**Slide 1:**

NTFS in drugi datotečni sistemi so predstavljeni kot naložljivi gonilniki (loadable drivers). Aplikacije jih kličejo posredno.

* Okoljski podsistemi NT kličejo sistemske servise NT, ti pa nato ustrezne gonilnike.
* Gonilniki se povezujejo z:
  + Servisom LFS (Log Service File) (beleženje dogodkov)
  + Upravnikom medpomnilnika
  + Upravnikom navideznega pomnilnika
* Aplikativni programi dostopajo do datotek tako, kot do drugih objektov: preko ročajev na objekte.

**Slide 2:**

Osnova strukturi NTFS je zvezek (volume). Ta ustreza logični particiji na disku in ga dobimo s formatiranjem (dela) diska za NTFS.

* Disk ima lahko enega ali več zvezkov. lahko pa imamo zvezek, ki zaseda več diskov (na primer fault tolerant volume)
* Za NTFS je osnovna alokacijska enota na disku grozd (cluster) in ne sektor. Zato je NTFS neodvisen od fizične velikosti diska.
* NTFS naslavlja fizične lokacije na disku preko takoimenovanih LCN (Logical Cluster Numbers).
* Do podatkov v posameznih datotekah pridemo preko takoimenovanih VCN (Virtual Cluster Numbers), ki se za vsako posamezno datoteko štejejo od 0 naprej. VCN niso nujno zvezni in jih preslikujemo na poljubno število LCN v danem zvezku.

**Slide 3:**

* Srce strukture NTFS v zvezku (na disku) je glavna tabela datotek (MFT, Master File table).
* Vsak tak zapis je dolg 1k, ne glede na velikost grozdov. Za vsako datoteko na disku ima MFT eno vrstico, vključno z vrstico za samo MFT.
* Poleg tabele MFT je na zvezku še množica metadata datotek.
* Vsak datotečni atribut pomnimo kot ločen tok bajtov v datotekah. Strogo rečeno, NTFS ne bere ali piše datotek, on bere ali piše tokove atributov.

**Slide 4:**

* NTFS in FAT dovoljujeta 255 mest dolga imena datotek. Ta imena imajo lahko tudi po več pik in presledkov.
* Imenski prostor POSIX je enak imenskemu prostoru NTFS.
* MS DOS imena datotek pomnimo v istem datotečnem zapisu kot dolga imena datotek

**Slide 5:**

* Vsak atribut začenja zaglavje (header), v katerem so podatki o atributu (ali je rezidenčen, dolžina njegove vrednosti (dolžina podatkov) itd).
* Če vrednost (njegov podatek) nekega atributa pomnjena kar v datotečnem zapisu, je to rezidenčni atribut.
* Če je datoteka majhna, pomnimo vse atribute in tudi njihove vrednosti (podatke) v datotečnem zapisu. Dostop do (zelo) kratkih datotek je zato hiter.
* Majhen direktorij je primer take,kratke datoteke

**Slide 6:**

* Veliko datotek pa ni dovolj majhnih. Za atribute z dolgimi vrednostmi (na primer za datoteke z več podatki) uvede NTFS izven tabele MFT nekj kB velik podaljšek (pravimo mu tudi extent, run), v katerem pomni vrednost atributa (v našem primeru podatke datoteke).
* Atributi, katerih vrednosti so pomnjene izven MFT, so nerezidenčni atributi.

**Slide 7:**

* Če atributov datoteke (ali direktorija) ne moremo shraniti v datotečni zapis MFT, potrebujemo dodatne, ločene alokacije.Pri tem si NTFS pomaga z virtualnimi številkami grozdov (VCN).
* Slika kaže , kako zaglavje podatkovnega atributa pomni preslikavo med virtualnimi in logičnimi številkami grozdov

**Slide 8:**

* Samo tisti atributi, ki lahko rastejo, so lahko nerezidenčni. Pri datotekah so lahko taki atributi opisniki varnosti (security descriptors),podatki, pa tudi seznam atributov.
* Standardna informacija in atribut z imenom datoteke sta vedno le rezidenčna.
* Slika kaže primer večjega direktorija. Le del indeksov datotek je pomnjen v korenskem atributu, preostali pa so pomnjeni v nerezidenčnih podaljških (ki jim pravimo "index buffers".

**Slide 9:**

* Če ima neka datoteka preveč atributov, da bi jih lahko pomnili v MFT v enem datotečnem zapisu, uporabimo dodatni datotečni zapis in uvedemo dodatni atribut "atribute list". V tem seznamu pomnimo ime in tip atributa ter ustrezen naslov datotečnega zapisa v MFT.
* Pri NTFS so datotečni direktoriji kar indeksirana zbirka datotečnih imen in njihovih naslovov.
* Slika prikazuje datotečni zapis korenskega direktorija v nekem zvezku.

**Slide 10:**

* Volume set uporabljamo za konsolidiranje večjega števila neizkoriščenih prostorov na diskih v enoten zvezek. Volume set je en logični zvezek, ki ga sestavlja do 32 področij prostora na enem ali več diskih.
* Z orodjem "Windows NT Disk Administrator" združimo ta področja v enoten volume set. in ga formatiramo za izbrani datotečni sistem.
* Slika kaže tak volume set, ki je označen kot pogon D:

**Slide 11:**

* Stripe set (stripe = proga) je zaporedje več particij (vsaka particija na svojem disku), ki skupaj predstavljajo en logični zvezek. Vse particije morajo biti enako velike.
* Sistem tak zvezek optimizira glede na čase dostopa tako, da podatke primerno porazdeli po več fizičnih diskih. Primer take porazdelitve ponazoruje slika.
* Taka organizacija je primerna predvsem pri sistemih z veliko gostoto dostopov do diskov.

**Slide 12:**

* Stripe set s parnostjo zagotavlja tolerantnost do napak tako, da rezervira ekvivalent enega diska za vsak stripe. Vidimo, da parnost kroži med diski, kar povečuje hitrost prenosa podatkov.
* Ukrepi v primeru okvarjenih grozdov.
* Če sistem odkrije slab sektor, restavrira podatke in slab sektor nadomesti, če se da.

**Slide 13:**

* KO NTFS odkrije slab sektor, preveže celoten grozd, v katerem se tak sektor nahaja.
* V primeru, da se je to zgodilo med branjem na redundančnem zvezku, lahko restavrira podatek, sicer pa le vrne ugotovitev, da je podatek napačen, v bodoče pa tak slab sektor ne bo več klican.