladine potrebe. **4.2BSD** je obezbedio rad u mreži po DARPA-inom Internet mrežnom protokolu (TCP/IP), tako da je obezbedjena uniformna komunikacija medju različitim elementima, uključujući lokalne mreze (kao što je Ethernet i token rings) i mreže računara (Arpanet i NSFNET). Berkeley-eva implementacija TCP/IP protokola se distribuirala na trakama kao "Networking Release 1", tj. "**Net/1**" trake.

Sledeća verzija, **4.3BSD**, donela je mnoga unapredjenja i poboljšanja performansi mrežne komunikacije. Berkeley je dozvoljavao distribuiranje '**Net/2**' traka korporacijama koje su licencirale AT&T-ov UNIX, pod uslovom da one u svojim distribucijama priznaju da koriste Berkeley-eve trake. Ovo je omogućilo proširenje Interneta od 60 povezanih mreža (1984), do preko 8000 mreža i 10 miliona korisnika u 1993.

Na Berkeley-u su razvijeni mnogi drajveri za terminale, novi korisnički interfejs (C shell), novi tekst editor (ex/vi), kompajler za Pascal, interpreter za Lisp, kao i mnogi drugi sistemski programi.

Poslednje Berkeley-evo izdanje UNIX-a **4.4BSD**, dovršeno je 1993. godine i ispunjava sve uslove koje zadaje POSIX standard. Pored toga sadrži radikalno novu organizaciju fajl sistema koja omogućava lako dodavanje novih mogućnosti. Uključena je i implementacija NFS-a, poboljšana sigurnost i unapredjena struktura kernela. Izdavanjem verzija 4.4BSD i **4.4BSD Lite** (za distribuciju) Berkeley je obustavio dalji razvoj.

Istorija Unix operativnog sistema

Standardizacija Unix-a

Kako je popularnost UNIX-a rasla, različite korporacije, organizacije i univerziteti su portovali UNIX na mnoge platforme računara, tako da je stvoren veliki broj UNIX-sličnih operativnih sistema. Primeri su DEC-ov **Ultrix**, IBM-ov **AIX**, Microsoft-ov **Xenix**, Sun-ov **SunOS** i **Solaris**, Hewlett-Packard-ov **HP-UX**, Compaq-ov **Tru64 UNIX**, Novell-ov **UNIX Netware**, SGI-ov **IRIX**, **Chorus**, **MACH**, **OSF/1**, **386BSD**, **SCO UNIX**, kao i mnogi drugi operativni sistemi koji su razvijani po ugledu na UNIX, kao što su: **Tunis**, **Xinu**, **Minix**,**Linux**.

UNIX je danas dostupan na svim računarima opšte namene, personalnim računarima (PC), radnim stanicama, minikompjuterima (PDP-11, VAX), mainframe računarima i super kompjuterima (cray).

Većina ovih sistema je bazirana na Version 7, System III, 4.2BSD i SystemV UNIX-ima. Kako UNIX nije bio razvijan od strane samo jednog proizvodjača, niti za samo jednu vrstu računara, postao je najprenosiviji i najzastupljeniji operativni sistem. Ovo bi moglo da omogući korisnicima da imaju jedinstveno okruženje na svim računarima bez obzira na proizvodjača. Ali je veliki broj implementacija sistema doveo do značajnih razlika u programiranju i korisničkom interfejsu ovih sistema. Da bi se korisnicima pružilo jedinstveno okruženje, mora postojati jedinstveni programski interfejs, tako da sve UNIX aplikacije koje poštuju taj interfejs mogu da se izvršavaju na svim UNIX sistemima.

Standardizacija UNIX-a je veoma težak zadatak i to zbog više faktora. Naime, kako je Ken Tompsonov prvobitni cilj pisanja prvog UNIX kernela bio poboljšanje performansi igre zvane Space Travel, on nije ni razmišljao da poštuje bilo kakve standarde. Dalji razvoj i korišćenje UNIX-a bilo je samo u okviru AT&T-a, tako da opet nije bilo potrebe za nekim zvaničnim standardom. Kada je UNIX izašao iz AT&T-a, počeo je njegov brzi razvoj u raznim pravcima, što je vodilo do pojave velikog broja varijanti UNIX-a i nemogućnosti pojave bilo kakvog standarda. Takodje, sa razvojem UNIX kernela, razvijane su i razne alatke, bez kojih današnji UNIX sistemi ne bi bili to što jesu, tako da standardizacija UNIX-a svakako mora da obuhvati i standardizovanje alatki kao što su lex, yacc, awk, grep...

Prvi pokušaj standardizovanja UNIX operativnih sistema bio je osnivanje nekormercijalne organizacije **/usr/group** u 1980. god, 7a 1984. god. **IEEE** (*Institute of Electrical and Electronics Engineers*) je iskoristio nacrt ovog standarda za **POSIX** (*Portable Operating System Interface for UNIX*) standard, koji bi trebalo da obuhvata 20 standarda, ali su do sada u potpunosti dovršena i objavljena samo dva - POSIX 1003.1 koji opisuje sistemske pozive i POSIX 1003.2 koji opisuje shell. POSIX ustvari predstavlja opis interfejsa, ali ne i implementaciju, što znači da i nije pravi standard. Ali, on predstavlja generalizaciju UNIX sistema, tako da su ga druge organizacije koristile kao osnovu za svoje standarde.

Odprilike u isto vreme AT&T je objavio **System V Interface Definition (SVID)**, a X/Open je objavio **X/Open Portability Guide (XPG)**. 1988 god. Open Software Foundation objavljuje OSF/1 verziju UNIX-a kompatibilnu sa POSIX-om i Motif - grafički interfejs koji danas predstavlja standardni korisnički grafički interfejs.

1993. godine Novell je kupio UNIX System Laboratory od AT&T-a, a zatim objavio sledeću verziju UNIX-a pod imenom **UnixWare**. Kako UnixWare nije stekao veliku popularnost, Novell je shvatio da ukoliko ne dodje do ozbiljne standardizacije UNIX-a, nijedan komercijalni UNIX sistem neće preživeti. Zbog toga je već 1994. god. Novell zvanično dao pravo korišćenja imena UNIX organizaciji X/Open i tako započeo najuspešniji pokušaj unifikacije UNIX-a do danas.

X/Open je utvrdio specifikaciju "**Spec 1170**", koja predstavlja nacrt više od 10 000 API-ja (*Application Programmer Interfaces*) kojima se opisuju UNIX-slični operativni sistemi, a 1995. god. objavljuje konačnu specifikaciju - "**Single UNIX Specification**". UNIX sistemi koji su uskladjeni sa ovom specifikacijom nazivaju se "**UNIX 95**".

Ujedinjenjem organizacija X/Open i OSF 1996. god. nastala je organizacija **Open Group** koja je objavila drugu verziju standarda "**Single UNIX Specification, Version 2**", a uskladjeni sistemi se nazivaju "**UNIX 98**". U ovoj specifikaciji uključeno je procesovanje u realnom vremenu, tredovi i 64-bitni (i više) procesori. Specifikacija je besplatna i može se dobiti preko web-a (www.opengroup.org).

2001. godine objavljena je i treća verzija specifikacije "**Version 3 of the Single UNIX Specification**".

Ovako je najpopularniji operativni sistem na svetu prošao težak put definisanja i standardizovanja.

Istorija Unix operativnog sistema

Dalji razvoj BSD-a

Kako je Berkeley dopuštao distribuciju svojih Net/1 i Net/2 traka pojavili su se mnogi operativni sistemi zasnovani na ovim distribucijama. Jedan od njih je i **386BSD** čiji je razvoj počeo kada je 1991. godine ***William Jolitz*** počeo sa portovanjem Net/2 (4.3BSD-Lite) trake na 386 računare. Medjutim, on nije bio u stanju da isprati sve popravke bug-ova i sve promene koje su vršene, tako da 1993. godine on prestaje sa razvojem. Programeri koji su sa njim radili podelili su se u 2 grupe koje su nastavile započeti posao. Takodje, 1992. godine je osnovana i firma **BSDi** koja je predstavila svoj komercijalni **BSD/OS** sistem. Na ovaj način je sa velikim uspehom nastavljen razvoj koji je započeo CSRG i stvoreni su najstabilniji, najportabilniji i najsigurniji sistemi današnjice.

FreeBSD

Početkom 1993. godine ***Jordan K. Hubbard***, ***Rod Grimes*** i ***Nate Williams*** su pokrenuli projekat čiji je cilj bio rešavanje problema koji su postojali u principima razvoja Jolitzovog 386BSD-a. Posle konsultovanja sa tadašnjim korisnicima sistema, uspostavljeni su principi i smišljeno je ime - **FreeBSD**. Pre nego što je konkretan razvoj i počeo, Jordan Hubbard je predložio je firmi **Walnut Creek CDROM** (danas **BSDi**) da pripreme distribuiranje FreeBSD-a na CD-ROM-ovima. Walnut Creek CDROM su prihvatili ideju, ali i obezbedili (tada potpuno nepoznatom) projektu mašinu na kojoj će biti razvijan i brzu Internet konekciju. Bez ove pomoći teško da bi FreeBSD bio razvijen u ovolikoj meri i ovolikom brzinom kao što jeste.

Prva distribucija FreeBSD-a na CD-ROM-ovima (i naravno na netu) bila je FreeBSD 1.0, objavljena u decembru 1993. godine. Bila je zasnovana na Berkeley-evoj 4.3BSD-Lite ("Net/2") traci, a naravno sadržala je i komponente 386BSD-a i mnoge programe *Free Software Foundation* (fondacija besplatnog-slobodnog softvera).

Nakon što je Novell otkupio UNIX od AT&T-a, Berkeley je morao da prizna da Net/2 traka sadrži velike delove UNIX koda. Posle sudskog spora, dogovoreno je da Berkeley pripremi 4.4BSD-Lite distribuciju iz

koje će biti izbačeni pojedini delovi koda. Od tada su BSD i UNIX, kontradiktorno, dve različite stvari. Svi korisnici Net/2 traka bili su prinudjeni da predju na 4.4BSD-Lite. Ovo je uključivalo i FreeBSD, tako da je pred projektom bio zadatak da dovrši sistem bazirajući se na novoj i nepotpunoj 4.4BSD-Lite distribuciji. Veliki delovi sistema su ponovo isprogramirani tako da je u novembru 1994. godine objavljena FreeBSD 2.0 distribucija koja je napokon mogla legalno da se distribuira preko neta i CDROM-ova.

Tekuća **-stable** grana projekta je 4.x-stable. **4.5-RELASE** je objavljen u januaru 2002. godine, a sledeće izdanje FreeBSD 4.6 zakazano je za 1. jun. 2002.

Krajem 2002. godine FreeBSD**-current** će biti objavljen kao **FreeBSD 5.0-RELASE**.

NetBSD

NetBSD je takodje nastao sa ciljem da poboljša mehanizam razvoja 386BSD-a. Ljudi iz firme **Virginia Tech** uzeli su za osnovu 386BSD i Net/2 traku i počeli su sa portovanjem BSD-a na računare Macintosh. Razvoj se ubrzo proširio i na Atari ST, Amiga, i PC platforme.

Kako se projekat razvijao, postalo je očigledno da će FreeBSD biti namenjen za i386, a NetBSD za sve ostale platforme. Ime je dobio zato što je do 1998. godine rad na njemu bio vezan isključivo za Net, a komunikacija medju developer-ima je bila bazirana na chat-u i mail-u.

Danas snaga NetBSD-a leži u podršci za čak 51 hardversku platformu. NetBSD je postao stabilni, multiplatformski, istraživački orijentisan operativni sistem. Posvećenost ka portabilnosti uticala je na druge operativne sisteme da i oni pružaju podršku za više platformi. Nekoliko operativnih sistema (OpenBSD, KAME, Darwin, MacOS X) izvedeno je direktno iz NetBSD-a. NetBSD je besplatan i može se snimiti sa neta. (www.netbsd.org). Aktivna verzija je 1.5.2 (maj 2002.)

OpenBSD

OpenBSD projekat je 1996. započeo ***Theo De Raadt***, jedan od glavnih programera projekta NetBSD. Umesto velike portabilnosti, OpenBSD se orijentisao ka sigurnosti. OpenBSD je operativni sistem sa izrazito jakim security-em, kao i sa kriptografijom (Blowfish, OpenSSH, PRNG, OpenSSL...) integrisanom u sam sistem. Takodje, programeri OpenBSD-ja su od 1996. godine počeli sa detaljnim pregledom koda kernela u potrazi za potencijalnim bug-ovima i exploit-ima, što je rezultiralo ispravljanju mnogih grešaka u kodu pre nego što su one i bile primećene u drugim sistemima.

OpenBSD je svakako najimpresivniji operativni sistem današnjice sa aspekta security-a, a takodje se i on može dobiti besplatno preko neta. (www.openbsd.org). Aktivna verzija je 3.1.

Dizajn

O FreeBSD-u

FreeBSD je istaknuti operativni sistem za arhitekture računara koje su kompatibilne sa Intel x86, DEC Alpha i PC-98 arhitekturama (druge platforme su u planu razvoja).

Cilj projekta je da se razvije softver koji će moći da se koristi u svaku svrhu.

FreeBSD je kompletan operativni sistem - kernel, C biblioteke i kompajler, mrežni serveri i klijenti, X-

Windows sistem, mnogi programi - sve što je potrebno za svakodnevni rad na kompjuteru. Izvorni kod celog sistema dostupan je na netu (ftp://ftp.FreeBSD.org/pub/FreeBSD/).

FreeBSD pruža napredne mrežne servise, dobru iskoristivost memorije, visoku stabilnost i performanse rada, tako da predstavlja ekonomičnu alternativu komercijalnim UNIX sistemima. Prilagodjen je radu sa velikim brojem desktop i server aplikacija. Pored dobrog security-a (bezbednosti), odlikuje se i fantastičnim performansama, kao i najboljom implementacijom TCP/IP stack-a što ga čini jednim od omiljenih izbora za web servere. U principu, FreeBSD je najpopularnija BSD distribucija.

Licenca

Free u imenu znači da se može koristiti besplatno, čak i za komercijalne upotrebe, ali takodje i u smislu slobodno. Sa izvornim kodom FreeBSD-a korisnik može da radi bilo šta - da ga čita, menja, prepravlja, kopira, prodaje, doslovno sve **osim da tvrdi da ga je on napisao**.

Osobine

FreeBSD ima mnogo osobina vrednih pažnje

- 32-bitni operativni sistem (64-bitni na Alpha-ma) koji je od početka bio dizajniran kao takav
- preemptivni multitasking sa dinamičnim podešavanjima prioriteta koji obezbedjuje fer i glatku podelu procesorskog vremena, memorije i drugih resursa računara čak i pri najvećim opterećenjima sistema
- mogućnost da opslužuje više korisnika u isto vreme tako što svi korisnici mogu da koriste resurse računara (štampače, mrežu...), ali takodje i da ograniči njihovu upotrebu
- SMP - podrška za mašine sa više procesora (samo za Intel)
- straničenje na zahtev koje omogućava brzo izvršavanje programa koji zahtevaju dosta memorije
- zaštita memorije koja obezbedjuje da programi (ili korisnici) ne mogu omesti jedni druge; ako se jedan program sruši, to neće uticati na ostale ni na koji način
- jak TCP/IP sa odličnom implementacijom TCP/IP stack-a, podrška za mnoge standarde kao što su SLIP, PPP, NFS, DHCP, NIS, serveri za e-mail, WWW, FTP, DNS, routing, firewall...
- X Window System (X11R6) kojim se pruža grafički korisnički interfejs (GUI)
- hiljade gotovih programa se nalaze u FreeBSD ports&packages kolekciji
- binarna kompatibilnost sa mnogim programima kompajliranim za Linux, SCO UNIX, SVR4, BSD/OS i NetBSD
- kompatibilnost izvornog koda programa sa većinom popularnih komercijalnih UNIX sistema, što omogućava jednostavno portovanje progama sa ovih sistema na FreeBSD
- kompletna zbirka razvojnih alata za mnoge programske jezike kao što su C, C++, Java, Fortran, Perl, Pascal (Delphi-Kylix)...
- izvorni kod celokupnog sistema, od kernela, preko alatki, do igara i način kompajliranja svega toga korišćenjem samo jedne komande
- velika količina veoma dobre dokumentacije

Zastupljenost

FreeBSD svakodnevno koriste razne kompanije, Internet provajderi, istraživači, studenti, korisnici kod kuće širom sveta.

Slogan "**The Power to Serve**" (snaga da služi) dobija pravo značenje kada se vidi da FreeBSD koriste mnogi veliki sajtovi da opsluže svoje korisnike, kao što su Yahoo!, Hotmail, Apache, Be,Inc., BSDi i mnogi drugi.

Model razvoja

FreeBSD se razvija tako da korisnicima sistema bude olakšano unapredjivanje sistema. Cilj je da se stvori stabilan operativni sistem i veliki broj dodatnih programa koje korisnici mogu lako da instaliraju (i deinstaliraju), koriste, prijave greške i daju sugestije za poboljšanje sistema i programa.

Da bi se ovo omogućilo, koristi se **CVS** (*Concurrent Version System*). To je alatka za kontrolisanje izvornog koda programa koja omogućuje korisnicima da u svakom trenutku dobiju doslovno svaku verziju FreeBSD-a, od prvih razvojnih verzija, pa sve do još neobjavljenih verzija koje se trenutno programiraju.

Committers (odbornici) su ljudi koji imaju pravo da prepravljaju CVS drvo. Oni koji pripadaju ovoj grupi su autorizovani da prave modifikacije izvornog koda FreeBSD-a. Naziv committer su dobili po komandi cvs-a *commit*, kojom prepravljaju CVS drvo.

Za dobro stanje projekta u celini i za planiranje daljeg razvoja odgovoran je **FreeBSD Core tim**. Oni daju commitersima pravo da prepravljaju CVS drvo i pozivaju posvećene i odgovorne programere da se pridruže projektu. Obično su odgovorni za razvijanje nekog većeg dela sistema. Biraju se svake 2 godine na izborima na kojima pravo glasanja imaju svi committersi.

Članovi core i committers timova su volonteri kada je u pitanju razvoj FreeBSD-a i nemaju nikakve finansijske koristi od projekta, pa bi bilo pogrešno smatrati da su oni obavezni da pružaju tehničku podršku. Za to su zadužene komercijalne firme koje zaradjuju distribuiranjem FreeBSD-a na diskovima, izdavanjem knjiga i pružanjem tehničke podrške.

Takodje, veliki udeo u razvoju FreeBSD-a imaju i sami **korisnici** koji šalju svoje sugestije i predloge, prijavljuju greške i šalju ispravke, dopunjuju i prevode dokumentaciju. Ovakvih ljudi je uvek bilo i uvek će biti mnogo, pogotovo kada je u pitanju FreeBSD projekat.

Razvoj

Postoje dve glavne grane razvoja FreeBSD-a

- current
- stable

Current

FreeBSD CURRENT je, bukvalno, ništa više nego dnevni prikaz radnih kopija izvornog koda FreeBSD-a. Nabavljanjem FreeBSD CURRENT preko CVS-a, korisnici mogu da vide na kom problemu trenutno rade programeri FreeBSD-a. Ovo je naročito potrebno članovima razvojne grupe FreeBSD-a, bilo da programiraju bilo da testiraju nove funkcije koje su dodate sistemu. FreeBSD CURRENT nije način za dobijanje novih programa pre zvaničnog izdavanja, jer su to još neistestirane, nekompatibilne i bagovite verzije.

Stable

FreeBSD STABLE je aktivna verzija FreeBSD-a koja je proglašena stabilnom i za koju se smatra da je ispravljena većina grešaka koje su se pojavile. Ove verzije koriste korisnici kojima je stabilnost celokupnog sistema važnija od nekih novijih mogućnosti. Objavljuju se na svaka 4 meseca.

Snap

FreeBSD SNAP je binarna verzija current, a **FreeBSD RELASE** stable izdanja.

Dizajn

Instalacija

FreeBSD se može instalirati sa CDROM-a, flopija, MS-DOS particije ili preko mreže (NFS ili FTP). Instalacija se smatra jednom od najlakših UNIX instalacija koje postoje. U idealnom slučaju treba boot-ovati mašinu sa CD-a (ako mašina to ne dozvoljava onda sa flopija), FreeBSD će prepoznati hardver i putem *Kernel Configuration* dijaloga će dati mogućnost njegovog podešavanja, zatim sledi particionisanje hard-diska i alociranje prostora, biranje distribucija koje će se instalirati i samo kopiranje fajlova. Prilikom sledećeg boot-ovanja kompjutera korisniku su dostupne sve mogućnosti FreeBSD-a.

FreeBSD-u je potrebna jedna primarna particija (koja se u FreeBSD terminologiji naziva **slice**) unutar koje će disklabel program napraviti **particije** fajl sistema.

FreeBSD nije zavistan od "logical BIOS drive mapping" kao drugi operativni sistemi već naziva diskove redom kako ih pronalazi. Fizički diskovi mogu biti IDE, SCSI, RAID, i dr.

Uredjaji

FreeBSD naziva ove uredjaje na sledeći način:

Dizajn

Ports & Packages

U FreeBSD-u korisnici imaju na raspolaganju dva načina za instaliranje novih programa: portovi i paketi. Kolekcija portova i paketa se sastoji od preko 6000 programa i mogu se naći na netu ili na CDROM-ovima. Ove kolekcije pored toga što pružaju veliki broj programa, znatno pojednostavljuju intstaliranje. Ideja je da korisnici ne moraju da traže programe "negde na netu", već da se stvori centralna kolekcija programa i mehanizam kojim bi se ti programi, koristeći samo jednu komandu, skinuli sa neta (ili snimili sa CDROM-a), raspakovali, zakrpili, kompajlirali i instalirali.

Kolekcija portova

Kolekcija portova se može snimiti prilikom instaliranja sistema. Port nekog programa se sastoji iz **makefile**-a (skript kojim se program kompajlira), **distinfo** fajla koji sadrži informacije o fajlovima koji se moraju download-ovati da bi se sagradio port, **checksum**-ova tih fajlova kako bi se proverio njihov integritet, i zakrpe (**patch**) koja sadrži informacije o promenama koje treba izvršiti kod odredjenih fajlova (zbog ispravljanja bug-ova programa i sl.). Instaliranje programa iz kolekcije portova se obavlja komandom *make install*. Deinstaliranje se vrši komandom *make deinstall*.

Kolekcija paketa

Kolekcija paketa predstavlja način snimanja binarnih verzija programa. Paket je kompresovani .tgz fajl. Da bi se instalirao jedan program iz kolekcije paketa sve što treba uraditi je skinuti odgovarajući .tgz fajl sa adrese ftp.FreeBSD.org/pub/FreeBSD/ports/packages/ i pokrenuti komandu *pkg_add*(1).

Deinstaliranje se vrši komandom *pkg_delete*(1).

Na CDROM-ovima se paketi nalaze u /packages direktorijumu.

Dizajn

Kernel

Kernel je deo sistema koji se izvršava u zaštićenom (protected) modu hardvera i koji posreduje izmedju korisničkih programa i hardvera. Kernel pruža osnovne funkcije sistema; on kreira, izvršava i upravlja procesima, pruža pristup fajl sistemu i mrežnim protokolima i dr. Ove funkcije se nazivaju **sistemski pozivi** (system calls) i korisniku su dostupne kroz biblioteku rutina koje može pozivati iz svojih programa. Sistemski pozivi su jedini način na koji procesi mogu da pristupe ovim servisima.

Govoreći tradicionalnom terminologijom operativnih sistema, kernel je malo jezgro softvera koje pruža minimalne servise koji su neophodni za implementiranje dodatnih servisa operativnog sistema. U savremenim istraživačkim operativnim sistemima kao što su **Chorus**, **Mach**, **Tunis** i **V Kernel**, servisi kao što su fajl sistemi, mrežni protokoli i dr. potpuno su izdvojeni iz kernela. No, UNIX kernel je od samog početka bio monolitni, tj. nije bio podeljen u više procesa. Ovakva struktura kernela je odabrana zbog jednostavnosti i performansi.

Najveći deo kernela implementira sistemske servise kojima aplikacije pristupaju kroz sistemske pozive. U 4.4BSD (pa tako i u FreeBSD) on obuhvata:

osnovne servise kernela
: manipulisanje tajmerom i sistemskim satovima, deskriptorima i procesima

upravljanje memorijom
: straničenje (paging) i zamena stranica (swapping)

opšti sistemski interfejsi
: I/O, kontrola deskriptora

fajl sistem
: upravljanje fajlovima, direktorijumima, prevodjenje putanja fajlova

manipulisanje terminalima
: interfejs terminala i discipline za procesovanje karaktera

medju-procesna komunikacija
: pipe i socket

mrežna komunikacija
    komunikacioni protokoli, routing

Većina ovih servisa su nezavisni od mašine i mogu se portovati na druge arhitekture. Delovi kernela koji su zavisni od konkretne mašine su izolovani od ovih delova. To su:

- startovanje sistema
- manipulisanje zamkama (trap handling) i greškama (fault handling)
- manipulisanje radnog konteksa procesa (run-time context) procesa
- konfigurisanje i inicijalizovanje hadrverskih uredjaja
- radna podrška za ulazno/izlazne (I/O) uredjaje

Granica izmedju programa koji se izvršavaju u kernel-modu i korisničkom modu se ostvaruje pomoću hardverske podrške ovim servisima. Kernel se izvršava u adresnom prostoru koji je nedostupan korisničkim programima. Privilegovane operacije, kao što su pokretanje ulazno/izlazne operacije ili upravljanje procesorom (CPU) dostupne su samo kernelu. Sistemske pozive koriste korisnički programi (aplikacije) kako bi pristupili ovim operacijama. Sistemski poziv se izvršava *sinhronizovano* sa aplikacijom koja ga je pozvala; program se ne izvršava dok kernel izvršava akciju koju mu nalaze sistemski poziv. Ukoliko izvršavanje sistemskog pozva traje dugo (npr. upisivanje podataka na disk), izvršavanje aplikacije se može nastaviti i pre nego što se sistemski poziv završi (podaci će biti snimljeni u bafer, pa kasnije snimljeni na disk).

Postoje tri vrste ulaza u kernel: hardverski interapt, hardverska zamka (trap) i softverska zamka (softwer-initiated trap). **Hardverske interapte** stvaraju spoljni dogadjaji, kao što je ulazno/izlazni uredjaj ili sat koji prijavljuje prolazak vremena. Hardverski interapti se javljaju *asinhronizovano* i nisu vezani za proces koji se trenutno izvršava. Za manipulisanje hardverskim interaptom odgovoran je odgovarajući device-driver **interupt handler** u kernelu.

Hardverska zamka je vezana za proces koji se trenutno izvršava i može biti ili sinhronizovana ili asinhronizovana. Javljaju se usled neke greške kao npr. deljenje nulom. Proces sazna za grešku ili tako što primi signal, ili tako što bude ugašen.

Softverske zamke se koriste da zakažu što skorije izvršavanje nekih dogadjaja (mrežna komunikacija, vremensko planiranje izvršavanja...) i javljaju se sinhronizovano. Za softverske i hardverske zamke odgovoran je deo kernela pod imenom **trap handler**.

Sistemski pozivi su specijalni slučaj softverske zamke i za njih je odgovoran deo kernela pod imenom **syscall handler**. Nakon izvršavanja sistemskog poziva programu se šalje informacija o tome da li je sistemski poziv uspešno izvršen i identifikator greške ukoliko se javila, posle čega kernel odlučuje koji proces će nastaviti da izvršava.

Svaka aplikacija ima svoj sopstveni adresni prostor koji se razlikuje od adresnog prostora kernela. Kernel odredjuje koje će procese kada da izvršava, prekine sa izvršavanjem ili suspenduje radi izvršavanja drugog procesa.

Dizajn

Procesi

Proces je program koji je trenutno u izvršavanju. Svaki proces zauzima odredjene sistemske resurse, kao što su memorija i procesorsko vreme. Kernel pruža korisnicima iluziju *konkurentnog izvršavanja* više procesa u isto vreme rasporedjivanjem tih resursa medju procesima koji su spremni za izvršavanje.

Proces se izvršava ili u *korisničkom,* ili u *kernel-modu*. U korisničkom modu instrukcije se izvršavaju u neprivilegovanom modu hardvera. Kada proces zatraži od kernela da izvrši neki sistemski poziv, instrukcije se vrše u privilegovanom, ali kernelom ograničenom i kontrolisanom modu. Ovako su aplikacije odvojene od hardvera.

Kernel razlikuje procese prema njihovom **proces identifier (PID)** broju. Korisnici mogu kreirati procese, kontrolisati izvršavanje procesa i primiti obaveštenje kada se status izvršavanja procesa promeni.

Procesi mogu stvoriti nove pozivanjem sistemskog poziva **fork**. Njime se stvara istovetna kopija procesa; dobiju se dva ista programa i sa istim promenljivama i istim vrednostima. Oba procesa (roditelj i dete) nastavljaju sa izvršavanjem instrukcije koja sledi posle pozivanja fork-a, s tim što postoji jedna razlika: detetu fork vraća vrednost nula, a roditelju vraća vrednost PID-a deteta (koja je različita od nule).

Obično posle pozivanja fork-a jedan od ova dva procesa poziva sistemski poziv **execve**, kojim se vrši prepisivanje postojeće memorijske slike procesa novom, koja se čita iz binarnog fajla, posle čega počinje izvršavanje novog programa.

Proces može da završi svoje izvršavanje pozivanjem sistemskog poziva **exit**. Ako je njegov roditelj pozvao sistemski poziv **wait**, PID deteta se signalom šalje roditelju tako da on može da zna koje dete je završilo sa izvršavanjem. Ako se dete-proces sruši, sistem simulira exit sistemski poziv i obaveštava roditelja. Prilikom izlaženja, dete-proces postaje*zombi proces*, koji se ne izvšava, ali postoji u memoriji, kako bi roditelj mogao, koristeći **wait4** sistemski poziv, da dobije više informacija o detetovom izvršavanju.

Multiprogramiranje

Multiprogramiranje predstavlja iluziju paralelnog izvršavanja više procesa ili programa. Kernel to omogućava pomoću smenjivanja konteksa procesa (*context switching*). Kontekst procesa obuhvata sve informacije koje kernel koristi za izvršavanje procesa (PID, prioritet, stanje izvršavanja, data, stack i text segmente, stanje signala, akcije na signale, tajmere, deskriptore...). Kernel odlučuje koji će proces sledeći da izvršava - ovo se naziva scheduling - vremensko planiranje.

Scheduling

Zadatak kernela je da isplanira i zakaže izvršavanje procesa tako da svi procesi mogu da zauzmu podjednako resursa. Programi se ocenjuju prema količini računanja i količini I/O koje izvršavaju i na osnovu toga im se odredjuje *prioritet*. On se u FreeBSD-u izračunava svake sekunde, na osnovu količine računanja u prethodnom periodu, količine memorije koju zauzima, ukupne zauzetosti resursa sistema i vrednosti **nice** koju ima svaki proces i koja označava poželjni prioritet. Svi procesi naizmenično dobijaju odredjen vremenski period izvršavanja. U FreeBSD-u ovaj period je 0,1 sekunda, posle čega kernel prelazi na izvršavanje procesa sa najvišim prioritetom.

Prioritet procesa

Procesi koji čekaju da se I/O operacija završi imaju veći prioritet izvršavanja, jer se izvršavaju u kernel-modu. Nasuprot tome, procesima koji zahtevaju dosta računanja (procesorskog vremena) prioritet se smanjuje, kako ne bi zauzeli kompletan procesor i zaustavili izvršavanje ostalih procesa. Na ovaj način 4.4BSD kernel daje prednost *interaktivnim programima* (kakvi su npr. tekst editori) nad dugotrajnim *batch-type* procesima koji zahtevaju dosta procesorskog vremena i koji zagušuju sistem.

Ograničavanje resursa

Kernel podržava ograničavanje količine resursa koji su dostupni pojedinačnim procesima. Ova ograničenja mogu biti: maksimalno CPU vreme koje proces može da zauzme, količina memorije koju može da alocira, veličina i broj fajlova koje proces može da kreira, broj simultanih procesa koje korisnik može da pokrene i dr.

Limiti, soft i hard

Za svako ograničenje postoje **soft** (mekani) i **hard** (čvrsti) limit. Kada proces predje soft limit, stiže mu signal koji obično rezultira prekidanjem izvršavanja procesa, ali proces može i da ignoriše ovaj signal i pokuša da oslobodi resurse. Ukoliko to ne uradi sledeći pokušaj alociranja još resursa neće uspeti.

Hard limit se ne može prekoračiti. Svi korisnici mogu svojim procesima da smanje hard limit, a jedino superuser može da ga poveća.

Sistemski procesi

Svi procesi u 4.4BSD-u (i naravno FreeBSD-u) potiču od jednog procesa kojeg startuje kernel. Prilikom startovanja sistema kreiraju se tri procesa i oni uvek postoje i izvršavaju se. Procesi swapper i pagedaemon su procesi koji se izvršavaju u kernel modu. **Swapper** (PID 0) je odgovoran za vremensko planiranje premeštanja procesa iz RAM memorije u sekundarni memorijski prostor (swap) ukoliko sistem ima manjak memorije. **Pagedaemon** (PID 2) je odgovoran za ispisivanje delova adresnog prostora procesa na swap. **Init** (PID 1) je proces koji se izvršava u korisničkom modu, van kernela i koji je odgovoran za administrativne zadatke, kao što je pokretanje drugih procesa (po jedan *getty* proces za svaki terminal mašine) zatvaranje procesa prilikom gašenja sistema (ulaskom u beskonačnu petlju ili restartovanjem sistema).

Identifikatori korisnika

Važna odgovornost operativnog sistema je da pruži mehanizam za kontrolu i ograničenje pristupa privilegovanim operacijama kernela (npr. gašenje sistema ili slanje mail-a). U FreeBSD-u ovaj mehanizam se zasniva na razlikovanju korisnika i grupa korisnika sistema. Svaki korisnik ima jedinstveni *user identifier (UID)*, 32-bitni broj koji mu je pripisao sistemski administrator. Korisnici su svrstani u grupe, koje se označavaju *GID (group identifier)* brojem. Jednom korisniku koji se naziva **superuser** dostupne su sve operacije kernela, dok su mnoge od njih nedostupne običnim korisnicima.

Svakom procesu su pripisani UID i GID korisnika koji ga je pokrenuo, tako da obični korisnici mogu samo svojim procesima da šalju signale. Svaki fajl ima pristupne dozvole (*access permissions*) za vlasnika fajla, grupu korisnika kojoj pripada vlasnik i ostale korisnike. Ove dozvole uključuju čitanje (**read**), prepravljanje (**write**) i izvršavanje fajla (**execute**). Ovako se obezbedjuje kontrola pristupa fajlovima.

Setuid i setgid

U UNIX sistemima postoje **setuid** i **setgid** programi koji omogućavaju da korisnik kontrolisano izvši neke operacije koje mu nisu dozvoljene. Ovakvi programi moraju biti veoma pažljivo napisni kako ne bi došlo do zloupotreba privilegija.

Manipulisanje memorijom

Centralni deo operativnog sistema zauzima sistem za manipulisanje memorijom (*memory management system*). On je zadužen za kontrolisanje memorijskih resursa koji su dostupni mašini. Primarna (glavna) memorija je RAM memorija računara, a sekundarna je svaki drugi medijum za čuvanje podataka (hard disk, cdrom...).

Svaki program ima svoj privatni adresni prostor. On se sastoji iz data, text i stack segmenata. **Data segment** sadrži inicijalizovane i neinicijalizovane podatke programa. **Text segment** sadrži mašinske instrukcije i dele ga svi procesi koji izvršavaju taj fajl, dok su data i **stack segmenti** privatni za svaki proces. U stack segmentu se nalazi run-time stack programa i njega kernel automatski proširuje dok se program izvršava. Data segment može biti proširen ili skupljen pozivanjem sistemskog poziva, dok veličina i sadržaj text segmenta mogu jedino biti promenjeni prilikom pozivanja sistemskog poziva *execve*, kada se postojeća memorijska slika prepisuje nekom drugom i stvara se novi proces.

Prilikom započinjanja izvršavanja nekog programa, kernel prvo učita text segment programa na početak adresnog prostora. Posle toga kernel mapira inicijalizovane podatke data segmenta i nulira ostatak adresnog prostora (koji je veličine neinicijalizovanih podataka i stack-a). Iako to nuliranje nije neophodno (čak je i suvišno jer usporava učitavanje programa) rani UNIX sistemi su to ostavili u nasledje (mnogi programi računaju na to da je početna vrednost promenljive nula).

Kopiranje celog text segmenta i inicijalizovanih podataka u memoriju mnogo usporava proces inicijalizovanja programa. 4.4BSD ovo izbegava korišćenjem **straničenja na zahtev** (*demand paging*) koje obezbedjuje da se program učitava u malim delovima - stranicama prvi put kada one postanu potrebne. Adresni prostor se deli na **stranice** jednakih veličina, a kernel vodi računa o adresama stranica koje su učitane. Kada program pokuša da pristupi stranici koja nije učitana javi se *page-fault trap* u kernelu, page-fault trap handler učita dotičnu stranicu i program nastavlja sa izvršavanjem. Ovako se u memoriju učitavaju samo delovi programa koji se koriste.

Usled primene multiprogramiranja neophodno je da operativni sistem efektivno deli memorijske resurse medju procesima. Ceo adresni prostor procesa ne mora u svakom trenutku biti prisutan u primarnoj memoriji da bi se proces izvršavao. Ukoliko su memorijski resursi sistema niski, kernel sprovodi neki od mehanizama za oslobadjanje resursa. Kernel može izbaciti stranice koje nisu dugo bile upotrebljivane (**paging**), ili može ceo kontekst procesa prebaciti u sekundarnu memoriju (**swapping**). Straničenje na zahtev i zamena stranica su transparentni za procese, ali procesi mogu pružiti sistemu informacije o budućim memorijskim zahtevima, radi poboljšanja performansi.

Dobra strana memory management-a je to što se omogućava da postoji više procesa nego što može da se smesti u primarnu memoriju, a loša je ta što se izvršavanje procesa usporava. Sistem mora da se pobrine da minimizuje *trashing*. To je pojava kada sistem više vremena troši da izvrši zamenu stranica, nego što troši na korišćenje tih stranica. Sistem detektuje trashing tako što posmatra količinu slobodne memorije. Ukoliko postoji mali broj slobodnih stranica, a veliki je broj zahteva za alociranje memorije, sistem blokira izvršavanje odredjenog broja procesa koji zauzimaju dosta resursa i prebacuje ih u sekundarnu memoriju. Kada zagušenje prodje, blokirani procesi se vraćaju u glavnu memoriju i nastavljaju sa izvršavanjem.

I/O System

Osnovni model UNIX-ovog ulazno/izlaznog sistema je **sekvenca bajtova** kojoj se može pristupiti bilo **nasumično** (*random access*) bilo **sekvencijalno** (*sequential acces*).

Različiti programi rade sa različitim strukturama podataka, ali ih kernel ne razlikuje. Uobičajeno je da se tekstualni fajlovi posmatraju kao niz linija ASCII karaktera koje se završavaju karakterom za novu liniju. Sto se tiče kernela svi fajlovi predstavljaju **ulazno/izlazni tok bajtova** (*I/O stream*).

UNIX procesi koriste **deskriptore** za pristup I/O stream-u. Deskriptori su mali, neoznačeni celi brojevi koje vraćaju *open* i *socket* sistemski pozivi. Deskriptori predstavljaju odgovarajuće objekte koje kernel podržava.

Sistemski poziv *open* se koristi za kreiranje novog, praznog fajla, ili za otvaranje postojećeg. Prenos podataka iz i u objekat kog deskriptor predstavlja obavlja se pomoću *read* i *write* sistemskih poziva, a sistemski poziv *close* se koristi sa oslobadjanje deskriptora.

Deskriptori mogu da predstavljaju file, pipe i socket.

Fajl
: Niz bajtova. On ima svoje ime i postoji dokle god postoji referenca na njega u fajl sistemu ili dok god neki proces ima otvoren deskriptor koji ga predstavlja. I/O uredjajima se pristupa kao fajlovima.

Pipe
: Takodje niz bajtova, ali se, za razliku od fajlova koristi kao ulazno/izlazni tok podataka u jednom smeru. Nema ime, tako da ne može biti otvoren open sistemskim pozivom. Umesto njega koristi se sistemski poziv *pipe*. Kernel takodje podržava **named pipe** (FIFO - First In First Out) - pipe koji postoji u fajl sistemu. Kada dva procesa žele da komuniciraju oni otvore FIFO, jedan za čitanje, jedan za pisanje.

Socket
: Privremeni objekat koji se, kao i pipe koristi za *medjuprocesnu komunikaciju*. On postoji samo dok postoji otvoren deskriptor kojim je predstavljen. U suštini socket je generalizovan dvosmerni pipe. Socket pruža mogućnost komunikacije izmedju dva procesa koji se izvršavaju na različitim mašinama, čak i pod različitim operativnim sistemima.

Svakom procesu kernel prilikom startovanja obezbedjuje tri deskriptora: **standardni ulaz** (0), **standardni izlaz** (1) i **standardna greška** (2). Obično ova tri deskriptora predstavljaju korisnikov terminal. Program prima podatke čitajući standardni ulaz (tastaturu), a izlaz i greške ispisuje po ekranu. Ovi deskriptori mogu da predstavljaju i druge objekte osim tastature i ekrana, npr. fajlove. Ovo se zove *I/O preusmeravanje*. Pipe je način preusmeravanja izlaza jednog programa u ulaz drugog. Ako jedan program preusmeri izlaz, a drugi ulaz na pipe dobija se pipeline (cevovod).

Svi hardverski uredjaji su predstavljeni fajlovima i programi im mogu pristupiti koristeći iste sistemske pozive kao i za obične fajlove. Terminalima, štampačima i drugim sličnim uredjajima pristupa se preko I/O toka. Hardverski uredjaji se mogu podeliti na struktuirane i nestruktuirane. Nestruktuirani (*character*) uredjaji obradjuju znak po znak (komunikacione linije), a struktuirani (*block*) obradjuju vece grupe podataka od jednom, na primer, blokove od 512 bajtova (diskovi, magnetne trake, cdrom-ovi).

Dizajn

Fajl sistem

Fajlovi mogu biti obicni fajlovi, direktorijumi i specijalni fajlovi.

Obični fajl je niz bajtova. Velicina fajla je jednaka ukupnom broju znakova koje fajl sadrzi. Fajlu se može pristupiti (čitati ili pisati) počev od bilo kog bajta. Podaci unutar fajla nemaju nikakvu strukturu osim one koju sam korisnik definiše. U samom fajlu se ne nalaze nikakve informacije o tom fajlu, već su one u fajl sistemu.

Fajl sistem čuva sledeće informacije o fajlovima: ime fajla, jedinstven broj fajla nazvan inode, velicinu u bajtovima, vreme kreiranja, vreme poslednje izmene, vreme poslednjeg pristupa, pristupne dozvole, vlasnika fajla i grupu kojoj pripada vlasnik fajla.

Direktorijumi daju hijerarhijsku organizaciju celog fajl sistema. Direktorijum su obični fajlovi, ali poseduju strukturu. Direktorijum je lista imena fajlova i drugih direktorijuma koji se nalaze u njemu. Kernel ne dozvojava procesima da prepravljaju direktorijume.

Specijalni fajlovi se nalaze u /dev direktorijumu i predstavljaju hardverske uredjaje.

Fajl sistem UNIX operativnog sistema je hijerarhijski struktuiran u vidu stabla. Na vrhu stabla je direktorijum koji se naziva korenom (**root**) i označava se kosom crtom **(/)**.

Procesi identifikuju fajlove po njihovoj putanji i imenu. **Putanja** (put) je niz imena direktorijuma na čijem se kraju nalazi ime ciljnog direktorijuma ili fajla. Sva imena u putanji su razdvojena kosom crtom. Ako putanja počinje kosom crtom znači da je formirana od korena, to jest od root direktorijuma, i tada se naziva potpunim ili **apsolutnim putanjom**. Putanja se takodje može kreirati relativno u odnosu na radni (working)-tekući direktorijum i tada se naziva **relativna putanja** i ne počinje kosom crtom. Svi direktorijumi i fajlovi, osim root direktorijuma, imaju neposrednog prethodnika (parent directorijum) koji je označen sa "..", dok je radni direktorijum označen sa ".".

Iako postoji više fajl sistema, svi su smešteni pod root fajl sistemom. **Root fajl sistem** je uvek prisutan, dok se fajl sistemi sa drugih uredjaja (diskova, cdrom-ova) montiraju pod neki direktorijum na root fajl sistemu. Montiranje se obavlja eksplicitno, bilo pri startovanju sistema, bilo na korisnikov zahtev. FreeBSD pruža korisnicima jednu olakšicu što se ovoga tiče - fajl sistem se može montirati dinamički svaki put kada mu korisnik pristupi.

Sistemski poziv *link* stvara još jedno ime kojim se jednom fajlu može pristupiti. Fajl se može ukloniti sistemskim pozivom *unlink*. Fajl će biti izbrisan kada se ukloni i poslednje ime kojim se on može pristupiti i kada se zatvori poslednji deskriptor koji je neki proces koristio za pristupanje fajlu.

Tipičan fajl sistem FreeBSD-a

bin/
: programi

boot/
: konfiguracioni fajlovi boot-manager-a

cdrom/
: direktorijum pod kojim se montira cdrom

compat/
: biblioteke koje omogućuju izvršavanje linux programa

dev/
: specijalni fajlovi koji predstavljaju uredjaje

etc/
: konfiguracioni fajlovi sistema

fat/
	direktorijum pod kojim se montira ms-dos particija
home/
	direktorijumi korisnika
mnt/
	za montiranje ostalih fajl sistema (floppy)
modules/
	moduli koji se mogu dinamički učitati u kernel
proc/
	virtualni fajl sistem koji prikazuje stanje procesa
root/
	superuser-ov home direktorijum
sbin/
	sistemski programi
stand/
	instalaciona alatka (sysinstal)
sys/
	izvorni kod kernela i programa
tmp/
	privremeni fajlovi
usr/
	korisnički programi, kolekcija portova
var/
	razne informacije (o instaliranim portovima, logovi...)
kernel
	kernel binarni fajl

Direktorijum /usr sadrži sve korisničke programe, kolekciju portova, izvorni kod sistema i programa i drugo: X11R6/, home/, libexec/, sbin/, bin/, include/, local/, share/, compat/, lib/, obj/, src/, games/, libdata/, ports/, tmp/.

Dizajn

Mrežni fajl sistem

*Network file system (**NFS**)* predstavlja standardni način deljenja fajlova medju UNIX sistemima. Svrha NFS-a je da omogući korisnicima pristup fajl sistemu nekog drugog računara tako što se on montira na lokalni fajl sistem. Lokalni korisnici ovako mogu da promene aktivni direktorijum u direktorijum na mrežnom fajl sistemu, mogu da čitaju, prepravljaju i izvršavaju fajlove isto kao fajlove na lokalnom fajl sistemu.

Kada lokalni klijent zatraži neku operaciju nad udaljenim fajl sistemom, zahtev se kompresuje (zbog performansi) i šalje serveru. Server izvršava zatraženu operaciju ili prijavljuje grešku i razlog odbijanja servisa. Zbog performansi klijent čuva podatke o podacima kojima se često pristupa.

NFS protokol i implementaciju razvio je Sun Microsystems.

Dizajn

Startovanje FreeBSD-a

Startovanje kompjutera se naziva **bootstrap**, ili skraćeno **boot**.

Na x86 platformi *Basic Input/Output System (BIOS)* je odgovoran za pokretanje kompjutera. Na hard disku koji se boot-uje, BIOS pronalazi *Master Boot Record (MBR)*. Ukoliko na računaru postoji samo jedan operativni sistem, onda je standardni MBR sasvim dovoljan za pokretanje sistema. Takav MBR će pronaći prvi boot slice (butabilnu particiju) na disku i pokrenuće kod koji učitava ostale delove operativnog sistema. Ukoliko je na računaru instalirano više sistema, onda će MBR morati da pita korisnika koji sistem da startuje. Ovakav MBR se naziva **boot-manager**. BootEasy je zvanični boot-manager FreeBSD-a. Nalazi se u MBR-u, ali i u /boot/**boot0** fajlu. Za razliku od većine drugih boot-manager-a, njemu nije neophodno podesiti šta sve može da boot-uje, već će pri startovanju sam pronaći sve sisteme koji mogu da se startuju.

Dalje učitavanje FreeBSD-a je zbog ograničenja koje postavlja PC hardver, podeljen u tri faze: boot1, boot2 i loader.

boot1
: /boot/boot1 je jednostavan program, velik samo 512 bajtova. On može da pročita informacije o particijama FreeBSD-a i može da pokrene boot2.

boot2
: /boot/boot2 može da pristupi FreeBSD-ovom fajl sistemu i startuje loader (ili kernel direktno).

loader
: /boot/loader - vrši učitavanje modula kernela zajedno sa kernelom. Pre startovanja kernela korisnik može da pokrene komandni interpreter kojim bi podesio koje module treba da učita i koji kernel da startuje.

Posle ove tri faze kernel se startuje i počinje sa ispitivanjem i inicijalizovanjem hardvera. Kada završi sa tim, pokreće se prvi korisnički proces **/sbin/init**. Ako je prilikom boot-ovanja prosledjen odredjeni parametar, ili ukoliko init otkrije neispravnost fajl sistema, FreeBSD se startuje u **Single-User** modu radi otklanjanja grešaka. Ukoliko nije, FreeBSD se startuje u **Multi-User** modu. Prvo se startuje shell, zatim se čita konfiguracioni fajl **/etc/rc.conf**, montiraju se fajl sistemi koji su podešeni u **/etc/fstab** fajlu, startuju se mrežni servisi, sistemski daemoni. Posle toga init startuje po jedan **/usr/libexec/getty** program za svaki terminal sistema. On čita sa terminala korisničko ime, a zatim pokreće **/usr/bin/login** program koji proverava ispravnost šifre i ako je ispravna pokreće korisnikov shell.

Dizajn

Kompajliranje kernela

Kompajliranje kernela je jedna od najznačajnijih stvari koje bi trebalo da se urade pošto se FreeBSD instalira. Kernel koji se snima prilikom instalacije je GENERIC kernel, što znači da je podešen da podržava veliki broj hardverskih komponenti. Kompaliranjem kernela dobija se kernel koji je optimizovan za konkretan sistem, ostvaruje se brže startovanje sistema, manja zauzetost memorije, pruža se podrška dodatnom hardveru koji nije dostupan u GENERIC kernelu.

Izvorni kod

Da bi se kernel prekompajlirao treba imati izvorni kod kernela. On se nalazi u /usr/src/sys direktorijumu, ili /sys. Konfiguracioni fajl kernela se nalazi u /usr/src/sys/i386/conf. Posle podešavanja potrebnih stavki

sledi - kompajliranje.

Ukoliko korisnik želi da unapredi verziju FreeBSD-a koju koristi, za to postoji odličan mehanizam. Sve što treba da uradi je da snimi najnoviji izvorni kod, ili unapredi postojeći pomoću CVS-a, i pokrene jednu komandu - *make world*. Ovim će ceo sistem biti prekompajliran.

KorisniČki interfejs

Shell

Shell (ljuska) je omotač, ljuska oko jezgra operativnog sistema, kernela. U UNIX sistemima se dosta posla obavlja iz **komandne linije** (*command line*). Korisnički i sistemski programi se (obično) pokreću komandnim interpreterom. U UNIX-u je shell korisnički proces čiji je zadatak da prima komande od korisnika i da ih izvršava.

FreeBSD dolazi sa barem dva shell-a: bash i tcsh, a dosta drugih shell-ova se može instalirati iz kolekcije portova.

Najčešćih shell-ovi

Bourne-Shell
: **sh** - jedan od prvih shell-ova koji su napisani. I danas se dosta koristi, pogotovo za pisanje skripti (za startovanje i gašenje sistema).

C-Shell
: **csh** - standardni shell Berkeley-evih distribucija, prvi shell sa istorijom komandi i sintaksom u stilu C-a. Napisao ga je Bil Joy.

Korn-Shell
: **ksh** - standardni shell UNIX System V, prvi shell sa editovanjem komandne linije korišćenjem drugih programa (vi, emacs).

Boune-Again-Shell
: **bash** - moderni naslednik sh-a, koristi se za skripte

Turbo-C-Shell
: **tcsh** - moderni naslednik csh-a, lako prilagodljiv.

Shell je takodje i programski jezik visokog nivoa. U njemu se mogu pisati **skripte** koje se mogu pokretati kao obične komande.

Na svakom BSD sistemu se javljaju sh i csh. sh se koristi za sistemske skripte kao što je /etc/rc, a csh je zvanični shell koji koristi root. Korn-Shell se javlja jedino u System V UNIX sistemima.

Komandni interfejs je dosta popularan u UNIX sistemima zbog lakoće obavljanja mnogih zadataka. Skripte omogućavaju automatizovanje tih zadataka.

C shell FreeBSD sistema pruža **kontrolu poslova** (*job control*) koja omogućava prebacivanje poslova iz prednjeg u zadnji plan, stopiranje i restartovanje poslova. Ovo je našlo primenu i u X-Windows-u, grafičkom korisničkom interfejsu gde se svaki prozor smatra terminalom i može postojati više programa koji će biti u prvom planu.

Korisnicki interfejs

X Window System

X Window System je veliko i moćno (neki kažu preveliko i previše složeno) grafičko okruženje UNIX sistema. Razvijen je na MIT-u, a komercijalni proizvodjači su ga postavili kao standard na UNIX sistemima. Danas velika većina UNIX radnih stanica koristi neku varijantu X Windows-a.

XFree86

Besplatni port MIT-ovog X Windows-a, verzije 11, izdanja 6 (X11R6) za x86 računare koji rade pod UNIX sistemim se naziva **XFree86** i može se instalirati iz kolekcije portova FreeBSD-a.

KDE i **GNOME** su desktop okruženja koja se najčešće koriste. Oni predstavljaju kolekciju programa, alatki i igara koje korisnici mogu da koriste.
