Generalni pregled UNIX sistema

- Pojavio se 1969
- Od svoje pojave , UNIX je postao krajnje popularan
- funkcioniše na velikom broju računara - CPU
  - sa različitom procesorskom snagom
  - od mikroprocesora
  - do mainframe mašina
- OS2 će obuhvatiti pregled UNIX System V + (Linux)

Istorijat

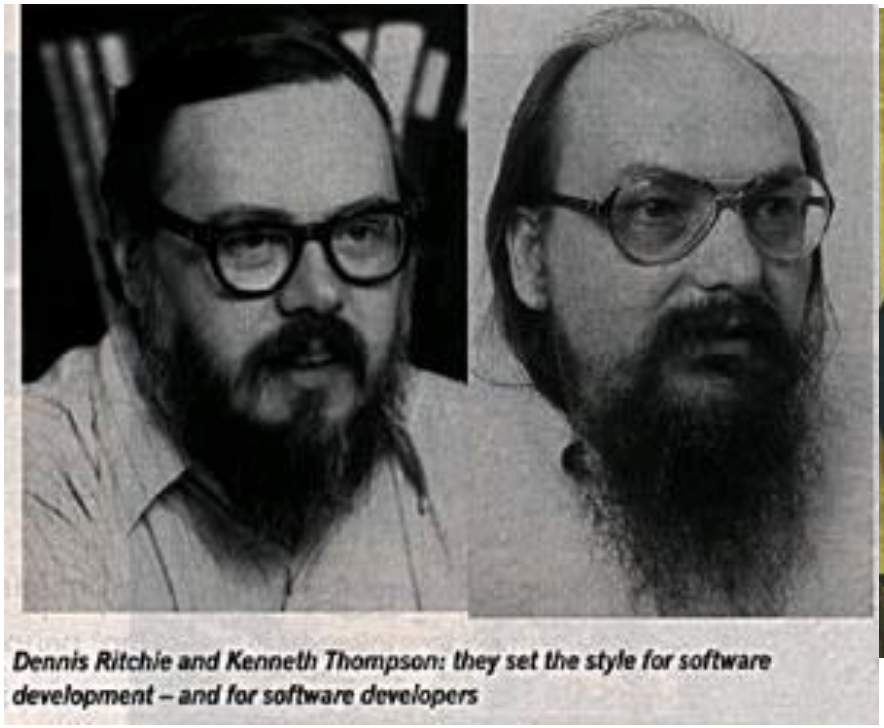
- **1965 Bell + General Electric Company + Project MAC**
  - razvijali su novi operativni system **MULTICS**
  - (**Multiplexed Information and Computing Service**)
  - sa ciljem da to bude moćan OS za veliki broj korisnika.
- Iz tog projekta nastao je **OS UNIX, 1969**,
  - **Ken Thomson, Dennis Ritchie**
  - najpre za processor **PDP-7,**
  - a potom za **PDP-11** 1971,
  - **sa skromnim hardverskim mogućnostima.**
    - (16K za UNIX, 8K za korisničke programe, disk veličine 512K, file limit 64K)
- **Dennis Ritchie**
  - je napisao novi PL koga je nazvao C,
  - koji realizuje mašinski kod, deklaraciju tipova podataka i definiciju struktura podataka.
- **1973. godine, prepisali su UNIX na C**.
- Kasnije se UNIX razvijao:
  - **UNIX System III**
  - **UNIX System V**
  - **BSD UNIX** (paralelna generacija UNIX-a)
- Ovde će se uglavnom obrađivati UNIX System V

Generation 3 (1965-1989) Integrated Circuits

- **DEC PDP-11 - A 16-bit successor to the PDP-8 also utilizing byte addressable memory.** The PDP series was an extremely popular computer due to low cost and good performance. Its 16-bit address space caused it to decline in popularity as large memories became cheaply available with the introduction of VSLI technology. This picture is of Dennis Ritchie and Kenneth Thompson, creators of the UNIX operating system, at a PDP-11. (1970)

Dennis Ritchie and Ken Thompson

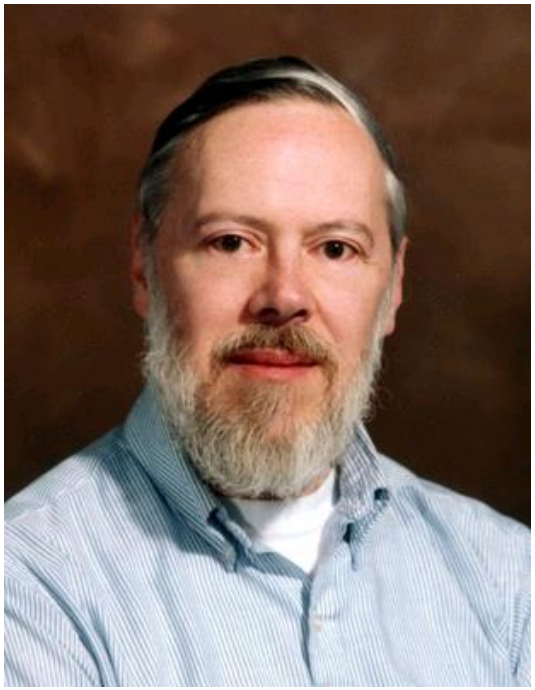
- WASHINGTON, D.C. (April 27, 1999) -- Dennis Ritchie and Ken Thompson of Bell Labs received the U.S. National Medal of Technology today from President Bill Clinton at ceremonies televised at the White House.



Dennis Ritchie and Kenneth Thompson: they set the style for software development – and for software developers

Dennis Ritchie and Ken Thompson

	Dennis Ritchie	Kenneth Thompson
Name:	Dennis Ritchie	Kenneth Thompson
Born:	1941	1943
Country:	USA	USA
Specialism:	Software	Software
Theories:	None	None
Products:	Unix, C and more	Unix, C and more
Companies:	None	None



In Memoriam:
Dennis Ritchie, father of Unix and C, dies in 2011

- **NEWS**
- **Dennis Ritchie, creator of the C programming language and co-creator of the Unix operating system, has died aged 70.**

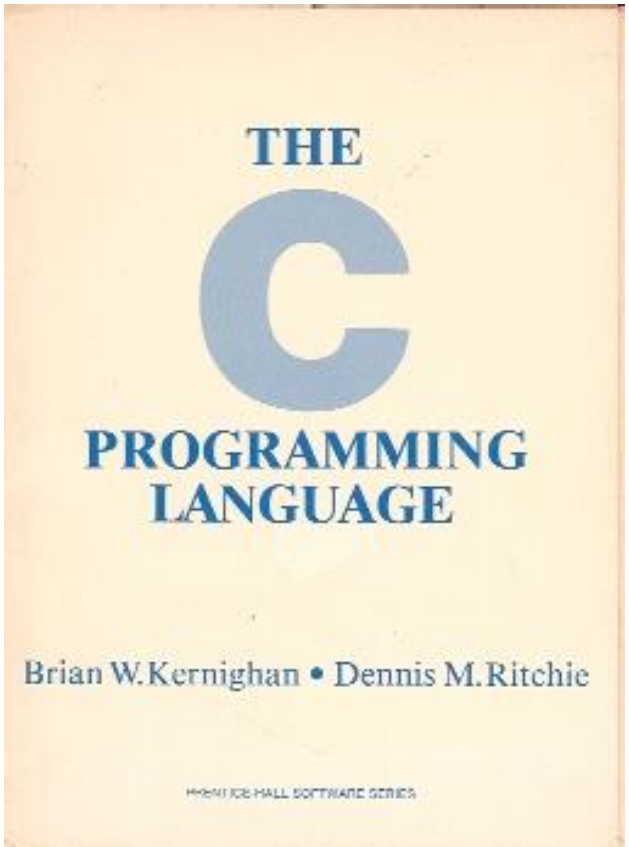
- **Dennis Ritchie, creator of the C programming language and co-creator of the Unix operating system, has died aged 70.**

In Memoriam:
Dennis Ritchie, father of Unix and C, dies in 2011

- Dennis MacAlistair Ritchie was born in Bronxville, New York, on 9 September, 1941, and grew up in New Jersey, where his father, Alistair Ritchie, worked as a switching systems engineer for Bell Laboratories. Ritchie went to Harvard University and received his degree in Physics in 1963.
- It was at Harvard that Ritchie first encountered a computer, attending a lecture on Univac 1 that captured his imagination. He moved to the Massachusetts Institute of Technology, where the first shifts away from the mainframe to smaller, cheaper computers were being ardently investigated, and thence in 1967 to Bell Labs — birthplace of the transistor and, at the time, one of the most important centres of digital innovation in the world.

In Memoriam:
Dennis Ritchie, father of Unix and C, dies

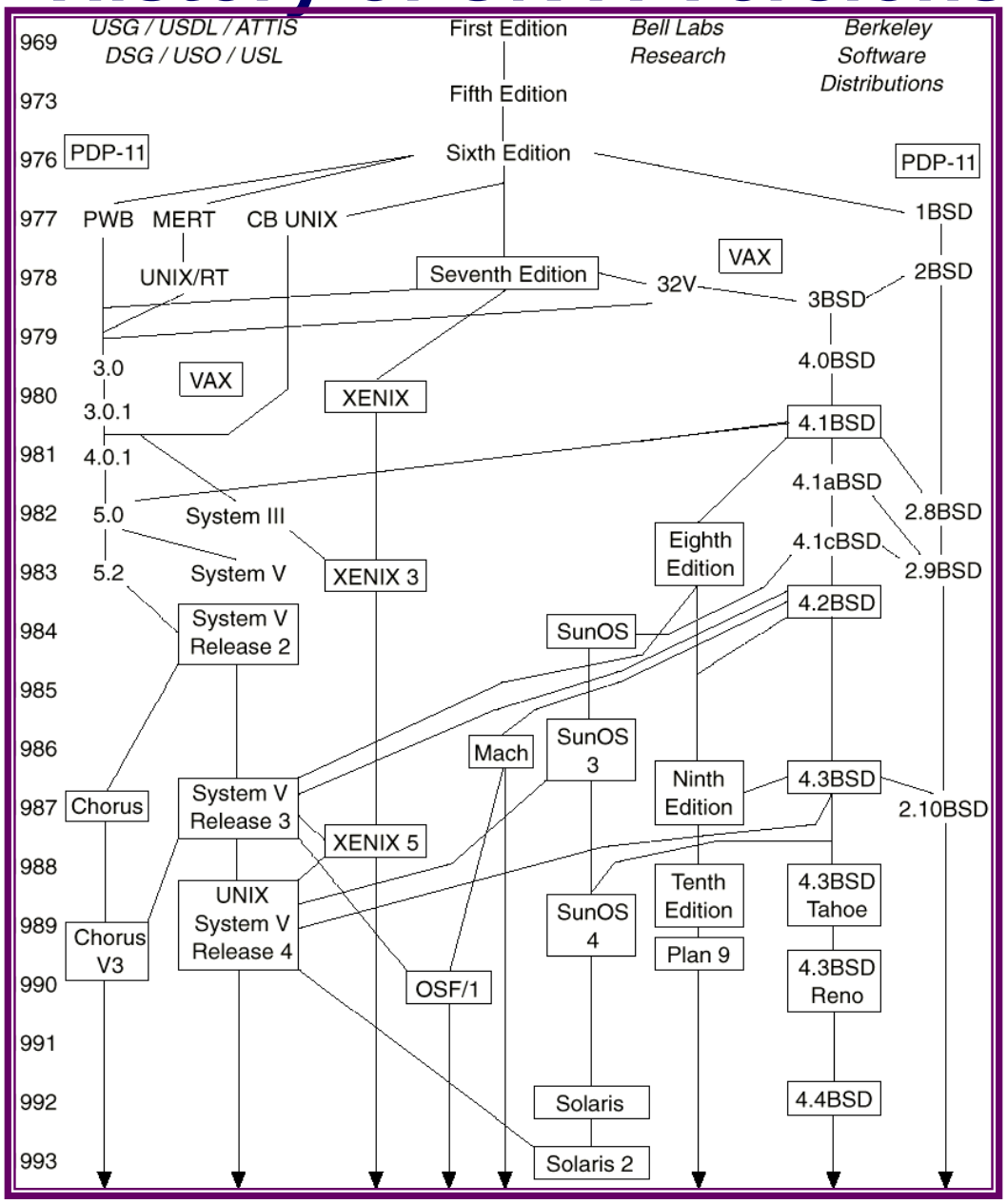
- **The C Programming Language, also known as K&R, was published in 1978. It was a peerless introduction to the techniques of programming in C.**



Poreklo imena-Name Origin

- **MULTICS**
  - **Multiplexed Information** and **Computing Service**
- **UNICS**
  - **Uniplexed Information** and **Computing Service**
- **UNIX**
  - **Radi lakšeg izgovora i pisanja**
  - **ime UNICS je kasnije evoluiralo u UNIX**

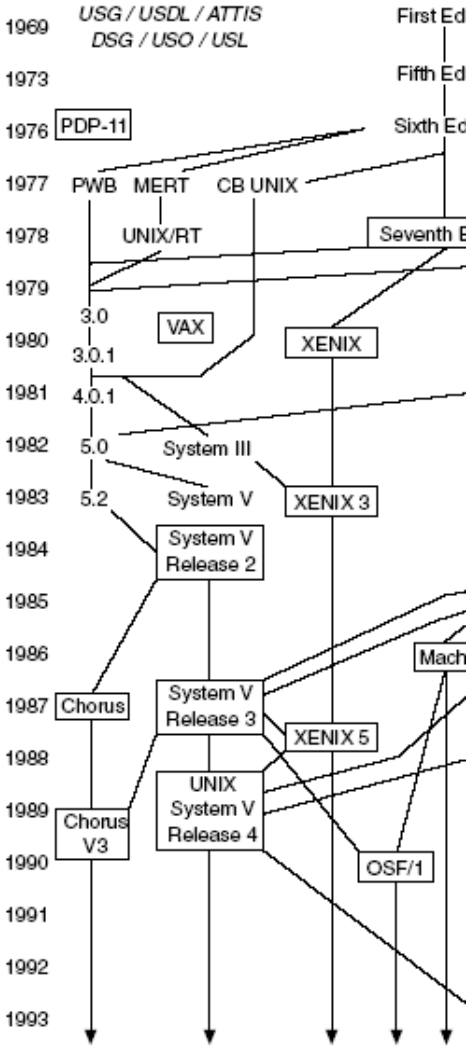
History of UNIX Versions



UNIX: značajne verzije

- **Prvi UNIX razvijen 1969**
  - glavni kreatori: **Ken Thompson** & **Dennis Ritchie**
  - **Research Group** iz **Bell Laboratories**
  - **sa ugrađenim funkcijama drugih OS**
  - **posebno MULTICS OS**
- **III UNIX verzija napisana u** C jeziku
  - **koji je takođe razvijen u Bell Labs**
  - **sa posebnom podrškom za UNIX**
- **Najuticajniji UNIX**
  - **koji ne pripada Bell LAbs (non-Bell Labs)** and
  - **koji ne spada u familiju AT&T UNIX (non-AT&T UNIX development groups)**
  - je **BSD UNIX**, **University of California at Berkeley (Berkeley Software Distributions)**
- **4BSD UNIX je nastao iz DARPA zahteva**
  - **da se razvije standard UNIX OS**
  - koji će se koristiti za SAD vladu (USA for **government)**
- **Razvijen za VAX CPU**
  - **4.3BSD je jedna od najuticajnijih verzija**
  - **portiran je na mnoge druge platforme**.

AT&T UNIX


1969
1973
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
USG / USDL / ATTIS
DSG / USO / USL
First Ed
Fifth Ed
Sixth Ed
PDP-11
PWB
MERT
CB UNIX
UNIX/RT
Seventh E
3.0
3.0.1
4.0.1
5.0
5.2
VAX
XENIX
System III
System V
XENIX 3
System V
Release 2
Mach
Chorus
System V
Release 3
XENIX 5
UNIX
System V
Release 4
Chorus
V3
OSF/1

UNIX: Prva verzija

- **Prvu verziju UNIX-a je 1969. godine**
  - razvio **Ken Thompson** iz istraživačke grupe Bellovih laboratorija,
  - da bi se koristio na računaru **PDP-7**
    - (računar koji nije imao neku naročitu primenu u to vreme).
  - Ubrzo, njemu se pridružio **Dennis Ritchie**.
  - Thompson, Ritchie i drugi članovi istraživačke grupe su napravili prvu verziju UNIX-a.

- Ritchie je predhodno radio na MULTICS projektu, I MULTICS je imao veliki uticaj na novonastali operativni sistem. Čak je i ime UNIX igrom reči nastao od MULTICS.
  - Osnovna organizacija fajl sistema
  - ideja da komandni interpretator bude korisnički proces
  - korišćenje zasebnog procesa sa svaku komandu
  - originalni način editovanja karaktera
    - (# da se izbriše poslednji karakter a @ za brisanje celog reda)
  - brojne druge osobine dolaze dirketno od MULTICS-a

- Takođe su korišćene ideje iz nekih drugih operativnih sistema, kao npr. MIT-ov CTSS i XDS-940.

III verzija (C)

- Ritchie i Thompson su radili na UNIX-u bez publiciteta godinama. Njihov rad na prvoj verziji im je omogućio da ga prebace na **PDP-11/20 u drugoj verziji**.

- U **trećoj verziji**, ponovo su pisali većinu koda, ali sad na programskom jeziku C, a ne kao pre u asembleru. C je razvijen u Bellovim laboratorijama zbog UNIX-a. UNIX je prebačen na veće PDP-11 modele, kao što su 11/45 i 11/70. Multiprogramiranje i ostale prednosti su dodate kad je sistem ponovo napisan u C-u, i prebačen na sisteme (kao npr. 11/45) koji su imali hardversku podršku za multiprogramiranje.

- Kako se UNIX razvijao, postao je široko upotrebljivan u Bellovim laboratorijama i postepeno se preneo na nekoliko univerziteta. Prva verzija koja je bila puštena van Bellovih laboratorija je verzija 6, izbačena 1976. (Broj Verzije za prve UNIX sisteme odgovara broju UNIX-ovog programerskog uputstva – UNIX Programmer's Manual, koji je bio aktuelan u trenutku distribucije izdanja; kod i uputstva su razvijani nezavisno).

- **U 1978 je izbačena verzija 7**, koja je radila na PDP-11/70 i na Interdati 8/32, i to je predak većine modernih UNIX sistema. Tačnije, verzija 7 je uskoro puštena na ostale PDP-11 modele i na VAX kompjutere. Verzija za VAX je nosila oznaku 32V. Posle toga je nastavljen rad na UNIX projektu.

USG i UNIX SYSTEM III

- Posle distribucije verzije 7 u 1978. godini, **UNIX Support Group (USG)** je preuzela kontrolu i odgovornost za UNIX. UNIX sad postaje proizvod, a ne samo alat za istraživanje. Ipak, istraživačka grupa nastavlja da pravi svoje verzije UNIX-a, za svoje unutrašnje potrebe.
- Nastaje verzija 8, koja ima nešto što se zove **stream I/O system**, koji omogućuje fleksibilnu konfiguraciju kernelovih IPC modula. Ova verzija sadrži i RFS (remote file system), sličan Sun-ovom NFS-u. Posle toga dolaze verzija 9 i 10 (poslednja verzija postoji samo u Bellovim laboratorijama).

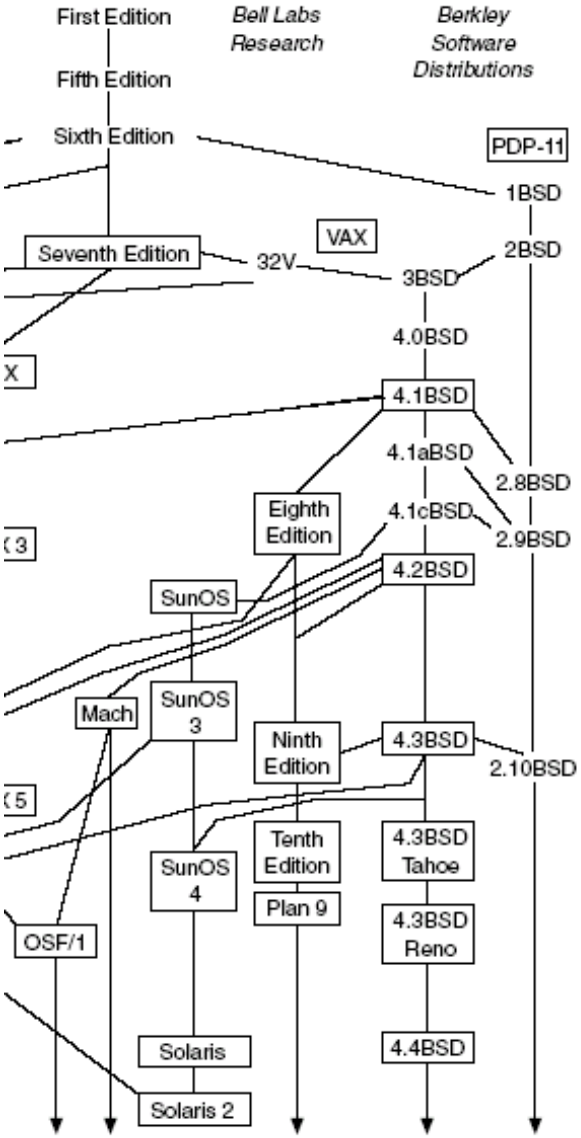
- USG je uglavnom pružao podršku za UNIX unutar AT&T.
- **Prva spoljna distribucija USG grupe je UNIX Sistem III**, 1982 godine. Sistemu III su pridružene mogućnosti iz verzija 7 i 32V, i još nekih verzija UNIX-a koji su se nezavisno razvijali. U Sistem III su uključene mogućnosti UNIX/RT, real-time UNIX sistema, kao i velikog dela PWB softverskih paketa (Programmer's Work Bench).

UNIX SYSTEM V

- **1983, USG izbacuje Sistem V**, koji je uglavnom nastao iz UNIX Sistem III. Izlazak raznih Bellovih kompanija iz AT&T-a, su naterali da AT&T agresivno marketinški gura Sistem V. Od USG-a nastaje USDL, UNIX System Development Laboratory, koji izbacuje verziju Sistema V, drugo izdanje (V.2), u 1984. A verzija Sistema V, V.2.4 dodaje novu implementaciju virtualne memorije sa copy-on-write paging i deljenom memorijom. UDSL postaje ATTIS (AT&T Information Systems), koji izbacuje V.3 verziju Sistema V, u 1987. V.3 prilagođava implementaciju iz verzije 8, ondosno stream I/O system, koji je sad dostupan kao **STREAMS**. On takođe sadrži RFS – remote file system.

- Mala veličina, modularnost, i čist dizajn ranih UNIX sistema dovodi do UNIX baziranog rada na mnogim drugim naučno-kompjuterskim organizacijama, kao što su Rand, BBN, Univerzitet Illinoisa, Harvard, Purdue i DEC. Najveći uticaj među njima je imao Kalifornijski Univerzitet u Berkliju.

Berkley UNIX-BSD



Prvi Berkley UNIX i 4BSD

- Prvi Berkley VAX UNIX nastaje 1978. kao skup sledećih mogućnosti:
  - **virtualne memorije**
  - **straničenja po zahtevu**
  - zamenu stranica kao kod 32V
  - ovo su odradili Bill Joy i Ozalp Babaoglu, da bi nastao 3BSD UNIX.
- **3BSD** je bila prva verzija UNIX-a koji je imao neku od ovih mogućnosti. Veliki prostor virtualne memorije, omogućuje razvoj veoma velikih programa, kao što je Berkley Franz LISP.
- BSD upravljanje memorijom je **ubedilo DARPA agenciju** (Defense Advanced Research Projecr Agency), **da finansira Berkley za razvoj UNIX sistema za vladine institucije; 4BSD UNIX je rezultat toga.**
- U razvoju 4BSD sistema uticali su brojni poznati ljudi iz poznavanja UNIX-a i mreža. Jedan od ciljeva ovog rada je bio da se **napravi podrška za DARPA Internet protokol (TCP/IP).** Ova podrška je urađena generalno.
- Omogućeno je da 4.2BSD komunicira između različitih tipova mreža, uključujući
  - LAN mreže (kao Ethernet i token ring) i
  - WAN mreže (kao NSFNET).
- Ta implementacija je glavni razlog za današnju popularnost ovih protokola.
- Korišćen je kao osnova
  - za neke druge implementacije drugih izdanja UNIX-a,
  - čak i za druge operativne sisteme.
- Omogućio je da Internet poraste sa 60 povezanih mreža 1984. na više od 8000 mreža, a po proceni 10 miliona korisnika u 1993.

BSD: 4.2 i 4.3

- UNIX softver kreiran na Berkley-u dobija ime BSD UNIX (Berkley Software Distributions.
  - Obično se kaže da Berklijev VAX UNIX prate 3BSD i 4BSD, iako su zapravo postojala nekoliko specificnih izdanja, a glavna među njima su 4.1BSD i 4.2BSD.
  - Oznake **2.x BSD** sistema se koriste **za PDP-11**
  - oznake **4.xBSD** za **VAX distribucije Berklijevog UNIX-a**

- 4.2BSD, izbačen 1983. godine, je bio kulminacija originalnog Berklijevog DARPA UNIX projekta. 2.9BSD je ekvivalentan verziji za PDP-11 sisteme.

- 4.3 BSD
  - U 1986. godini izašao je 4.3BSD. On je bio toliko sličan 4.2BSD-u, tako da je njegovo uputstvo jasnije objašnjavalo 4.2BSD, nego uputstva od 4.2BSD-a.
  - Na njemu nisu radili na nekim brojim unutrašnjim promenama,
  - već na ispravljanju bagova, i na unapređenju performansi.
  - Dodate su neke nove mogućnost, uključujući podršku za Xerox Network System protokole.

BSD 4.3 Tahoe, Renoe

- Sledeća verzija nosila je naziv 4.3BSD Tahoe, a puštena je 1988. Obuhvatala je brojne nove mogućnosti, kao:
  - **poboljšanu kontrolu mrežnog zagušenja**
  - **i**
  - **TCP/IP performansi**
  - Takođe konfiguracije diskova su odvojene od drajvera uređaja, tako da se sada čitaju direktno sa diskova.
  - Dodata je i podrška za vremenske zone
- 4.3BSD Tahoe je ustvari razvijen za CCI Tahoe System (Computer Console, Inc., Power 5 computer) a ne za uobičajenu VAX bazu.
- Paralelno izdanje za PDP-11 je 2.10.1BSD, koje distribuira USENIX Asocijacija, koja je i izdala uputstva za 4.3BSD.

- U verziji 4.32BSD Renoe je dodata implementacija ISO/OSI mrežnog modela

4.4BSD

- Poslednje Berklijevo izdanje **4.4BSD** je završeno Juna 1993. Uključuje podršku
  - **za novi X.25 mrežni protokol**
  - za POSIX standard
- Takođe ima radikalno promenjenu organizaciju fajl sistema,
  - sa novim interfesjom prema virtuelnoj memoriji i
  - podršku za stack-bazirani fajl sistem,
  - koji omogućuje fajl sistemu da bude u slojevima za lako ubacivanje novih mogućnosti.
- **Dodata je i implementacija NFS**, kao i novi fajl sistem sa dnevnikom.
- Još nekoliko promena je dodato
  - kao poboljšana sigurnost
  - poboljšana struktura kernela
- Sa ovim izdanjem Berkli obustavlja rad na ovom projektu.
- 4BSD operativni sistem je bio izbor za VAX CPU od njegovog prvog izdanja (1979) do pojave Ultrix-a, DEC-ove BSD implementacije.
- 4BSD je i dalje najbolji izbor za mnoge istraživačke i mrežne oraganizacije.
- Mnogo organizacije su kupile 32V licencu i naručile 4BSD od Berklija.

Other UNIX

- Trenutni skup verzija UNIX operativnih sistema nije ograničen samo na one iz Bellovih laboratorija (čiji je vlasnik Lucent Technology) i iz Berklija.
- Sun Microsystems je pomogao u popularizaciji BSD verzije UNIX-a, tako što je isporučivao svoje radne stanice za njim.
- UNIX je rastao u populaciji, i bio je instaliran na mnogo različitih računara i računarskih sistema.
- Kreirana je široka lepeza UNIX i UNIX bazirnih operatinih sistema.
- DEC ima svoj UNIX (zvan Ultrix) na radnim stanicama, i njegovog nastavljača OSF/1, koji je takođe nastao od UNIX-a.
- Microsoft je preradio UNIX za Intel 8080 familiju i nazvao ga XENIX, a i Windows NT operativni sistem je nastao pod jakim uticajem UNIX-a.
- IBM koristi UNIX (AIX) na svojim PC-jevima i na serverima.
- Praktično, UNIX je dostupan na gotovo svim vrstama računara za opštu upotrebu, ima ga na personalnim računarima, radnim stanicama, miniračunarima, serverima i superračunarima, od Applovog Mekitoša do Cray-a.
- Zbog široke raspoložinosti, on se koristi na raznim mestima, počev od akademskih i vojnih organizacija, do fabrika. Većina ovih sistema je bazirano na Verziji 7, Sistemu III, 4.2BSD-u, ili Sistemu V.
- **MAC OS X**

UNIX standardizacija- potreba

- Široka popularnost UNIX-a je dovela do toga da UNIX bude najrašireniji operativni sistem, i da korisnici očekuju da UNIX okruženje bude nezavisno u odnosu na specifičnost hardvera.
- Ali veliki broj njegovih implementacija doveo je do toga da postoje razne varijacije u programiranju i korisničkom interfejsu, koji je distribuiran od izdavača.
- Za pravu nezavisnost, oni koji razvijaju programe treba da imaju konzistentan interfejs.
- Takav interfejs bi omogućio da **sve "UNIX" aplikacije rade na svim UNIX sistemima, što svakako nije trenutna situacija.**
- Ovo je postalo veoma važno kako je UNIX postao omiljena platforma za razvoj programa pocevši od baza preko grafičkih programa, pa sve do mrežnih aplikacija, i zato je tržište zahtevalo da se postavi standard za UNIX.

Standardizacija UNIX OS -projekti

- Postoje nekoliko projekata standardizacije u toku,
- počevši od /usr/group 1984 Standard koga finansira UniForum industry users' group.
- Od tada, mnoga tela za standardizaciju se bave tim problemom, uključujući IEEE i ISO (POSIX standard).
- Grupa pod imenom **X/Open Group** internacionalni konzorcijum je napravila XPG3, koji je tipično okruženja za aplikacije (Common Application Enviroment), koji podvodi pod širu kategoriju IEEE standarde interfejsa.
- Nažalost, XPG3 je baziran na skici ANSI C standarda, a ne na konačnoj specifikaciji i zato je morao da se preradi. XPG4 je završen 1993.
- U 1989., telo za ANSI standardizaciju, pravi ANSI C standard, kome se proizvođači lako prilagođavaju. Dok se rad na ovim projektima nastavlja, razna izdanja UNIX-a nastaju a postoji samo jedan programerski interfejs za UNIX i zato UNIX postaje sve popularniji.
- Praktično postoje dva različita seta moćnih UNIX proizvođača koji rade na ovom problemu. **AT&T-guided UNIX International (UI)** i **Open Software Foundation (OSF)**, koji su se oboje složili da prate POSIX standard.
- Nedavno, mnogi proizvođači iz ove dve grupe su se složili oko dalje standardizacije (COSE sporazum) na Motif okruženju prozora, i ONC+ (koji uključuje Sun RPC i NFS), kao i DCE mrežne kapacitete (koji uključuju AFS i RPC paket).

UNIX, System V, Release 4

- ATA&T 1989,
  - zamenio ATTIS grupu sa **USO** (UNIX Software Organization),
  - koja isporučuje prvi **spojeni-merged UNIX**
  - UNIX System V, Release 4.
- Ovaj sistem kombinuje funkcionalnost
  - UNIX Sistema V
  - 4.3BSD
  - Sun-ovog SunOS,
  - uključuje duga imena za datoteke, Berkeley fajl sistem,
  - menadžment sa virtualnom memorijom,
  - simboličke linkove,
  - višekorisniče grupe,
  - kontrolu poslova, i sigurne signale;
  - takođe poštuje POSIX standard, POSIX.1.
- Nakon toga USO pravi SVR4 i postaje nezavisan od ATA&T,
  - pod imenom Unix System Laboratories (USL);
  - 1993, USL kupuje Novell, Inc

UNIX for teaching

- UNIX sistem je porastao od ličnog projekta dvojce saradnika Bell laboratorija do operativnog sistema koji je definisan međunarodnim standardizacionim telima.
- Ipak, ovaj sistem je i dalje interesantan akademskim institucijama. Verujemo da je UNIX postao i da će ostati važan deo teorije i prakse o operativnim sistemima.
- UNIX je odličan OS za akademske studije. Na primer:
  - **Tunis operativni sistem**
  - **Xinu operativni sistem**
  - **Minix operativni sistem**
- su bazirani na konceptima UNIX-a, ali su razvijai eksplicitno za učionice.
- Preobilje je aktuelnih istraživačkih OS baziranih na UNIX-OSu, uključujući Mach, Chorus, Comandos i Roisin.
- Originalni tvorci, Riči i Tompson, su nagrađeni 1983. od Asociation for Computing Machinery, nagradom Turing, za njihov rad na UNIX-u.

FreeBSD – Intel BSD – verzija 1

- Verzija UNIX-a korišćena u ovom poglavlju je Intelova verzija Free BSD-a. Sistem je korišćen zato što implementira razne interesantne koncepte operativnih sistema, kao što je zahtev za straničenjem sa klasterovanjem, i umrežavanje.

- **FreeBSD projekat je počeo 1993.**
- kako bi napravio snimak 386BSD-a i
- rešio problem koje je bilo nemoguće rešiti koristeći postojeći patch mehanizam.

- 386BSD je nastao od 4.3BSD-Lite (Net/2) i originalno je izbačen u junu 1992 od strane Williama Jolitza.
- FreeBSD (David Greenman) 1.0 je izbačen decembra 1993.
- FreeBSD 1.1 je pušten maja 1994. i obe verzije su bile bazirane na 4.3BSD-Lite.
- Zbog nekih ugovora između UCB-a i Novella, bilo je potrebno je da se kod iz 4.3BSD više ne koristi, tako da je konačan 4.3BSD-Lite izbačen jula 1994. (FreeBSD 1.1.5.1).

FreeBSD – Intel BSD – verzije 2, 3, 4

- FreeBSD je ponovo napravljen,
  - bazirano na kodu 4.4BSD-Lite-a,
  - koja je nepotpuna, i izdat je novembra 1994,
  - pod oznakom FreeBSD 2.0.

- Kasnija izdanja su
  - 2.0.2 u junu 1995,
  - 2.1.5 i avgustu 1996,
  - 2.1.7 u februaru 1997,
  - 2.2.1 u aprilu 1997,
  - 2.2.8 u novembru 1998,
  - 3.0 u oktobru 1998, 3.1 u februaru 1999,
  - 3.2 u maju 1999,
  - 3.3 u septembru 1999,
  - 3.4 u decembru 1999,
  - 3.5 u junu 2000,
  - 4.0 u martu 2000,
  - 4.1 u julu 2000 i
  - 4.2 u novembru 2000.

FreeBSD - osobine

- Cilj celog FreeBSD projekta je da omogući softverski alat koji bi mogao da se koristi za svaku svrhu bez bilo kakvih obaveza.
- Ideja je da se kod maksimalno koristi i da pruži najviše koristi.
- Osnova je ista kao ona opisana u McKusick et al. [1984],
  - sa dodatkom povezane virtualne memorije i
  - fajlsistemskog baferovanog keša,
  - kernelskih upita, i
  - softverskih update-a za fajlsistem.
- Trenutno, FreeBSD radi prvenstveno na Intelovim platformama, iako su Alpha platfome podržane.
- U toku je rad na tome da se podrže i druge procesorske platforme.

popularnost UNIX OS

- **sistem je napisan** na **high level PL**, sto mu omogućava
  - **da bude lakši za čitanje**
  - **razumevanje**
  - **modifikaciju**
  - **prenos na druge računarske konfiguracije.**
- **Ritchie je procenio** da je **na prvom UNIX-u** C napravio
  - **povećanje koda i usporenje**
  - **od 20 do 40%**
  - **u odnosu na assembler realizaciju,**
  - **ali su prednosti višeg PL kasnije došle do izražaja**
- **UNIX poseduje jednostavni korisnički interfejs**
  - koji userima omogućava sve što žele
- **UNIX obezbeđuje primitive**
  - koje omogućavaju kompleksnijim programima
  - da se realiziju iz jednostavnijih programa

popularnost UNIX OS

- **UNIX koristi hijerarhijski FS** koji omogućava
  - lako održavanje
  - efikasno korišćenje i realizaciju
- **UNIX koristi konzistentan format za datoteke** i **nizove bajtova**,
  - što omogućava lakše pisanje programa
- **UNIX** obezbeđuje **jednostavan konzistetan intefejs** za **periferijske uređaje**
- **UNIX** je **multi-user, multi-task OS**:
  - svaki user može izvršavati više programa istovremeno
- **UNIX apstrakuje mašinsku arhitekturu** od **korisnika**
  - **što omogućava lakšu realizaciju programa**
  - na **različitim hardverskim arhitekturama**
- **Mada su OS i većina komandi realizovana na C PL-u,**
- UNIX podržava **masu drugih jezika** kao što su
  - **Fortran, basic, Pascal, Ada, Cobol, Lisp, Prolog...**

Early Advantages of UNIX

- **Napisan u visokom programskom jeziku (high-level language).**
- **Distribucija izvornog koda (Distributed in source form).**
- **Obezbeđuje**
  - **moćan skup OS baziranih primitiva**
  - **na jeftinim hardverskim platformama**
- **Poseduje**
  - **malu veličinu**
  - **modularan i jasan dizajn**
- **Unix je bio napravljen**
  - **od programera**
  - **za programere**

UNIX Design Principles

- **Projektovan da bude time-sharing** system.
- **Poseduje**
  - **standardni user interface** (**shell**)
  - **koji se lako može zameniti**
- **FS (File system)**
  - sa **multilevel tree-structured direktorijumima**.
- **Kernel**
  - **podržava datoteke**
  - **kao nestruktuiranu sekvecu bajtova**
- **UNIX je multitask OS**
  - procesi mogu jednostavno da kreiraju nove procese
  - **Mehanizam se naziva fork**

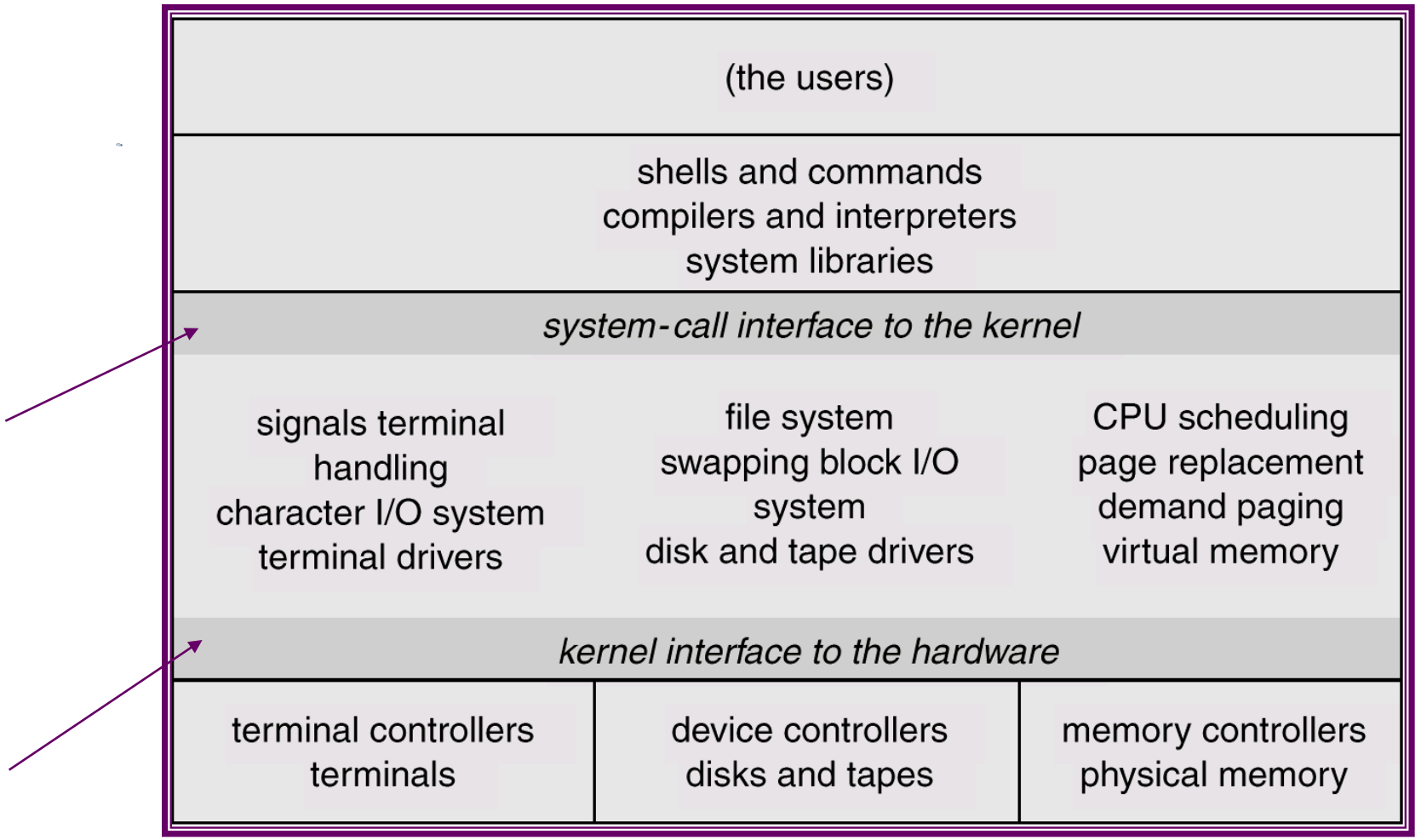
Programmer Interface

- **Kao i većina OS, UNIX** se sastoji od 2 odvojena dela :
- **1. Kernel**:
- **kernel je sve**
  - **ispod SC (system-call) interface**
  - **i**
  - **iznad fizičkog hardvera**
- **Kernel obezbeđuje:**
  - **file system upravljanje (management)**
  - **upravljanje raspoređivanje procesa process management and CPU scheduling**
  - **upravljanje memorijom memory management**
  - **druge OS funkcije preko SC (system calls)**
- **2. Sistemski programi**:
  - **koriste kernel-supported SC (system calls)**
  - obezeđuju korisne funkcije, kao što su
    - **prevođenje (compilation)**
    - **file manipulacija**

Struktura sistema

- 1. Hardver
- 2. Kernel i SC interfejs
- 3. System programs
- cc:
- sastoji se od:
  - C preprocessor,
  - two-pass compiler,
  - assembler and
  - loader (link-editor)
- 4. Other application programs

UNIX Layer Structure



System Calls

- **1. SC interface: definiše programerski interface** za UNIX OS

- **2. user interface :**
  - **set sistemskih programa**
  - **koji su raspoloživi korisnicima**
  - **definiše** korisnički-**user interface**.

- **programerski** i **user interface**
  - **definišu kontext koji**
  - **kernel mora podržavati**

- **Postoje 3 kategorije sistem calls na UNIX OS**:
  - **1. Rad sa datotekama (File manipulation)**
  - **2. Kotrola procesa (Process control)**
  - **3. Informacije (Information manipulation)**

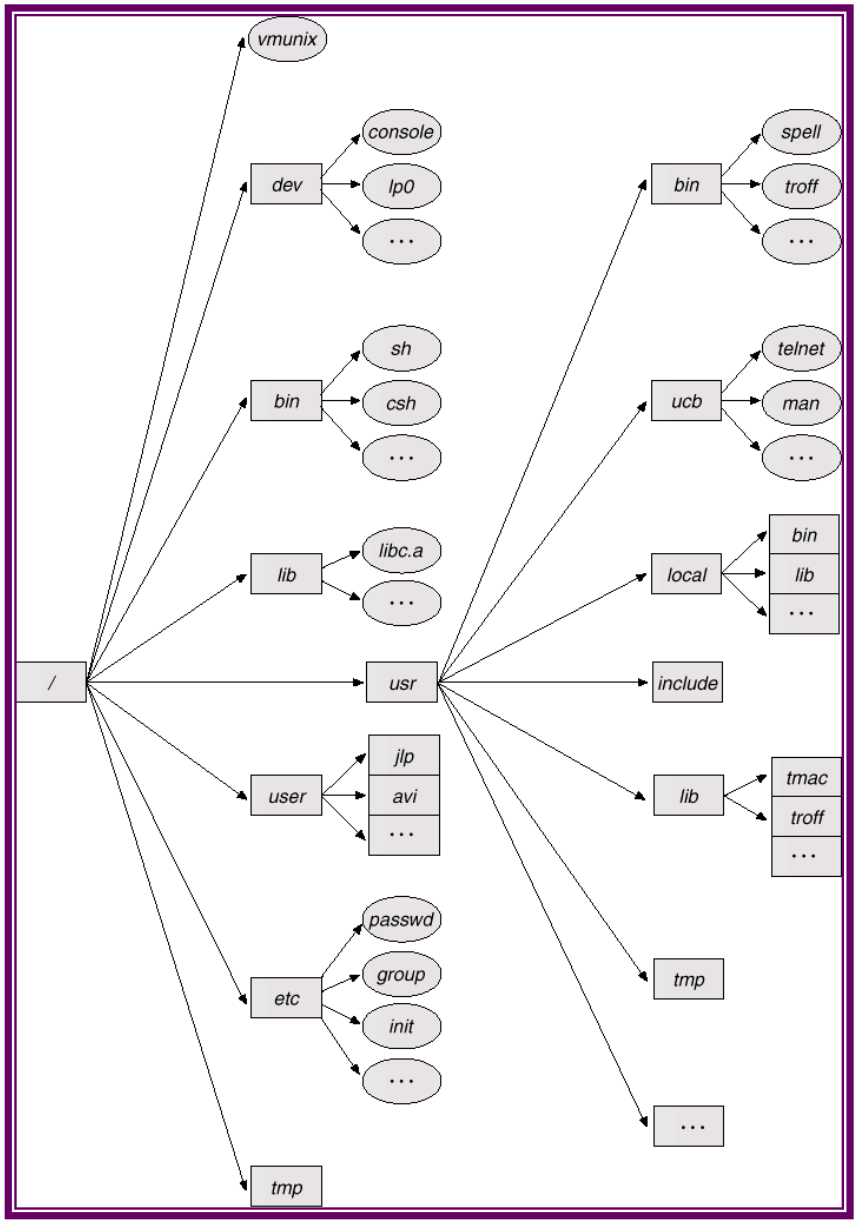
Introduction to UNIX FS

- **UNIX FS** se **karakteriše** sa **sledećim osobinama**:
  - **hijerarhijska struktura**
  - **konzistetno tretiranje podataka**
  - **mogućnost kreiranja i brisanja datoteka**
  - **mogućnost dinaničkog rasta datoteka**
  - **zaštita datoteka preko prava pristupa**
  - **tretiranje periferijskih uređaja kao datoteka**
- FS je organizovan **kao stablo**
  - sa jednim **korenskim direktorijumom**
  - koji se zove **root** i obeležava kao ***/***
  - Sve ostalo u stablu su direktorijumi, regularne i specijalne datoteke
  - **ime datoteke** uključuje i **putanju koja joj određuju mesto u stablu**
- **Direktorijumi** su **specijalni nizovi bajtova** podataka koji sadrže opis datoteka
- **Uređaji** se **tretiraju kao datoteke**,
  - istim komandama se **radi sa uređajem** kao **sa datotekom** (cp, mv, ls, cat)
  - **na** isti način imaju zaštitu i kontrolu pristupa (rwx) kao datoteke

File Manipulation

- datoteka je sekvenca bajtova
- **UNIX kernel ne nameće strukturu za datoteke.**
- **Datoteke se organizuju u direktorijume koji imaju strukturu stabla (tree-structured directories)**
- **Direktorijumi su datoteke**
  - **koje sadrže informacije**
  - **kako da se pronađu druge daoteke**
- **Putanja (Path name): identifikuje datoteku**
  - **specificiranjem putanje kroz direktorijumsku strukturu do datoteke**
  - **a) apsolutna putanja počinje od korenskog direktorijuma**
  - **b) relativna putanja startuje u** tekućem **direktorijumu**
- **SC pozivi za osnovnu manipulaciju sa datotekama** :
  - **create**
  - **open**
  - **read**
  - **write**
  - **close**
  - **unlink**
  - **truncate.**

Typical UNIX Directory Structure



Process Control

- **Proces** je **jedna istanca programa u izvršavanju**.
- Pod **UNIX OS**, **proces je celina kreirana fork sistemskim pozivom**.
- **4 SC** su karateristična za kreiranje procesa pod UNIX-om
  - **1. fork: kreira novi proces**
    - **kopira adresni prostor od roditelja za proces dete**
    - i **svakome dodeli pid** {0 za dete, !=0 za roditelja).
  - **2. exec: izvršava novi proces**
    - **SC pomoću kojeg dete proces puni (overlay) odgovarajući program**
    - **u svoj adresni prostor.**
    - Kada se dogodi execl, roditelj i dete **nisu više isto.**
    - Dete se više ne vraća na početni kod jer ga je prepisalo sa **execl SC**
  - **3. exit:** proces završava aktivnosti
    - dete mora imati sistemski poziv exit
    - koji znači da je obavilo svoje
    - i **da se ukida, osobađajući resurse**
  - **4. wait:** roditelj čeka na dete
    - **SC koji omogućava procesu roditelju da se blokira** i
    - **čeka da dete obavi svoje**

Introduction to Processes

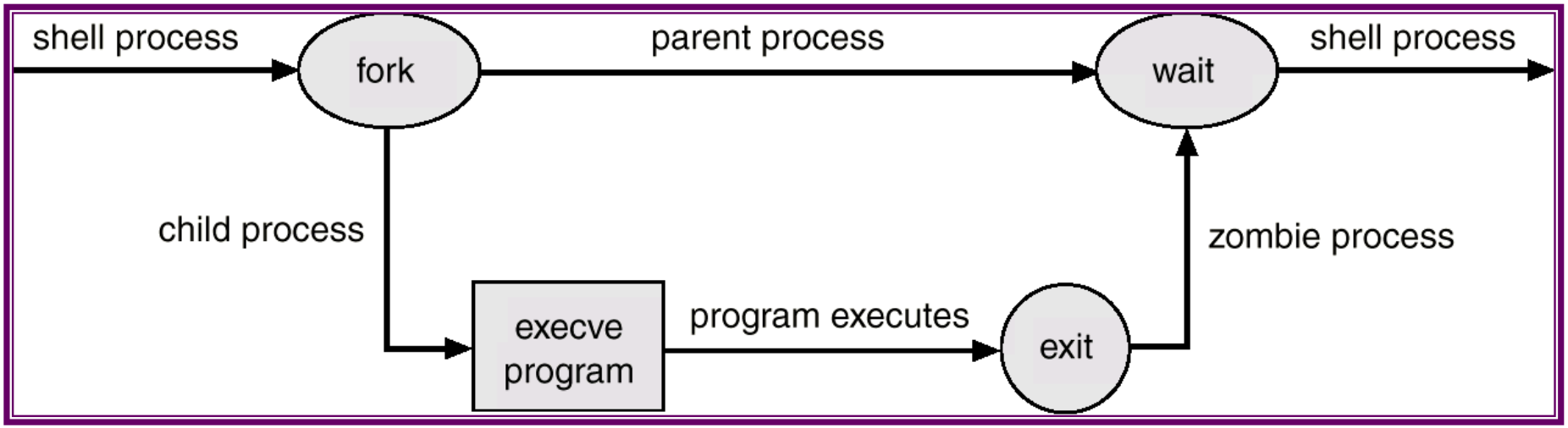
```
main (argc, argv)
int argc;
char *argv[];

{
/* assume 2 args; source file and target file*/
if ( fork() == 0)    execl("cp", "cp", argv[1],argv[2], 0)

        wait ((int *) 0 );

        printf("copy done\n")
}
```

Illustration of Process Control Calls



3 type of commands

- **UNIX shell** dozvoljava 3 tipa komandi:
  - **execulable binarne datoteke** koje sadrže object code nastao kompilacijom source koda
  - **shell script datoteke** koje nastaju kao programske konstrukcije sastavljene od shell komandnih linija
  - **interne komande shell-a** koje omogućavaju shell programske konstrukcija tipa if, while ... for, ali i neke komande tipa cd...
- **Shell** radi na **principu fork-a** za **foreground** procese
  - čekajući ih da se završe
  - to je sinhroni rad
- **Shell** može raditi i **asinhrono**
  - da napravi procese
  - ali da ih postavi u **background**
  - ne čekajući da se završe.
  - To je **asinhroni** rad shell-a a postiže se sledećom sintaksom:
- **command &**
- **Svaki proces ima svoju radnu okolinu**, a jedan od jako bitnih parametara je **radni direktorijum**
- **Shell nije deo kernela** može se **lako modifikovati** i **promeniti.**

Shells and Commands (Cont.)

- **Tipicani search path:**
  - **.**
  - **/home/prof/avi/bin**
  - **/usr/local/bin**
  - **/usr/ucb**
  - **/bin**
  - **/usr/bin**

- **shell**
  - obicno **suspenduje** svoje aktivnosti (its **own execution)**
  - **dok se komanda ne kompletira**

- **Foreground process**
- **Background process [&]**

Realizacija blok primitiva

- UNIX obezbeđuje **blok primitive** koje **omogućavaju korisnicima**
  - da pišu male, **modularne programe**
  - koji se koriste za za realizaciju kompleksnijih programa.
- **Jedna tipična blok primitiva** je **redirekcija ulaza i izlaza**. Svaki proces ima konvencionalno **3 datoteke** ili **3 file descriptora**:
  - **standard input**
  - **standard output**
  - **error output**
- **Tipični primeri su:**
  - **$ls >output**
  - **$mail mjb <letter**
  - **$nroff -mm <doc1 >doc1.out 2>errors**

- Druga **blok primitiva** je **pipe**, **mehanizam**
  - **koji obezbeđuje da se niz podataka prosledi**
  - **između procesa čitaoca i procesa pisca.**
  - **Proces pisac redirektuje svoj standardni izlaz na pipe,**
  - dok **proces čitaoc redirektuje svoj standardni ulaz na pipe.**
- Tipičan primer je:
  - **grep main a.c b.c c.c | wc -l**

Standard I/O

- **Vecina procesa ocekuje**
- **da 3 fd (file descriptors)**
- **budu otvoreni, kada se startuju:**
  - **standard input** – **fd=0**
    - **program can read what the user types**
  - **standard output – fd=1**
    - **program can send output to user's screen**
  - **standard error – fd=2**
    - **error output**

- Vecina procesa
  - **moze takodje da prihvati datoteku (rather than** a **terminal)**
  - **za standard input** i **standard output.**

- **Shells imaju** jednostavnu sintaksu za
  - **I/O redirection**.

Standard I/O Redirection

Command	Meaning of command
% **ls > filea**	direct **output** of **ls** to file **filea**
% **pr < filea > fileb**	input from filea and output to fileb
% **lpr < fileb**	input from fileb
%% make program **> &** errs	**save both standard output** and **standard error in** a **file**

Pipelines, Filters, and Shell Scripts

- **Mogu se udruziti individualne comande**
  - preko **vertical bar**
  - **koji kaze shell-u**
  - **da prosledi output prethodne komande**
  - **kao input za sledecu komandu**

$ ls | pr | lpr

- **Filter:**
  - komadne **kao** pr
  - **prosledjuju svoj standard input na svoj standard output**,
  - **obavljajuci neko procesiranje na njemu.**
- **Pisanje** novog **shell-a** sa **razlicitom sintaksom i semantikom**
  - **mogu promeniti korisniki pristup (user view)**,
  - **ali ne menjaju kernel i programmer interface**.
- **Shell programming**
  - Shell variables
  - Program structure: if-then-else, for, while, until, case….
- **X Window System is** a **widely accepted iconic interface for UNIX**

Servisi operativnog sistema

- **Ovde je reč o servisima kernela:**
- **1. upravljanje procesima** koje omogućava njihovu
  - kreaciju
  - terminaciju
  - suspenziju
  - komunikaciju
- **2. raspoređivanje procesa** (CPU scheduling)
- **3. alokaciju memorije procesima i swapping**
- **4. alokaciju sekundarne memorije (FS services)**
- **5. dozvoljavanje procesima da kontrolišu periferale (I/O system)**

Pretpostavke oko hardvera

- **Izvršavanje UNIX procesa se deli na 2 nivoa:**
  - kernelski (**kernel mode**)
  - korisnički (**user mode**)
- **Kada proces izvršava SC**
  - izvršni mod procesa se menja sa user mode u kernel mode
  - a OS pokušava da zadovolji korisnički zahtev
  - i vraća status da li je uspeo ili se dogodila greška
- No bez obzira da li će user proces da obavi SC ili ne,
  - OS obavlja masu aktivnosti kao što su
  - interrupt handling, CPU scheduling, ...
- **Neke procesorske arhitekture dozvoljavaju više nivoa, ali UNIX se zadovoljava sa 2:**
  - **korisnicki nivo: procesi u korisničkom modu**
    - mogu pristupati samo svojim instrukcijama i podacima
    - ali ne i kernelskim instrukcijama i podacima.
  - **kernelski nivo: procesi u kernelskom modu**
- **mnoge instrukcije su privilegovane** i
  - mogu se izvršavati samo u kernelskom modu

Prekidi i uzuzeci

- Hardverski prekidi
- Izuzeci (exception)

Hadverski prekid

- **Hadverski prekid** je **mehanizam** kojim se
  - označava neki događaj (I/O completion)
  - omogućava I/O uređajima
    - da prekinu CPU asinhrono.

- **Po prijemu prekida**
  - izvrši se poslednja instrukcija koja se obavljala u trenutku prekida
  - a kernel će sačuvati stanje prekinutog procesa
  - odrediti uzrok prekida
  - pozvati prekidnu rutinu

- **Nakon toga kernel obavlja po pravilu novi CPU scheduling**

- **Za vreme obrade prekida, mogući su novi prekidi**,
  - ali se to obavlja prioritetno
  - niži prekidi ne prekidaju
  - više prekide.

exceptions (uzuzeci)

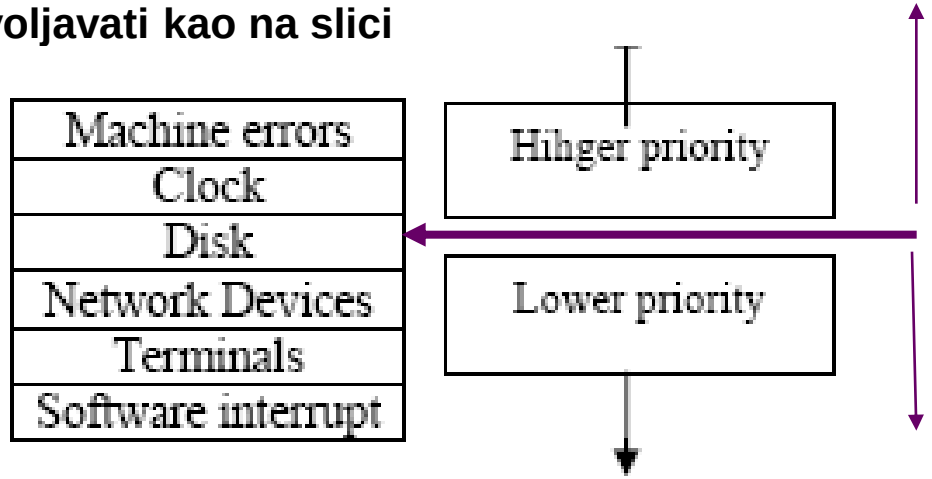
- **Izuzeci (exception)** su **neočekivani događaji**
- koje je **izazvao sam proces**
  - **kao što** je **page fault**
  - **illegal memory**
  - **izvršavanje privilegovane instrukcije**
  - **deljenje sa 0**

- **po tome se razlikuju** od **prekida**
  - **koje izazivaju neki eksterni događaji, van procesa.**

- **Jedino se obrađuju slično**, jer se
  - prekida instrukcija
  - kernel određuje vrstu trapa i trap rutinu
  - ***ako ima smisla***
  - obavlja se **restart** instrukcije koja je napravila izuzetak.

CPU execution level

- **Kernel mora da se zaštiti od prekida**
- za vreme nekih **kritičnih operacija**,
  - na primer, da ne dozvoljava disk prekid
  - **dok ažurira povezane liste**.
- **Računari koriste skup privilegovanih instrukcija** koje **postavljaju**
  - **CPU nivo izvršenja** ili kako se nazivaju
  - **prekidni nivoi** **(CPU execution level)**
- **Kada se postavi neki nivo izvršenja**,
  - **svi prekidi koji su ispod tog nivoa**
  - **se neće dozvoljavati kao na slici**



Upravljanje memorijom

- **Kernel je rezidentno u memoriji**
- **ili**
- **bar deo njega.**

- **Većina OS** pa i **UNIX**
- **koristi princip virtuelne memorije,**
  - **čija je fundamentalna osobina**
  - **mapiranje logičkih u fizičke adrese**

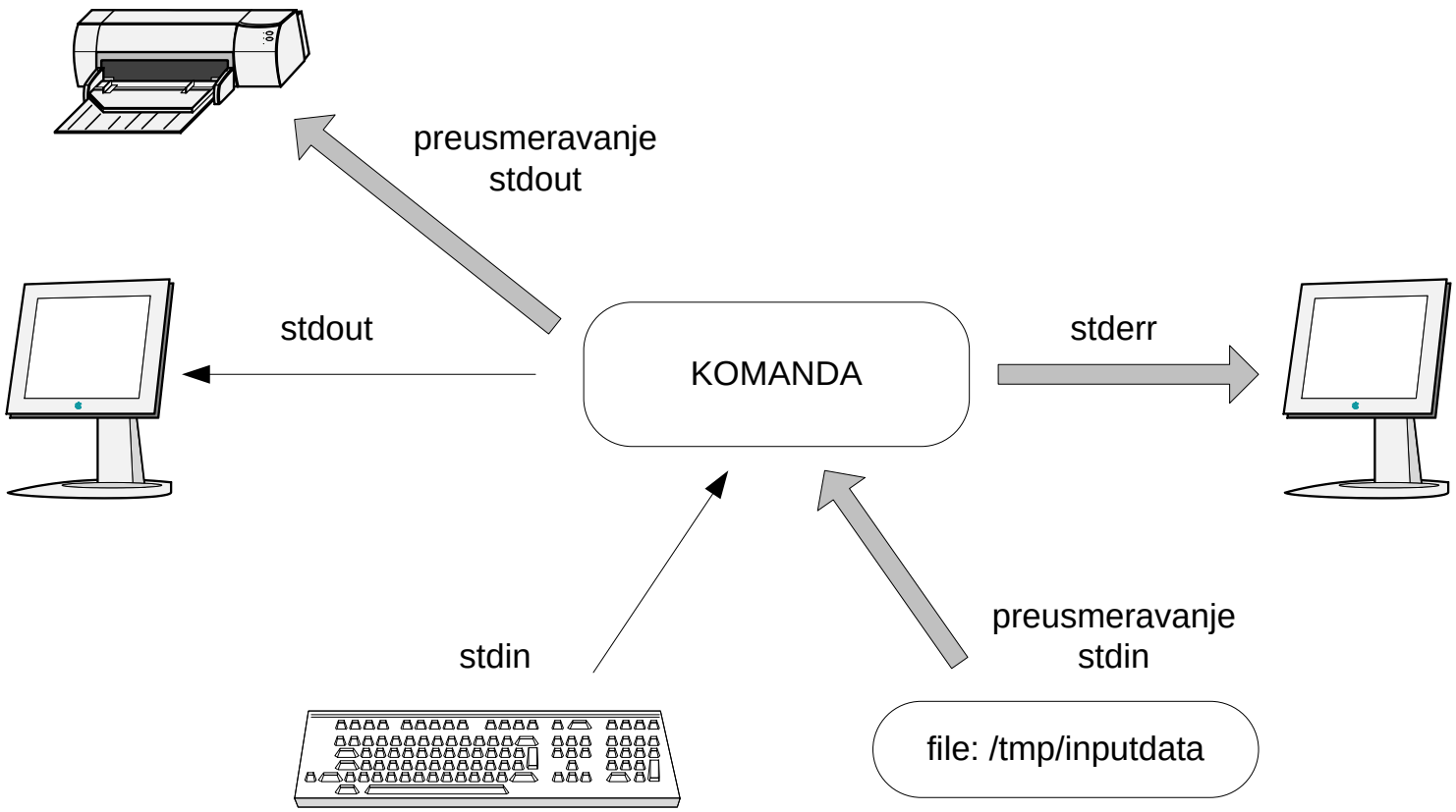
PRIMER BR. 1
< Redirekcija izlaza >

- Sadžaj direktorijuma /etc
- izlistati
- i
- upisati u datoteku etc.txt,
- na direktorijumu /tmp

- Proveriti sadržaj datoteke

PRIMER BR. 1
< Redirekcija izlaza >

- Pođimo od opisa redirekcija ulaza i izlaza



PRIMER BR. 1
< Redirekcija izlaza >

- Izlaz se češće preusmerava od ulaza.
- Za redirekciju se koristi **znak > (veće od)**
- Ukoliko se redirekcija vrši u postojeću datoteku, datoteka se briše, a zatim
- se kreira nova u koju se smešta rezultat izvršenja komande.
- Ukoliko korisnik želi da se rezultat izvršenja doda na postojeću datoteku bez brisanja njenog sadržaja, za redirekciju izlaza se koristi znak >>.
- Sledeći primer ilustruje redirekciju izlaza na štampač, u novu datoteku i postojeću datoteku i kreiranje prazne datoteke.
- **$ sort kyuss.txt > /dev/lp0**
- **$ ls -l /home/jsmith > myfile**
- **$ ls -l /tmp/jsmith >> myfile**
- **$ >emptyfile**

PRIMER BR. 1
< Redirekcija izlaza >

- Standardni izlaz za greške se preusmerava ukoliko korisnik želi da sačuva rezultat izvršenja komande u nekoj datoteci radi kasnije analize grešaka (na primer, debagovanje programa u fazi razvoja).

- Izlaz za greške se preusmerava u tekstualne datoteke pomoću znaka 2>.

- $ **./testprogram 2> debugging.txt**

PRIMER BR. 1
< Redirekcija izlaza >

- Vratimo se na naš problem.
- Sadžaj direktorijuma /etc izlistati i upisati u datoteku etc.txt, na direktorijumu /tmp
- Da bi smo listali sadržaj direktorijuma, moramo znati Linux komandu ls
- Korisnik može na ekranu pomoću komande ls (list) prikazati sadržaj bilo kog direktorijuma aktivnog UNIX stabla.

PRIMER BR. 1
< Redirekcija izlaza >

- Sintaksa komande ls je:
- **$ ls [options] [dir][filespec]**
- Komanda ls će na ekranu prikazati spisak objekata direktorijuma dir definisanih argumentom filespec. Argument filespec se formira pomoću džoker karaktera i nije obavezan. Ukoliko se ne navede, podrazumevaju se svi objekti u direktorijumu. Argument dir takođe nije obavezan, i ako se ne navede, prikazuje se sadržaj tekućeg direktorijuma. Ukoliko se ne navedu dodatne opcije, ls funkcioniše prikazuje samo imena objekata sortiranih u abecednom redu. Od značajnijih opcija komande ls navode se sledeće:
- -a prikazuju se i imena skrivenih objekata
- -B imena rezervnih kopija datoteka se ne prikazuju
- -d prikazuje se kontekst direktorijuma umesto sadržaja-
- -i prikazuje se i-node broj datoteke
- -R rekurzivno se prikazuje sadržaj svih poddirektorijuma
- -L dereferenciranje (umesto simboličkih linkova se prikazuju imena datoteka na koje linkovi upućuju)
- -h veličine datoteka se prikazuju u čitljivom formatu (1K, 234M, 2G)
- -k veličine datoteka se prikazuju u kilobajtima
- -l osim imena objekata, prikazuju se i informacije upisane u i-node (prava pristupa, vlasnik, grupa, datum, vreme)
- -1 prikazuje samo imena objekata (jedno ime u jednoj liniji)
- -r imena objekata se prikazuju sortirana u opadajućem redosledu

PRIMER BR. 1
< Redirekcija izlaza >

- Da bi smo videli sadžaj direktorijuma /etc uradićemo sledeće komande
- **$cd /etc**
- **$ls**
- ili
- **$ls /etc**

- Ukoliko želimo da sadržaj direktorijuma, umesto na ekran prebacimo u datoteku, primenićemo redirekciju izlaza sa znakom >, tj uradićemo sledeću komandu
- **$ls /etc >/tmp/etc.txt**

- Proveravamo sadržaj datoteke sa komandom less
- **$less /tmp/etc.txt**

PRIMER BR. 2
< Redirekcija ulaza >

- Imate sistem sa sa 2 diska,
  - /dev/hda kao primarni master sa Linux OS
  - izmenljivi disk ZIP od 1GB na poziciji od sekundarnog mastera /dev/hdc
- Potrebno je primpremiti novi ZIP medijum, sa fdisk programom ali bez intervencije korisnika.
- Interaktivni program fdisk izvršiti automatski, tj neinteraktivno bez upotrebe tastature

PRIMER BR. 2
< Redirekcija ulaza >

- Ulaz komande se preusmerava pomoću znaka < (manje od) na sledeći način:
- $ **command < inputdevice**

- Inputdevice može biti datoteka ili ulazni uređaj (preusmeriti ulaz na izlazni uređaj kao što je štampač nema nikakvog smisla).

- Na primer, moguće je standardni ulaz sa tastature zameniti nekom tekstualnom datotekom:
- $ **wc -l < /tmp/jsnmith.dat**

PRIMER BR. 2
< Redirekcija ulaza >

- Vratimo se na problem
- Imate sistem sa sa 2 diska,
- /dev/hda kao primarni master sa Linux OS
- izmenljivi disk ZIP od 1GB kao sekundarni mastera/dev/hdc
- Potrebno je primpremiti novi ZIP medijum, sa fdisk programom ali bez intervencije korisnika. Interaktivni program fdisk izvršiti automatski, tj neinteraktivno bez upotrebe tastature

- Da bi smo rešili problem, upotrebićemo program fdisk sa redirekcijom ulaza. Opišimo prvo program fdisk. Program za particionisanje diskova koji se isporučuje uz Linux je fdisk (istiomeni program se isporučivao uz DOS i Windows 9x/ME). Detalji o njegovom korišćenju mogu se naći u on-line dokumentaciji (man pages).

PRIMER BR. 2
< Redirekcija ulaza >

- Program fdisk se pokreće na sledeći način:
- # **fdisk device**
- gde je device nod blok uređaja (diska) koga treba particionisati.
- # **fdisk /dev/sda**
- Command (m for help): m
- Command action
- a toggle a bootable flag
- b edit bsd disklabel
- c toggle the dos compatibility flag
- d delete a partition
- l list known partition types
- m print this menu
- n add a new partition
- o create a new empty DOS partition table
- p print the partition table
- q quit without saving changes
- s create a new empty Sun disklabel
- t change a partition's system id
- u change display/entry units
- v verify the partition table
- w write table to disk and exit
- x extra functionality (experts only)

PRIMER BR. 2
< Redirekcija ulaza >

- fdisk omogućava korisniku da uradi sledeće stvari:
  - prikazivanje particione tabele (p - print the partition table)
  - pregled podržanih tipova particija (l - list known partition types)
  - kreiranje primarnih, extended i logičkih particija (n - add a new partition)
  - brisanje particija (d - delete a partition)
  - promena tipa particija (t - change a partition's system id)
  - postavljanje flega aktivne particije (a - toggle a bootable flag)
- Pri tome, promene se ne upisuju na disk dok korisnik ne napusti program pomoću opcije w - write table to disk and exit.
- Napuštanje programa pomoću opcije q - quit without saving changes ne povlači upisivanje promena na disk.

PRIMER BR. 2
< Redirekcija ulaza >

- U našem slučaju mi bismo pozvali fdisk i u njemu otkucali njegove interne komande
- **#fdisk /dev/hdc**
  - n kreiranje nove particije
  - 0 početak particije
  - 1.1G veličina particije
  - w upis u MPT
  - q izlazak iz programa
- To bi bio interaktivni način rada. Da bi realizovali ne-interaktivni način rada, moramo formirati tekstualnu datoteku, na primer fdisk.txt, gde bi smo za svaku komandu kreirali posebnu liniju
- Izgled datoteke fdisk.txt bio bi
  - **dev/hdc**
  - n
  - 0
  - 1.1G
  - w
  - q
- Nakon toga rešenje našeg problema je
- **#fdisk </tmp/fdisk.txt**

PRIMER BR. 3
< Povezivanje komandi u pipeline>

- Analizirajte sistemski direktorijum **/bin.**

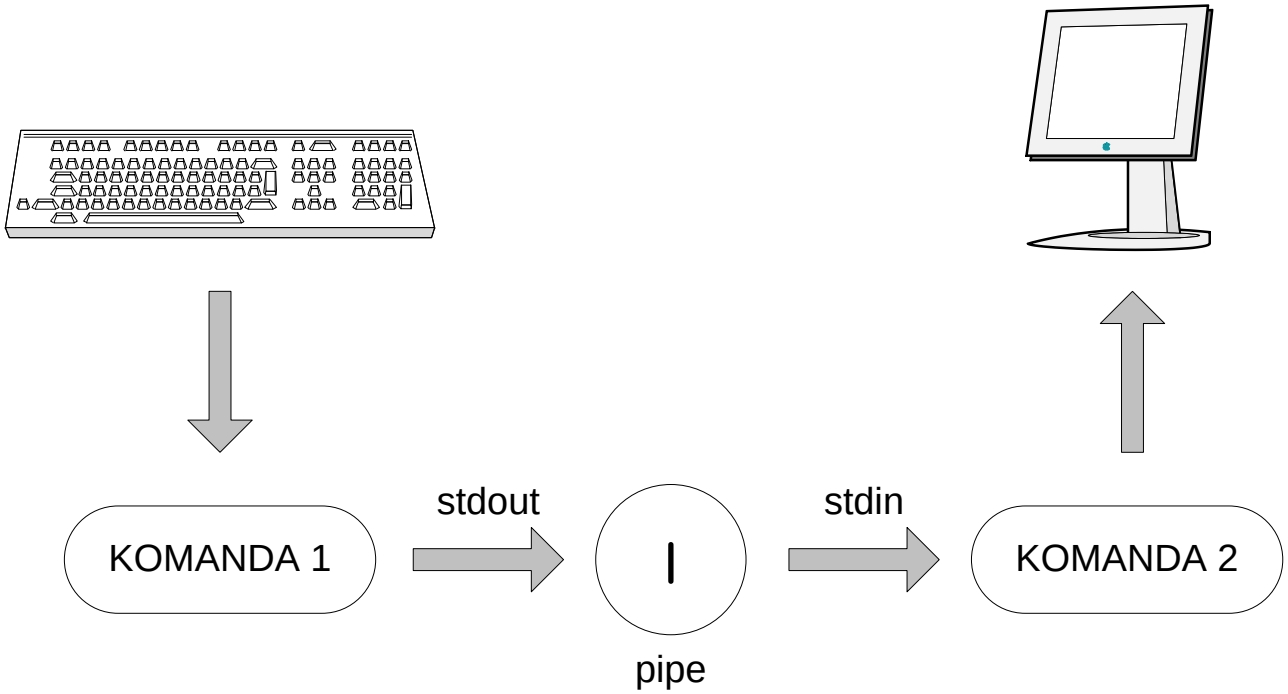
- Obavite prebrojavanje svih objekata u njemu,
- uz pomoć samo jedne komande

PRIMER BR. 3
< Povezivanje komandi u pipeline>

- Pođimo prvo od povezivanja komandi u pipeline
- Prebrojavanje datoteka u direktorijumu /etc može realizovati pomoću pipeline sprege komandi ls i wc:
- **$ ls -l /etc | wc -l**
- 145
- Opšta sintaksa pipeline sprege je sledeća:
- **$ command1 | command2 | .... | commandN**
- To znači da se u pipeline može povezati veći broj komandi (maksimalan broj zavisi od konkretnog UNIX sistema, a obično se kreće od 20 do 30).

PRIMER BR. 3
< Povezivanje komandi u pipeline>

- Princip funkcionisanja pipeline sprege prikazan je na slici



PRIMER BR. 3
< Povezivanje komandi u pipeline>

- Vratimo se na problem
- Analizirajte sistemski direktorijum /bin. Obavite prebrojavanje svih objekata u njemu, uz pomoć samo jedne komande.

- a ovaj problem potrebne su nam 2 Linux komande,
- ls
- wc.

- Komandu ls smo već opisali, a za komandu, a sada dajemo kratak opis komande wc.

PRIMER BR. 3
< Povezivanje komandi u pipeline>

- Komanda wc (word count) se koristi za brojanje karaktera, reči i linija u datoteci. Sintaksa komande je:
- $ **wc [-cwl] filename**

- Bez opcija komanda prikazuje sve tri vrednosti, dok se opcijama -c, -w i -l specifira prebrojavanje karaktera (bajtova), reči ili linija.

- Datoteka /etc/protocols je iskorišćena radi ilustracije upotrebe komande wc:
- $ **wc -c /etc/protocols**
- 1748 /etc/protocols
- $ **wc -w /etc/protocols**
- 297 /etc/protocols
- $ **wc -l /etc/protocols**
- 44 /etc/protocols
- $ **wc /etc/protocols**
- 44 297 1748 /etc/protocols

PRIMER BR. 3
< Povezivanje komandi u pipeline>

- U našem slučaju, problem ćemo da rešimo primenom povezivanja komandi u pipeline:

- **$ls –R /bin | wc -l**

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- Otkucajte tekst programa koji štampa rečenicu Hello, world ma ekranu.
- Program prevedite i linkujte
- Program izvršite

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- **Rešenje**
- Pođimo najpre od editora teksta. Editor je program koji koristite za uređivanje teksta izvornog koda.
- Mnoštvo različitih editora je raspoloživo na Linux sistemu, ali najpopularniji i najfunkcionalniji je verovatno GNU Emacs.
- Imate takođe program vi, joe....

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- Kreiranje datoteke

- #include <stdio.h>
- int main (int argc, char **argv)
- {
-
- **printf ("Hello, world\n");**
- return 0;
- }

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- **Prevođenje pojedinačne datoteke izvornog koda**

- Naziv C prevodioca je gcc. Da bismo preveli C datoteku izvornog koda, koristimo -c opciju. Tako se, na primer, zadavanjem sledeće naredbe u komandnoj liniji prevodi main.c datoteka izvornog koda:
- **% gcc -c main.c**

- Rezultujuća objektna datoteka se zove main.o.

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- **Povezivanje objektnih datoteka**
- Sada, kada ste preveli main.c, treba da je linkujete. Uvek koristite g++ prevodilac kada povezujete program koji sadrži C++ kod, čak i onda kada on sadrži i C kod. Ako vaš program sadrži isključivo C kod, koristite gcc prevodilac.
- Kako ovaj program sadrži i C i C++ kod, koristićete g++ prevodilac i to na sledeći način:
- **$ gcc -o main main.o**
- Opcijom -o se definiše naziv datoteke koja će biti rezultat povezivanja.

PRIMER BR. 4 <Primer prevođenja i linkovanja C programa >

- **Izvršavanje programa**
- Sada možete pokrenuti main na sledeći način:

- **$ ./main**

- **Hello, world**