Robert Meolic
meolic@uni-mb.si
zadnja sprememba: 12. 02. 2009

Zapiski predavanj 1

interno gradivo za predmet VSO, 2008/09

1. LITERATURA

Pri sestavljanju zapiskov predavanj sem uporabljal internet in naslednje vire:

- Andrej Štrancar: Prosojnice za predmet VSO, 2006/07
- Abraham Silberschatz, Peter B. Galvin, Greg Gagne: Operating System Concepts, Sixth Edition. John Wiley & Sons, 2002
- William R. Stanek: Microsoft Windows Command-Line Administrator's Pocket Consultant, Microsoft Press, 2004
- John Paul Mueller: Windows Administration at the Command Line for WindowsVista, Windows2003, WindowsXP, and Windows2000, Wiley Publishing Inc., 2007
- Eric Foster-Johnson, John C. Welch, Micah Anderson: Beginning Shell Scripting, Wiley Publishing Inc., 2005
- Ken O. Burtch: Linux Shell Scripting with Bash, Sams Publishing, 2004

2. Razvoj in vrste sistemske programske opreme

Na začetku računalniki niso imeli operacijskega sistema. Upravljali in poganjali so jih z enega, centralnega mesta – konzole. Programer je bil hkrati tudi operater. Za takratne sisteme so značilni čitalniki kartic, vrstični tiskalniki in pomnilne enote z magnetnim trakom.

Računalniški sistemi so postali bolj prijazni, ko so se je začela razvijati **sistemska programska oprema** namenjena razvoju programov:

- zbirnik (assembler) in nalagalnik (loader), ki sta omogočil pisanje programov v obliki izvorne kode,
- tolmač (interpreter), prevajalnik (compiler) in povezovalnik (linker), ki so zelo olajšali programiranje.

in naposled tudi **sistemska programska oprema** namenjena upravljanju računalniških sistemov:

- datotečni sistemi omogočajo organizacijo podatkov na pomnilniškimi medijih,
- gonilniki (drivers) olajšujejo delo z napravami,
- ukazni tolmač (command interpreter) omogoča upravljanje računalniškega sistema s pomočjo ukazov in
- grafični uporabniški vmesnik, ki s svojimi grafičnimi elementi olajša upravljanje z računalniškim sistemom.

Veliko sistemske programske opreme je danes sestavni del operacijskih sistemov.

Posebna vrsta sistemske programske opreme je tista, ki skrbi za zagon računalnika. Tradicionalno za to skrbi BIOS, dandanes pa je vedno pogostejši EFI.

3. Ukazna vrstica, lupina in skriptni jeziki

Lupina je program, ki omogoča uporabniku **ukazno upravljanje** z operacijskim sistemom in sistemsko programsko opremo. V osnovi razpoznava uporabnikove ukaze in jih izvršuje. Pomemben del lupine je njen skriptni jezik, ki mora omogočati celovito programiranje vključno s pogojnim izvajanjem in zankami. Če takega skriptnega jezika ni, potem je bolje, da namesto izraza lupina uporabimo starejši izraz ukazna vrstica. Uporaba skript zelo poveča zmogljivosti lupine in omogoča avtomatizacijo mnogih opravil, ki bi sicer zahtevala izvršitev množice ukazov.

Pomena lupine se mnogi današnji uporabniki računalnikov ne zavedajo, k čemer je veliko pripomogel Microsoft. Pri Microsoftu namreč niso svojega operacijskega sistema kar tako imenovali Windows. Njihov koncept ob uvedbi je bil namreč v tem, da uporabniku ponudijo grafične pripomočke za upravljanje z operacijskim sistemom, ukazno upravljanje pa zanemarijo. S pojavom MS Windows Viste se ta koncept poslavlja in sedaj tudi pri Microsoftu začenjajo bolj ceniti ukazno upravljanje, katerega glavna prednost so skripte.

V operacijskih sistemih MS Windows poznamo tri lupine:

- MS-DOS command (command.com)
- Windows command (cmd.exe)
- PowerShell (psh.exe)

Lupina **MS-DOS command** je bila vgrajena v prve sisteme MS Windows. Pravzaprav gre le dopolnjeno ukazno vrstico operacijskega sistema MS-DOS. V novejše operacijske sisteme je vgrajena le zato, da omogoča izvajanje starejših programov. Lupina MS-DOS command namreč podpira starejše ukaze, ki so namenjeni izvajanju v 16-bitnem okolju. Lahko pa v lupini MS-DOS command uporabljamo tudi vse novejše ukaze, ki pa se bodo izvedli na običajni način in ne v nekakšni 16-bitni različici. Lupina MS-DOS command ne zna delati z dolgimi imeni datotek ampak namesto njih uporablja njihova pripadajoča 8-črkovna imena.

Lupina **Windows command** je tista, ki jo na sistemih z MS Windows uporabljamo danes. Omogoča nam, da samo z ukazi in brez miške nadziramo delovanje operacijskega sistema. V lupino je vgrajen preprost skriptni jezik, ki se je razvil iz tako imenovanih datotek BATCH.

Najnovejša lupina za operacijske sisteme MS Windows je **PowerShell** (med razvojem so jo imenovali „Monad“ shell). Prinaša nov in zelo zmogljiv skriptni jezik in za razliko od lupine Windows Command omogoča učinkovito upravljanje prav z vsemi elementi operacijskega sistema in sistemske programske opreme. S tem odpravlja potrebo po učenju kompleksnejših skriptnih jezikov, kot je npr. VBScript, samo zato, da bi lahko izvedli nekatere osnovne operacije (npr. tvorili bližnjico na namizju in podobno).

Tudi v operacijskih sistemih, ki so se razvili iz **Unix-a** (npr. Linux), obstaja več različnih lupin:

- Bourne shell (sh),
- C shell (csh),
- Korn shell (ksh),
- TC shell (tcsh),
- Bourne Again shell (bash).

Med naštetimi je najstarejša lupina **Bourne shell**, po kateri se zgledujejo vse druge naštete lupine. Na Linuxu je privzeta lupina **Bash**, ki je združljiva z Bourne shell in ustreza standardu IEEE POSIX P1003.2/ISO 9945.2. V primerjavi z Bourne shell je izboljšana glede programiranja in interaktivne uporabe. Lupina Bash vsebuje močan skriptni jezik.

Skriptni jeziki se od programskih jezikov precej razlikujejo.

- Programski jeziki so v splošnem mnogo močnejši od skriptnih.
- Programski jeziki se prevajajo v izvršljive (strojne) programe, ki so težko prenosljivi na druge procesorje in druge operacijske sisteme. Skripte se ne prevajajo v strojni jezik in so zato neodvisne od procesorja ter enostavno prenosljive.
- Programi v skriptnih jezikih se tolmačijo, zato za njihovo izvajanje potrebujemo tolmača (interpreter). Tolmač v osnovi ukaze v skripti sproti razpoznava in jih izvaja, zato so skripte počasnejše od programov.

Ločimo skriptne jezike, za katere je tolmač vgrajen v lupino in bolj splošne skriptne jezike, za katere moramo tolmača namestiti posebej. Prednost slednjih je v tem, da niso odvisni od lupine in so zato načeloma prenosljivi med različnimi operacijskimi sistemi. Popularni splošni skriptni jeziki so Javascript, Perl, Python, Ruby, Tcl, Tk, VBScript in drugi.

Skripte so navadne tekstovne datoteke (običajno so dovoljeni le ASCII znaki). Zato jih lahko pišemo tudi v najpreprostejšem urejevalniku besedil in tudi pri delu na daljavo. Na operacijskem sistemu Windows obstaja preprost urejevalnik v tekstovnem načinu z imenom `edit`. Na Linuxu je več takih urejevalnikov besedil, skoraj vedno je na voljo urejevalnik `vi`, ki pa zahteva kar precej privajanja in učenja. Popularen urejevalnik, ki zna delati v tekstovnem načinu, je `emacs`. Če smo na računalnik prijavljeni lokalno, imamo na voljo tudi številne druge urejevalnike, ki so del grafičnega vmesnika (npr. gnome na Linuxu vsebuje dober urejevalnik `gedit`) ali pa namenski urejevalniki namenjeni programiranju (npr. `UltraEdit` in `EditPad` na sistemih Microsoft Windows).

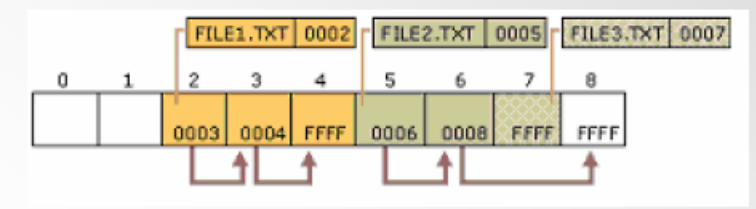
Skripte so najbolj učinkovite, če so tudi vsi podatki shranjeni v **navadnih tekstovnih datotekah**. Besedilo v njih naj bo preprosto strukturirano tako, da je vsaka vrstica en zapis. Tabele naredimo v formatu CSV, to pomeni, da so posamezna polja ločena z vejico (oz. podpičjem). V zadnjem času je sicer zelo popularno, da se razna poročila tvorijo v formatu HTML, da se lepo prikažejo v brskalniku, vendar pa je vsaka nadaljnja obdelava takih poročil dokaj kompleksna. Če gre za zares veliko količino podatkov, ki jo je brez neke kompleksnejše strukture težko obvladovati, pa je najbolje, da za oblikovanje uporabimo format XML, ki ga moderni skriptni jeziki vedno bolje podpirajo.
Skripte so nepogrešljivo orodje sistemskih administratorjev. Njihovo delo se npr. od dela programerjev precej razlikuje, še najbolj opazno je, da sistemski administrator pogosto ponavlja iste operacije (npr. vsakodnevna izdelava varnostnih kopij sistema). Zato je pomembno, da sistemski administrator pozna načine, kako svoje delo v čim večji meri avtomatizirati. Avtomatizacija je koristna iz več vidikov:

- razbremeni administratorja,
- razbremeni računalniški sistem, saj se lahko opravila razporedijo čez cel dan in ne samo čez tisti del dneva, ko je sistemski administrator v službi,
- dvigne kvaliteto, saj odpadejo razne tipkarske in podobne napake, ki se pojavljajo pri ročnem ponavljanju istih opravil.

4. Datotečni sistemi FAT, VFAT in FAT32

Z oznako FAT danes označujemo datotečni sistem FAT16, ki je bil vpeljan leta 1987 in je bil osnovni datotečni sistem operacijskega sistema MS-DOS 4.0. FAT je kratica, ki pomeni **File Allocation Table**. FAT je razmeroma preprost datotečni sistem, zato je bil idealen za diskete. Je tudi zelo razširjen, saj ga razume večina operacijskih sistemov, zato se še danes uporablja npr. pri spominskih karticah.

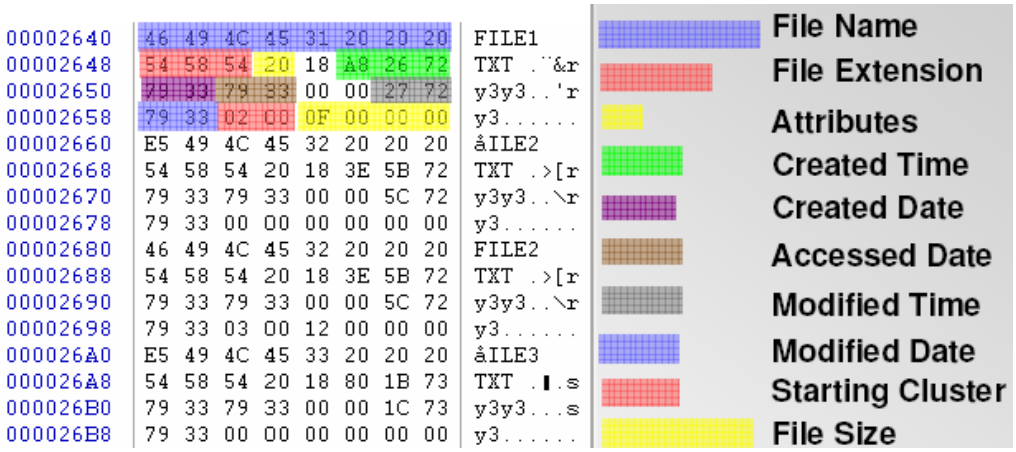
FAT uporablja parcelirano povezano dodeljevanje. Prazni prostor je evidentiran s pomočjo povezanega seznama blokov.



Osnovna enota pri dodeljevanju prostora se imenuje **grozd** (cluster). Grozd je sestavljen iz več blokov (pogosto jih imenujejo sektorji), katerih velikost je pogojena z medijem. Število blokov v enem grozdu je omejeno na 64 (ponekod 128). Če so npr. bloki veliki 512 B, kot je to na diskih, dobimo tako največjo velikost grozda 32 kB (oz. 64 kB).

Bloki so oštevilčeni z 32 bitnim številom, grozdi pa so oštevilčeni le s 16 bitnim številom. Nekatere vrednosti so rezervirane in imajo poseben pomen, tako da je največje število grozdov v enem zvezku 65,526. Na diskih je zato največja možna velikost zvezka okoli 2 GB (oz. 4 GB). Ker je velikost datoteke zapisana kot 32 bitno število, je največja velikost datoteke 4 GB. FAT16 ima še dodatne omejitve, npr. največje dovoljeno število datotek v zvezku je 65536.

Podatki o posameznih datotekah so shranjeni v tabeli FAT, po kateri je datotečni sistem dobil ime. Vsak zapis v tabeli FAT je velik 32 B in vsebuje naslednje zapise: ime datoteke (8 B), končnica (3 B), atributi (1 B), neuporabljeno (1 B), čas in datum nastanka (5 B), datum zadnjega dostopa (2 B), neuporabljeno (2 B), čas in datum zadnje spremembe (4 B), oznaka začetnega grozda (2 B) in velikost datoteke (4 B). V originalni specifikaciji ni bilo časa in datuma nastanka ter datuma zadnjega dostopa, ta polja so bila označena kot neuporabljena in so bila dodana pozneje.



VFAT je bil vpeljan leta 1995 in je po strukturi enak datotečnemu sistemu FAT16. Največja sprememba za uporabnike je podpora dolgim imenom datotek. Z uvedbo dolgih imen se ni povečala velikost zapisa v tabeli FAT, pač pa dolga imena uporabijo več zapisov. Druge spremembe se nanašajo bolj na izvedbo datotečnega sistema kot na njegovo funkcionalnost.

FAT32 je bil vpeljan leta 1996 v operacijskem sistemu Windows 95 OSR2. Grozdi so oštevilčeni z 28-bitnim številom (4 biti od 32 so rezervirani za posebne namene). Teoretična meja za velikost zvezka pri FAT32 zato ni postavljena s številom grozdov ampak s številom blokov, ki so še vedno oštevilčeni z 32 bitnim številom in znaša 2 TB. Če uporabimo manjše grozde, pa seveda tudi število grozdov lahko vpliva na največjo možno velikost zvezka. Velikost datoteke je ostala zapisana kot 32 bitno število, zato je tudi pri FAT32 največja velikost datoteke 4 GB. Največje dovoljeno število datotek v zvezku je 4.194.304.

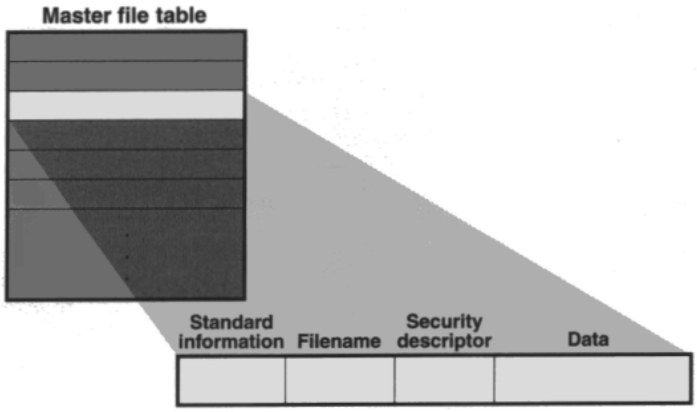
exFAT oz. FAT64 je najnovejša različica datotečnega sistema FAT. Vpeljal jo je Microsoft konec leta 2006 in zaenkrat ostaja zaprta tehnologija (uporabljajo jo npr. SDXC kartice).

5. Datotečni sistem NTFS

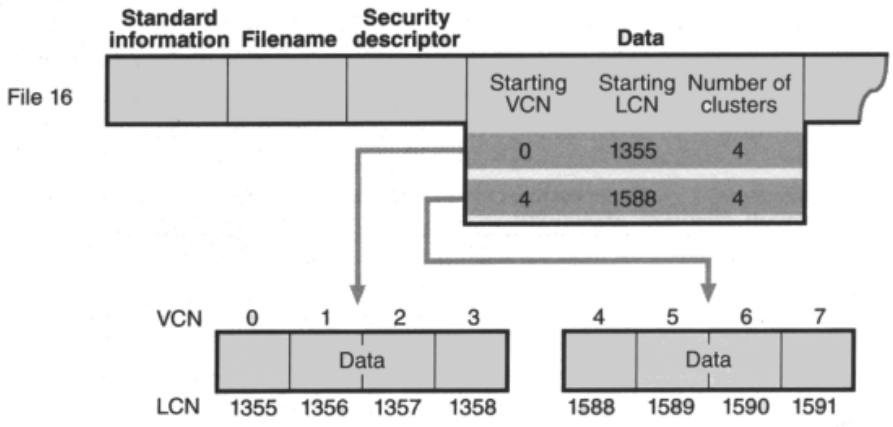
NTFS (NT File System) je bil vpeljan leta 1995 kot osnovni datotečni sistem operacijskega sistema Windows NT. Trenutno je v uporabi verzija 3.1 (nekateri jo imenujejo tudi verzija 5.1), ki jo uporabljajo operacijski sistemi Windows XP, Windows Server 2003 in Windows Vista. Ker Microsoft podrobnosti specifikacije in implementacije NTFS pojmuje kot poslovno skrivnost, je razmeroma slabo podprt v drugih operacijskih sistemih.

V osnovi je NTFS nadgradnja datotečnega sistema FAT32. Ravno tako imamo grozde (clusters). Uporabljena je kombinacija več tehnik dodeljevanja, najdemo lahko elemente zveznega, povezanega in indeksiranega dodeljevanja. Velikost grozda se določi ob formatiranju NTFS in običajno znaša 8 blokov (4 kB pri blokih velikosti 512 B). Velikost datoteke in zaporedna številka grozda sta zapisana s 64 bitno številko, zato so teoretične meje zelo visoke. Velikost zvezka je zaradi 32 bitnih številk blokov še vedno omejena na 2 TB. Vendar pa novejši operacijski sistemi omogočajo, da se zvezek (volume) razteza čez več particij tako, da se bloki v vsaki številčijo od 0 naprej. Na ta način lahko tvorimo zelo velike NTFS zvezke (trenutno je omejitev 16 TB).

Osnova datotečnega sistema NTFS je glavna tabela datotek (MFT, Master File table), ki podobno kot tabela FAT vsebuje atribute datoteke. Razlika pa je, da se tudi vsebina datoteke (pravzaprav kazalci na vsebino) obravnava kot en atribut. Če je atribut (npr. število kazalcev na vsebino) prevelik, da bi se shranil v MFT, potem se shrani izven tabele MFT kot poseben zapis imenovan nerezidenčni atribut.



Do podatkov v posameznih datotekah pridemo preko kazalcev, ki jih imenujemo VCN-ji (Virtual Cluster Numbers). Za vsako posamezno datoteko se štejejo od 0 naprej. VCN-ji na disku niso nujno zvezni, preslikujejo se v skupine zveznih LCN-jev.



NTFS ima posebno tehniko za „redke" datoteke. Grozdi, v katerih so same ničle se v resnici ne zapišejo na disk. Namesto tega se v (običajno zveznem) zaporedju VCN-jev pusti presledke. Če NTFS pri branju datoteke zazna presledek med VCN-ji, NTFS v uporabnikov medpomnilnik vpiše ustrezno število ničel.

Poleg datotek z uporabniško vsebino so na disku tudi datoteke s tako imenovanimi metapodatki (metadata), ki dodatno opisujejo sistem, imenike in posamezne datoteke. Vsaka datoteka z metapodatki vsebuje 64 bitno referenco datoteke. Ta je sestavljena i z dveh delov in vsebuje številko datoteke in sekvenčno številko. Številka datoteke ustreza poziciji datotečnega zapisa v MFT. Sekvenčna številka se poveča pri vsaki uporabi datotečnega zapisa.

Pomembne lastnosti NTFS so še:

- NTFS je datotečni sistem z dnevnikom.
- NTFS ima vgrajeno kriptiranje podatkov,
- NTFS omogoča stiskanje podatkov (compression unit = blok 16 zaporednih grozdov),
- NTFS omogoča omejevanje prostora posameznim uporabnikom (disk quotas),
- NTFS lahko za hitrejše iskanje datotek tvori imenik vseh datotek v obliki urejenega drevesa,
- NTFS omogoča priklop logičnih particij na poljubno mesto v imeniški strukturi, tako da ni vedno potrebno uporabiti novo črko,
- NTFS omogoča simbolične povezave (več imen za isto mapo oz. datoteko), ki pa imajo na Windows XP in Windows Server 2003 precejšnje omejitve in stranske učinke, dobro delajo šele na sistemu Windows Vista,
- NTFS omogoča združevanje particij v tako imenovani **stripe set** oz. **RAID 0.** To je poseben način povezovanja več fizičnih particij v eno logično particijo. Vse particije morajo biti enako velike in vsaka na drugem disku. Uporablja se za povečanje učinkovitosti, saj se podatki hkrati zapisujejo na več diskov.

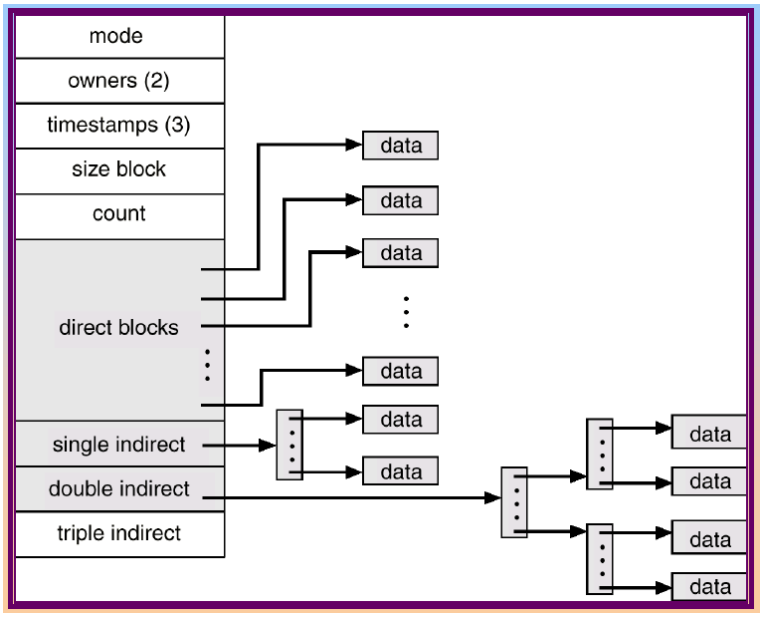
6. Datotečna sistema ext2 in ext3

Iz prvotnega operacijskega sistema imenovanega Unix, so izšle številne različice npr. Linux, ki uporabljajo različne datotečne sisteme, ki pa so vsi do neke mere podobni originalnemu. Za Linux sta značilna datotečna sistema ext2 in ext3. Sistem ext3 uporablja dnevnik, ext2 pa ne.

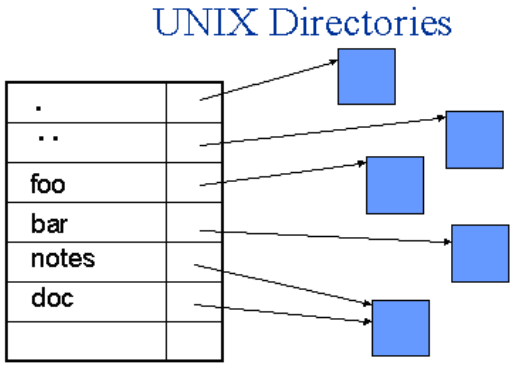
Prostor je razdeljen v bloke, ki so večkratnik vnaprej določenega fragmenta. Vsi bloki, ki pripadajo neki datoteki, so polni, razen zadnjega bloka, ki je zaseden le do nekega večkratnika fragmenta. Če pričakujemo majhne datoteke, izberemo majhno velikost fragmenta. Tipična velikost bloka je 4 kB.

Uporabljeno je **indeksirano dodeljevanje prostora**. Podatki o posameznih datotekah so shranjeni v posebnih blokih, ki jih imenujemo i-vozel (i-node). Poleg osnovnih informacij o datoteki (lastnik, velikost itd.) so v njem shranjeni tudi indeksi blokov. Vsak I-vozel vsebuje 15 kazalcev na bloke. Prvih 12 kaže na direktne bloke. Naslednji trije kažejo na indirektne bloke. Prvi kaže na single indirect block — indeksni blok, ki vsebuje naslove blokov s podatki. Drugi je double-indirect-block pointer, naslov bloka, ki vsebuje kazalce na bloke, v katerih so kazalci na podatkovne bloke. Zadnji je triple indirect pointer (ta je potreben le pri datotekah večjih od 4 GB). I-vozli imajo tudi polje, ki določa ali gre za datoteko, imenik, simbolično povezavo itd.

Imeniki so narejeni kot posebne vrste datotek. Vnosi v imenikih so spremenljive dolžine (različno dolga imena datotek). Vsak vnos vsebuje najprej dolžino vnosa, ime datoteke in številko ustreznega i-vozla. Uporabnik dostopa do datotek preko njihovih imen, s sistemskega vidika pa je datoteka definirana z i-vozlom. Operacijski sistem preslika ime datoteke v ustrezen i-vozel.



Vsak element v imeniku je tako imenovana trda povezava (hard link). Prvi dve imeni datotek v imeniku sta vedno enaki nizu "." oziroma "..", preko njiju pridemo do številke i_vozlov tekočega oziroma nadrejenega imenika. Več datotek si lahko deli isti i-vozel, torej ima lahko vsaka datoteka več imen. Če zbrišemo eno, se zbrišejo vse! Linux pozna tudi simbolične, mehke povezave (symbolic links), s katerimi prav tako dosežemo, da ima lahko ista datoteka več imen. Vendar pa se te obnašajo kot bližnjice do datotek. Če simbolično povezavo zbrišemo, datoteka, na katero je kazala, ostane nespremenjena.



notes and doc are the same file

Korenski datotečni sistem je vedno dostopen na določeni napravi. Druge datotečne sisteme je mogoče priklopiti (integrirati) v hierarhijo imenikov korenskega datotečnega sistema. Operacijski sistem za identifikacijo datotek uporablja par (številka logične naprave, številka i-vozla). Številke logičnim napravam določa datotečni sistem. I-vozli so številčeni zaporedno.

7. Upravljanje z datotečnim sistemom iz lupine Windows command

Pomembni **spremenljivki okolja** v operacijskem sistemu Microsoft Windows sta spremenljivka PATH in spremenljivka PATHEXT. V spremenljivki PATH je spisek imenikov, ki se uporabljajo pri iskanju programov. Posamezni imeniki so ločeni z podpičji. Pri iskanju programov se program vedno najprej išče v trenutnem imeniku. Spremenljivka PATHEXT vsebuje končnice datotek, ki jih sistem obravnava kot končnice izvajalnih datotek. Pomembni sta tudi spremenljivki DATE in TIME, vsebujeta trenutni datum in čas.

Zanimiva spremenljivke okolja, ki je tesno povezane z lupino Windows command, je spremenljivka PROMPT. V spremenljivki PROMPT je zapisan tekst, ki se izpiše kot pozivnik v ukazni vrstici.

V lupini Windows command spremenljivke okolja izpišemo z ukazom SET.

Ukaze za upravljanje z datotečnim sistemom iz lupine Windows command lahko razdelimo v dve skupini: eni so vgrajeni v samo lupino (npr. ukaza CD in DIR), drugi pa so izvršljiv program v imeniku C:\WINDOWS\system32 (npr. ukaza ATTRIB in SUBST). Za uporabnika je bolj ali manj vseeno, v katero od teh dveh skupin spada določen ukaz.

Osnovni ukaz pri upravljanju z datotečnim sistemom iz lupine Windows command je ukaz CD. Z njim se **premikamo med imeniki**. Ukaz CD ima tudi daljše ime CHDIR. Z ukazom CD .. se pomaknemo en imenik višje. Z ukazom CD \ gremo na vrh datotečnega sistema. Ukaz CD brez parametrov izpiše polno pot do trenutnega imenika.

Nekoliko manj znana ukaza za premikanje med imeniki sta PUSHD in POPD. Ukaz PUSHD deluje enako kot ukaz CD, le da si predhodno zapomni trenutni imenik. Ukaz POPD nas prestavi v imenik, v katerem smo nazadnje izvedli ukaz PUSHD. Lupina Windows command ima tudi zanimiv ukaz TREE, ki grafično prikaže drevesno strukturo imenikov.

Nov imenik naredimo z ukazom MD (daljše ime ukaza je MKDIR). Tvorimo lahko imenike v poljubni globini od trenutnega mesta, saj se avtomatično tvorijo vsi morebitni neobstoječi vmesni imeniki. Na primer: MD A\B\C tvori imenik A, v njem tvori imenik B, v njem pa nato imenik C. Če imenik ali pa kakšna datoteka z navedenim imenom že obstaja, potem ukaz sporoči napako in ne tvori ničesar. Za **brisanje imenikov** je na voljo ukaz RD (daljše ime ukaza je RMDIR). Ta ukaz pozna kretnici /Q in /S, katerih pomen je enak kot pri ukazu DEL (opisano v nadaljevanju). Če ne podamo kretnice /S, lahko zbrišemo le prazen imenik.

Uporabnik vidi datotečna sistema FAT in NTFS kot zbirko zvezkov (volumes). Posamezni zvezki so označeni s črkami od A do Z. Vsak zvezek ima lahko tudi kratko opisno ime. **Oznako in ime zvezka** izpišemo z ukazom VOL. **Ime zvezka** določimo, spremenimo ali zbrišemo z ukazom LABEL.

Z ukazom SUBST tvorimo **navidezne zvezke**, ki delujejo kot bližnjica do določenega imenika. Navidezni logični zvezki obstajajo samo tako dolgo, dokler se ne odjavimo. Ob naslednji prijavi jih moramo ponovno tvoriti, podobno kot pri priključitvi omrežnih pogonov. Nekaj primerov:

```
SUBST R: .   (tvorimo navidezni zvezek R, ki je bližnjica do trenutnega imenika)
SUBST R: /D   (zbrišemo navidezni logični zvezek R)
SUBST R: "C:\Documents and Settings\User\My Documents"
SUBST T: "C:\Recycler\S-1-5-21-2343136237-4081588881-2537991623-3014"
```

Na sistemu Windows XP, Windows Server 2003 in novejših lahko namesto ukaza SUBST uporabimo bolj zmogljiv ukaz LINKD.

Ukaz DIR **izpiše vsebino trenutnega imenika** oz. vsebino podanega imenika. Ima nekaj kretnic, med katerimi so najbolj uporabne naslednje:

- DIR /A (izpiše vse, tudi sistemske datoteke in imenike)
- DIR /AH (izpiše samo skrite datoteke in imenike)
- DIR /AD (izpiše samo imena podimenikov)
- DIR /A-D (izpiše samo imena datotek, ne pa tudi imena podimenikov)
- DIR /B (izpiše samo seznam z imeni datotek, primerno za obdelavo v skriptih)
- DIR /S (rekurzivno izpiše tudi informacije v vseh podimenikih)
- DIR /ON (datoteke uredi po imenih)
- DIR /OS (datoteke uredi po velikosti)
- DIR /OD (datoteke uredi po času)
- DIR /TA (namesto časa spremembe datoteke upošteva in izpiše čas zadnjega dostopa)

Ukaz DIR lahko uporabimo tudi za **iskanje datotek**. Na primer, vse datoteke *.TXT v logičnem zvezku D: dobimo z naslednjim ukazom:

```
DIR /B/S D:*.TXT
```

Premikanje in preimenovanje datotek med imeniki izvedemo z ukazom MOVE. Naenkrat lahko premaknemo več datotek, preimenujemo pa lahko samo eno datoteko. Samo preimenovanju datotek pa je namenjen ukaz REN (daljše ime ukaza je RENAME), s katerim lahko naenkrat spremenimo končnico tudi več datotekam naenkrat. Na primer, ukaz REN *.JPEG *.JPG preimenuje vse datoteke s končnico JPEG v datoteke s končnico JPG.

Datoteke kopiramo z ukazom COPY. Pri kopiranju lahko datotekam spremenimo imena, npr. ukaz COPY *.TXT *.OLD prekopira vse datoteke *.TXT v datoteke *.OLD.

Datoteke zbrišemo z ukazom DEL (drugo ime tega ukaza je ERASE). Če brišemo z ukazom DEL se datoteke NE PRESTAVIJO V KOŠ AMPAK V RESNICI ZBRIŠEJO! Če zahtevamo brisanje vseh datotek v neki mapi nas računalnik pred brisanjem vpraša, ali res želimo to narediti. Zbrišejo se le datoteke v trenutnem imeniku, ne pa tudi podimeniki. Pri ukazu DEL so uporabne naslednje kretnice:

DEL /P (računalnik vedno zahteva potrditev brisanja)
DEL /Q (računalnik nikoli ne zahteva potrditve brisanja, tudi če brišemo vse datoteke v imeniku)
DEL /S (rekurzivno zbriše tudi vse podimenike in datoteke v njih)

Vsebino tekstovne datoteke izpišemo z ukazom TYPE ali pa MORE. Pri ukazu MORE se izpis na koncu vsake strani ustavi, dokler ne pritisnemo poljubno tipko.

Datotekam in mapam lahko določimo **atribute** in **zaščito**. Atribute podanih datotek (oz. vseh datotek v trenutnem imeniku, če ne podamo ničesar) izpišemo z ukazom ATTRIB. Atributi so Read-only (R), Hidden (H), System (S) in Archive (A). Z ukazom ATTRIB lahko atribute tudi spremenimo, npr. naslednji ukaz nastavi Read-only in zbriše Hidden vsem *.BAT datotekam:

```
ATTRIB +R -H *.BAT
```

Najpomembnejši atribut je Read-only. Datoteke, ki imajo atribut Read-only, so zaščitene pred brisanjem, preimenovanjem, premikanjem in spreminjanjem vsebine. Lahko pa takšno datoteko prekopiramo in spreminjamo njeno kopijo.

Zaščito datotek spreminjamo z ukazom CACLS. Datotečni sistem FAT ne pozna zaščit, zato ta ukaz deluje le v datotečnem sistemu NTFS. Tukaj je nekaj primerov.

Dodaj zaščito „Read-only“ datoteki `Imena.txt` (zaščita je različno od atributa Read-only):

```
CACLS Imena.txt /E /G "Power Users":R
```

Postavi zaščito „Full Control permission“ datoteki `Imena.txt`:

```
CACLS Imena.txt /E /G "Tajnice":F
```

Postavi zaščito „Full Control permission“ imeniku `C:\Dokumenti` in vsem podimenikom:

```
CACLS C:\Dokumenti /E /T /C /G "Studenti":F
```

Na koncu naštejmo še nekaj drugih uporabnih ukazov (podrobnejši opis dobite z ukazom `HELP`):
`CHKNTFS` – status zvezkov NTFS
`CIPHER` - šifriranje in dešifriranje datotek in imenikov v datotečnem sistemu NTFS
`COMPACT` - stiskanje datotek na particiji NTFS
`EXPAND` - raztegovanje stisnjenih datotek na particiji NTFS
`CONVERT` - pretvorba datotečnega sistema FAT v NTFS (obratno ne gre!)

8. Upravljanje z datotečnim sistemom iz lupine Bash

Pomembni **spremenljivki okolja** v operacijskem sistemu Linux sta spremenljivka `HOME` in spremenljivka `PATH`. V spremenljivki `HOME` je shranjeno, kje je uporabnikov osnovni imenik. V spremenljivki `PATH` je spisek imenikov, ki se uporabljajo pri iskanju programov. Posamezni imeniki so ločeni z dvopičji. Iz varnostnih razlogov je dokaj običajno, da spremenljivka `PATH` ne vsebuje trenutnega imenika. Zato moramo pri zagonu programov v trenutnem imeniku napisati pot do njega, npr. `./program`.

Zanimiva spremenljivka okolja, ki je tesno povezana z lupino Bash, je spremenljivka `PS1`. V njej je zapisan tekst, ki se izpiše kot pozivnik v ukazni vrstici.

V lupini Bash vse spremenljivke okolja izpišemo z ukazom `set`.

Ukaze za upravljanje z datotečnim sistemom iz lupine Bash lahko razdelimo v več skupin:

- ukazi vgrajeni v lupino Bash,
- programi v imeniku `/bin`,
- programi v imeniku `/usr/bin`,
- programi v imeniku `/sbin`,
- programi v imeniku `/usr/sbin` ali pa
- okrajšave (aliasi).

Ukazi v imenikih `/sbin` in `/usr/sbin` so namenjeni le administratorjem. V teh imenikih se npr. nahajata ukaz za ugašanje računalnika (`/sbin/shutdown`) in dodajanje uporabnikov (`/usr/sbin/useradd`).

Katerega tipa je posamezen ukaz izvemo, če vpišemo ukaz `type`. Ukaz `type` ne deluje za ukaze v imenikih `/sbin` in `/usr/sbin`. Tukaj je nekaj primerov:

```
$ type type
type is a shell builtin
$ type cd
cd is a shell builtin
```

```
$ type ls
ls is /bin/ls
$ type find
find is /usr/bin/find
$ type dir
dir is aliased to `ls -l --color=tty'
$ type useradd
bash: type: useradd: not found
```

Bash si uporabljene ukaze zapomni v zgoščevalni tabeli, da mu jih naslednjič ni potrebno iskati. V tem primeru ukaz `type` to sporoči. Na primer:

```
$ type ls
ls is hashed (/bin/ls)
```

S pomočjo okrajšav lahko tvorimo svoje ukaze. Če npr. želimo, da se vsebina imenika izpiše v barvah in z vsemi podrobnostmi, uporabimo ukaz `ls -l --color=tty`. Da nam ni treba vedno znova pisati tega dolgega ukaza, lahko tvorimo npr. okrajšavo `dir` na naslednji način:

```
alias dir='ls -l --color=tty'
```

Opis posameznega ukaza dobimo, če napišemo `man` ime_ukaza ali pa `info` ime_ukaza.

Na Linuxu imamo na voljo tudi dva druga ukaza, s katerima ugotovimo **kje se nahaja izvršilna datoteka določenega programa**. Prvi je ukaz `which`, Ta pregleda vse imenike navedene v spremenljivki PATH in sporoči, kje je našel iskani program. Ukaz `whereis` pa išče programe bolj intenzivno in sicer v vseh imenikih, kjer se programi na Linuxu običajno nahajajo. Zato najde tudi tiste, do katerih nimamo nastavljene poti. Izpišejo se vsi zadetki povezani z iskanim programom in to ne samo pot do izvršilne datoteke ampak tudi, kje se nahaja izvorna koda (če ta obstaja) in kje se nahajajo datoteke s pomočjo za ta ukaz oz. program. Omenimo še. da obstaja tudi ukaz `whatis`, ki pa pregleda le datoteke s pomočjo in vrne naslove tistih poglavij, ki so povezani z določenim ukazom.

Osnovni ukaz pri upravljanju z datotečnim sistemom iz lupine Bash je ukaz `cd`. Z njim se **premikamo med imeniki**. Z ukazom `cd ..` se pomaknemo en imenik višje. Z ukazom `cd /` gremo na vrh datotečnega sistema. Ukaz `cd ~` nas premakne v naš osnovni imenik, ukaz `cd -` pa nas prestavi v imenik, kjer smo bili pred zadnjo spremembo. Lupina Bash pozna tudi ukaza pushd in popd. Ukaz `pushd` deluje enako kot ukaz `cd`, le da si predhodno zapomni trenutni imenik. Ukaz `popd` nas prestavi v imenik, v katerem smo nazadnje izvedli ukaz `pushd`. Polno pot do trenutnega imenika izpišemo z ukazom pwd.

Nov imenik naredimo z ukazom `mkdir`. Če imenik ali pa kakšna datoteka z navedenim imenom že obstaja, poem ukaz sporoči napako in ne tvori ničesar. Če uporabimo kretnico `-p` lahko tvorimo imenike v poljubni globini od trenutnega mesta, saj se avtomatično tvorijo vsi morebitni neobstoječi vmesni imeniki. Pri uporabi kretnice `-p` ne dobimo napake, če imenik s podanim imenom že obstaja, ne sme pa obstajati neka datoteka z istim imenom. Na primer: `mkdir -p a\b\c` tvori imenik `a`, v njem tvori imenik `b`, v njem pa nato imenik `c`. Za **brisanje imenikov** je na voljo ukaz `rmdir`. Zbrišemo lahko le prazen imenik.

Spisek vseh podimenikov in njihove velikosti dobimo z ukazom `du`. Tukaj je nekaj primerov:

- `du -h` (velikosti izpiše na razumljiv in nedvoumen način, npr. 13k, 1.1M itd.)
- `du -S` (k velikosti nekega imenika ne prišteje velikosti njegovih podimenikov)
- `du -s` (izpiše samo skupno velikost)

Velikost in zasedenost diskov izpišemo z ukazom `df`. Tukaj je primer izpisa:

```
Dat. sist.           1K-blokov      Upor. Na voljo Upo% Priklopljeno na
/dev/hda2             29753588   4184288  24033488  15% /
/dev/hda1               101086     16688     79179  18% /boot
/dev/hda3             30233928  16375756  12322360  58% /home
/dev/hda5              3968092   1100020   2663248  30% /var
```

Ukaz `ls` **izpiše vsebino trenutnega imenika** oz. vsebino podanega imenika. Ukaz ls brez kretnic tvori večstolpični izpis, v katerem so le imena datotek urejena po imenih. Najbolj uporabne so naslednje kretnice:

- `ls -a` (izpiše vse, tudi imenike, ki se začnejo s piko)
- `ls -b` (pred posebne znake, npr. presledke, doda znak \, uporabno v skriptih)
- `ls -p` (k imenom doda oznako, npr. * pomeni izvršilno datoteko, / pa imenik)
- `ls -l` (tvori podroben izpis, izpiše se zaščita, lastnik, velikost, zadnja sprememba itd.)
- `ls -1` (izpiše v obliki seznama in ne stolpcev, primerno za obdelavo v skriptih)
- `ls -R` (rekurzivno izpiše tudi informacije v vseh podimenikih)
- `ls -X` (datoteke uredi po končnicah)
- `ls -S` (datoteke uredi po velikosti)
- `ls -t` (datoteke uredi po času zadnje spremembe)
- `ls -tu` (datoteke uredi po času zadnjega dostopa)

Še nekaj opomb glede ukaza `ls`. Kretnica `-F` je enaka kretnici `-p`. Kretnica `-o` je podobna `-l`, edina razlika je, da v izpisu ni grupe, ki ji pripada lastnik datoteke. Kretnica `-r` obrne smer sortiranja. Če izpis preusmerimo v datoteko, se vedno tvori seznam in ne izpis po stolpcih. Ukaz `ls *` vedno deluje rekurzivno, torej izpiše tudi informacije v vseh podimenikih.

Za **iskanje datotek** v Bashu uporabljamo ukaz `find`. Tukaj je nekaj primerov:

Poišči vse datoteke `*.txt` v trenutnem imeniku in v vseh podimenikih:
```
find . -name "*.txt"
```

Poišči vse datoteke v `/etc` in podimenikih, ki imajo čas zadnjega dostopa manjši od 60 minut:
```
find /etc -amin -60
```

Poišči vse datoteke v `/etc` in podimenikih, ki imajo čas zadnjega dostopa večji ali enak 7 dni (<6):
```
find /etc -atime +6
```

Poišči vse datoteke v `/var` in podimenikih, ki so bile zadnjič spremenjene pred manj kot 60 min:
```
find /var -mmin -60
```

Poišči vse datoteke v `/var` in podimenikih, ki so bile zadnjič spremenjene pred 30 ali več dnevi:
```
find /var -mtime +29
```

Poišči vse datoteke v imeniku `Vaje` in njegovih podimenikih, katerih celotno ime (pot + ime + končnica) ustreza podanemu regularnemu izrazu (vsebuje niz znakov „vso“):
```
find Vaje -regex ".*vso.*"
```

Navedemo lahko več različnih kriterijev, po katerih iščemo datoteke. Ukaz `find` ne vrne datotek, ki se začnejo s piko, razen če to izrecno zahtevamo, npr.:
```
find /home/meolic -name ".bash*"
```

Premikanje in preimenovanje datotek med imeniki izvedemo z ukazom `mv`. Naenkrat lahko premaknemo več datotek, preimenujemo pa lahko samo eno datoteko. **Datoteke kopiramo** z ukazom `cp`. Če želimo prekopirati celoten imenik in vse njegove podimenike, moramo uporabiti kretnico `-r`. Pri premikanju in preimenovanju datotek se ohranijo podatki o zaščiti, lastniku in času nastanka. Če želimo te podatke ohraniti tudi pri kopiranju datotek, uporabimo ukaz `cp -p`.

Datoteke zbrišemo z ukazom `rm`. Če brišemo z ukazom `rm` se datoteke NE PRESTAVIJO V KOŠ AMPAK V RESNICI ZBRIŠEJO! Zbrišejo se le datoteke v trenutnem imeniku, ne pa tudi podimeniki. Če zahtevana datoteka ne obstaja, dobimo obvestilo o napaki. Pri ukazu `rm` so uporabne naslednje kretnice:

`rm -f` (če datoteka ne obstaja, ne izpiše obvestila o napaki)
`rm -i` (računalnik pred vsakim brisanjem zahteva potrditev)
`rm -r` (rekurzivno zbriše tudi vse podimenike in datoteke v njih)

Vsebino tekstovne datoteke lahko izpišemo z ukazoma `cat` in `more`. Pri ukazu `more` se izpis na koncu vsake strani ustavi, dokler ne pritisnemo poljubno tipko. Imamo tudi ukaz `less`, ki pa je namenjen pregledovanju tekstovne datoteke in ne njenemu izpisu. Ukaz `less` je podoben preprostemu urejevalniku, vendar pa vsebine ne moremo spreminjati. Po besedilu se lahko premikamo gor in dol, lahko pa tudi iščemo besede tako, da vnesemo „/iskana_beseda“. Pregled datoteke zaključimo s tipko `q`.

Datoteke in imeniki na Linuxu (velja za datotečne sisteme ext2 in ext3) imajo postavljeno **zaščito**, ki je sestavljena iz treh delov: zaščita za lastnika (user), zaščita za grupo (group) in zaščita za vse ostale (other). Zaščita je prikazana v prvem stolpcu običajnega izpisa vsebine imenika. Sestavlja jo 9 znakov, prvi znak v vrstici ni zaščita ampak določa vrsto datoteke. Primer:

```
-rw-r-----  1 meolic meolic   48 mar  7 15:06 Imena.txt
drwxr-xr-x  2 meolic meolic 4096 mar  7 15:10 Novo
-rwx--x--x  1 meolic meolic 2609 mar  7 15:09 skripta
-rw-rw----  1 meolic meolic   49 mar  7 15:08 Urejeno.txt
```

V prikazanem primeru ima datoteka `Imena.txt` zaščito `rw-` za lastnika, `r--` za grupo in `---` za ostale. Imenik `Novo` ima zaščito `rwx` za lastnika ter `r-x` za grupo in ostale. Pomen znakov `r`, `w` in `x` je pri datotekah in imenikih različen.

Pri datotekah imajo znaki `rwx` naslednji pomen:

- `r` - datoteko lahko preberemo in kopiramo,
- `w` - datoteko lahko preimenujemo, premaknemo, zbrišemo in spremenimo njeno vsebino,
- `x` - datoteko lahko izvajamo.

Pri imenikih pa je pomen znakov `rwx` naslednji:

- `r` - vsebino imenika lahko izpišemo,
- `w-` v imenik lahko dodamo nove datoteke,
- `x` - lahko dostopamo do datotek v imeniku.

Zaščito datotek in imenikov spremenimo z ukazom `chmod`. Naenkrat lahko spremenimo zaščito več datotekam. Tukaj je nekaj primerov:
`chmod u=r Imena.txt` (lastniku se zaščita nastavi na `r--`)
`chmod g+w Imena.txt` (grupi se doda pravica za pisanje)
`chmod o-x Imena.txt` (ostalim se odvzame pravica za izvajanje)
`chmod a=rw Imena.txt` (lastniku, grupi in ostalim se nastavi zaščita `rw-`)

Koristna sta tudi ukaza `chown` in `chgrp`, s katerima spremenimo lastnika oz. grupo za posamezno datoteko ali za skupino datotek.

V skriptih sta zelo uporabna tudi ukaza `basename` in `dirname`. Ukaz `basename` iz imena datoteke odstrani morebitna imena imenikov pred imenom, ukaz `dirname` pa obdrži samo imena imenikov. Če ni naveden nobeden imenik, `dirname` vrne eno piko. Tukaj sta primera:

```
$ basename /home/meolic/vss-vso/Urejeno.txt
Urejeno.txt

$ dirname /home/meolic/vss-vso/Urejeno.txt
/home/meolic/vss-vso
```

9. Priklop particij in pomnilniških medijev

Operacijski sistem MS Windows samodejno poišče in priklopi vse particije z datotečnim sistemom VFAT, FAT32 in NTFS, ki so na lokalnih diskih.

Po priklopu si sistem MS Windows zapomni oznako particije, ki jo imenujemo GUID (Globally Unique Identifier). GUID je naključna oznaka, ki se dodeli ob formatiranju particije. Izpišemo jo z ukazom `MOUNTVOL`. Dobljeni izpis je oblike:

```
\\?\Volume{GUID}\
```

S pomočjo GUID lahko sistemski in uporabniški programi v vsakem trenutku preverijo, ali je pod določeno črko (oz. potjo) še vedno ista particija, ali pa je uporabnik priklopil kaj drugega. Pri Microsoftu oznako GUID izkoriščajo tudi pri postopku aktivacije licenčne programske opreme.

Oznako ene določene particije dobimo s kretnico /L.

Spisek vseh trenutno priklopljenih particij in njihove oznake GUID dobimo (zanimivo!) z ukazom `MOUNTVOL /?`

Primer:

```
C:>MOUNTVOL D: /L
\\?\Volume{e621f1f0-cbee-11d6-a8a9-06d6172696f}\
```

Z ukazom `MOUNTVOL` lahko spremenimo točko priklopa tako, da namesto nove črke uporabimo obstoječ in prazen podimenik na (obvezno!) NTFS particiji. Recimo, da v prejšnjem primeru D: kaže na DVD enoto. Potem jo npr. v imenik `C:\DVD` priklopimo z naslednjim ukazom:

```
MOUNTVOL  C:\DVD  \\?\Volume{e621f1f0-cbee-11d6-a8a9-06d6172696f}\
```

Enako operacijo lahko izvedemo z grafičnim orodjem Disk Management, ki ga najdemo med administratorskimi orodji v nadzorni plošči ali pa ga enostavno poženemo tako, da v ukazno vrstico napišemo `DISKMGMT.MSC`

Ukaz `MOUNTVOL /N` ukine avtomatsko priklapljanje najdenih particij, ukaz `MOUNTVOL /E` pa to lastnost spet vklopi.

Na sistemih MS Windows imamo na voljo še dve družini ukazov in sicer FSUTIL ter RSM.

FSUTIL FSINFO – izpiše podatke o particijah
FSUTIL VOLUME DISKFREE – preverimo, koliko je še praznega prostora na particiji

Tudi operacijski sistem Linux samodejno najde vse particije, ki so na lokalnem disku in jim dodeli oznake hdc1, hdc2, itd. za diske PATA/SATA ter sda1, sda2, itd. za diske SCSI in particije na napravah USB. Najdene particije moramo sami priklopiti z ukazom mount.

Ukaz mount brez kretnic izpiše vse priklopljene particije, to lahko pogleda tudi navadni uporabnik.

```
$ mount
/dev/hdc2 on / type ext3 (rw)
/dev/hdc3 on /home type ext3 (rw)
/dev/hdc5 on /vfat type vfat (rw,nosuid,nodev,umask=000)
/dev/hdc6 on /data type ext3 (rw)
```

Tukaj je primer ukaza, ki priklopi particijo /dev/hdc3 kot imenik /windows:

```
mount -t vfat /dev/hdc3 /windows
```

Preden izvedemo ta ukaz, moramo tvoriti imenik /windows. Ni nujno, da je prazen, vendar pa v tem času, ko bo tam priklopljena particija /dev/hdc3, lokalni podatki ne bodo dostopni - se pa ne bodo zbrisali. Da ne bi vedno znova tipkali ukaza mount, lahko podatke o priklopu shranimo v datoteko /etc/fstab. Tam je za vsak priklop ena vrstica. Primer:

```
/dev/hdc2     /            ext3    defaults   1 1
/dev/hdc3     /home        ext3    defaults   1 2
/dev/hdc5     /windows     vfat    defaults,owner,umask=000 0 0
```

Računalnik bo vse najdene particije, ki ustrezajo zapisu v datoteki /etc/fstab samodejno priklopil. Z opcijami v četrtem polju lahko vplivamo na to, kdo in kako bo imel dovoljenje za dostop do particije. Več opcij ločimo z vejicami, ne smemo pa uporabljati presledkov. Pri priključitvi particije vfat na lokalnem disku so primerne opcije "defaults,owner,umask=000", v zadnjih dveh poljih pa napišemo dve ničli. Pomen nekaterih opcij je naslednji:

defaults: nastavi opcije rw, suid, exec, auto, nouser
owner: dovoli priklop navadnim uporabnikom, ki imajo dovoljenje za dostop do podanega imenika
umask: navedemo zaščito a obratno kor pri chmod, torej 0 pomeni rwx, 1 pomeni rw-, 7 pa ---
rw: particijo priklopimo za branje in pisanje (če želimo samo branje, uporabimo ro)
suid: pri zaščiti se bo upoštevalo lastništvo in grupe (obratno je nosuid, kar ni priporočljivo)
exec: dovolimo izvajanje programov s priklopljene particije (obratno je noexec)
auto: dovoli samodejno določanje vrste datotečnega sistema
nouser: prepovej priklop tistim uporabnikom, ki nimajo dovoljenja za dostop do podanega imenika

Pogosto imamo na računalniku instalirana operacijska sistema MS Windows in Linux in želimo, da bi iz obeh lahko dostopali do istih datotek. Da bi bilo to izvedljivo, moramo uporabiti takšen datotečni sistem, ki ga znata uporabljati oba sistema. Trenutno so na obeh sistemih dobro podprti le datotečni sistem FAT16 oz. VFAT in FAT32. Uporaba NTFS v Linuxu ni zanesljiva, ext2 in ext3 pa v sistemu MS Windows nista podprta.