Оперативни системи и рачунарске мреже

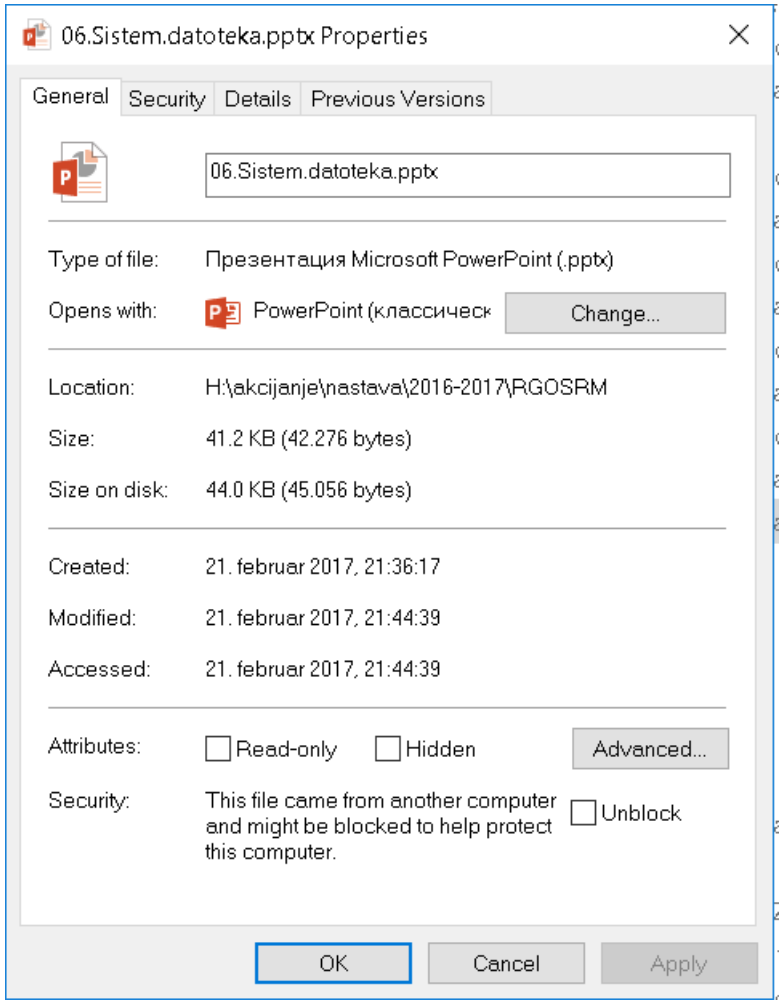
Александар Картељ
aleksandar.kartelj@gmail.com
Рачунарска гимназија

Систем датотека

- Такође се назива и Фајл систем
- Радна меморија није погодна за складиштење података
  - Губи се приликом гашења оперативног система
  - Чак и да се не губи, недовољне је величине
- Секундарне (спољашње) меморије се користе за Систем датотека
  - Најчешће је то Хард-диск
- Централни појам је датотека (или фајл)
  - Представља именовану колекцију информација
  - Може садржати податке (текст, слике, ...), али и програме (бинарни садржај)

Атрибути датотеке

- Име датотеке*
- Локација*
- Величина
- Време креирања
- Време модификације
- Време последњег приступа
- Власник датотеке
- Права приступа
- ...



Операције над датотекама

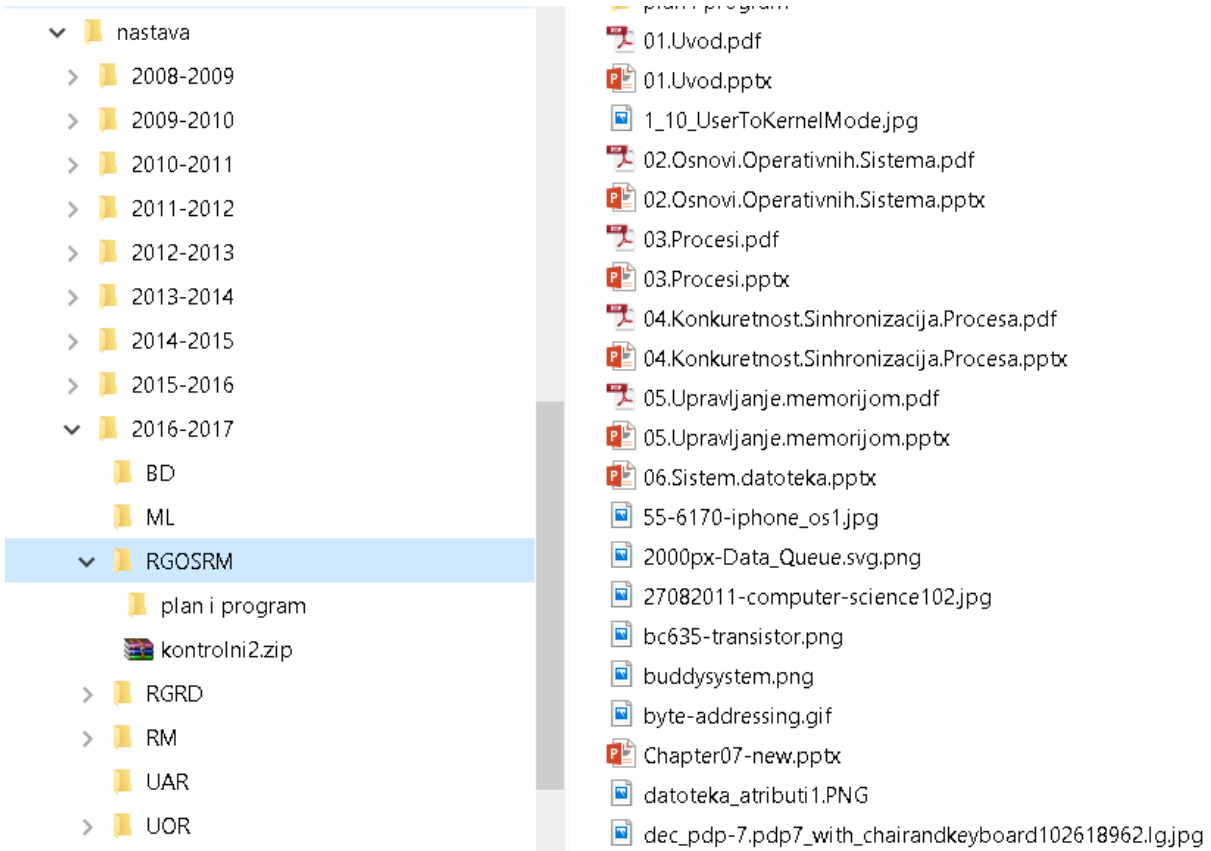
1. Креирање
2. Брисање
3. Читање
   - Отварање датотеке само за читање
4. Писање
   - Измена датотеке, додавањем новог садржаја или изменом постојећег
5. Репозиционирање
   - Померање показивача на одређени део датотеке
6. Скраћивање
   - Брисање крајњег дела датотеке

Типови датотека

- Екстензија је додатак имену датотеке
  - Она помаже оперативном систему да разуме намену датотеке
  - И кориснику потом предложи чиме се датотека отвара
- Нпр. бесмислено је извршну датотеку отварати текст едитором
- Неке стандардне екстензије су:
  - Таговани документи: .html, .htm, .xml
  - Текстуални документи: .txt, .dat
  - Слике: .jpg, .png, .tif
  - Изворни кодови: .c, .cpp, .cs, .java, .asm
  - Извршне датотеке: .exe, .jar, .com, .bin

Директоријуми

- Такође представљају један специјални тип датотека
- Уместо података, у себи садржи списак других датотека
  - А пошто је и сам датотека, то значи да може да садржи и друге директоријуме



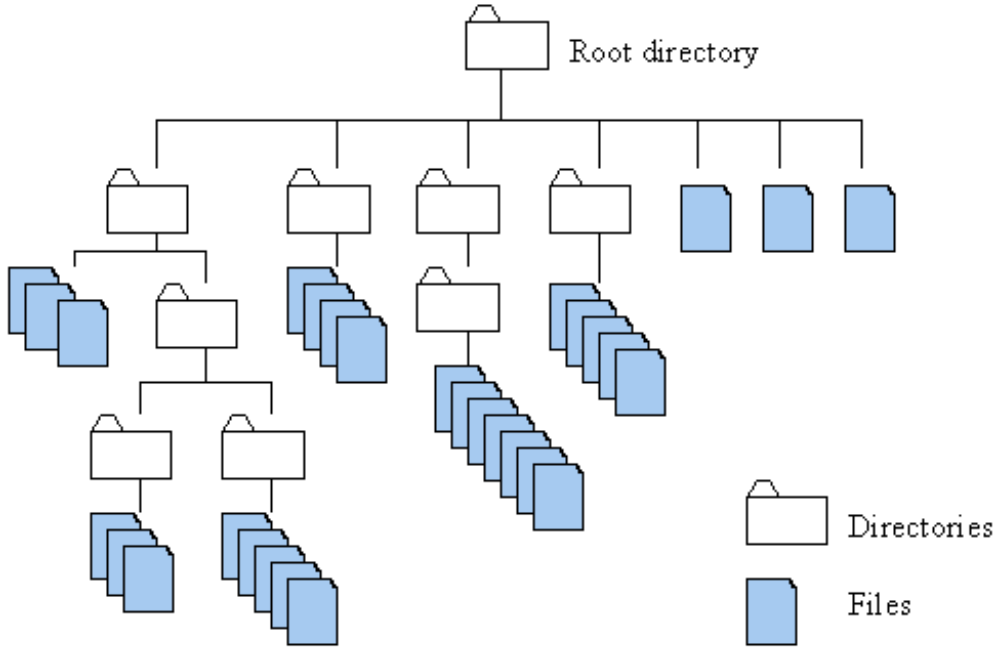
Операције над директоријумима

1. Креирање
   - Иницијално, у списку нема датотека
2. Брисање
   - Обично подразумева рекурзивно брисање свих припадајућих датотека
3. Читање (овде се зове листање)
   - Излиставање свих припадајућих датотека
4. Додавање нове датотеке
5. Брисање припадајуће датотеке

* Додавање и брисање су слични писању у случају обичних датотека

Организација Система датотека

- Општеприхваћена организација је заснована на дрвету
- Свака датотека има јединствену путању од корена
  - Због тога је пуно име са путањом јединствено иако може бити више истоимених датотека
  - Обичне датотеке су увек листови дрвета (крајњи чворови)
- Често се користе и релативне путање:
  - „.“ – референца на самог себе
  - „..“ – референца на претка (наддиректоријум)

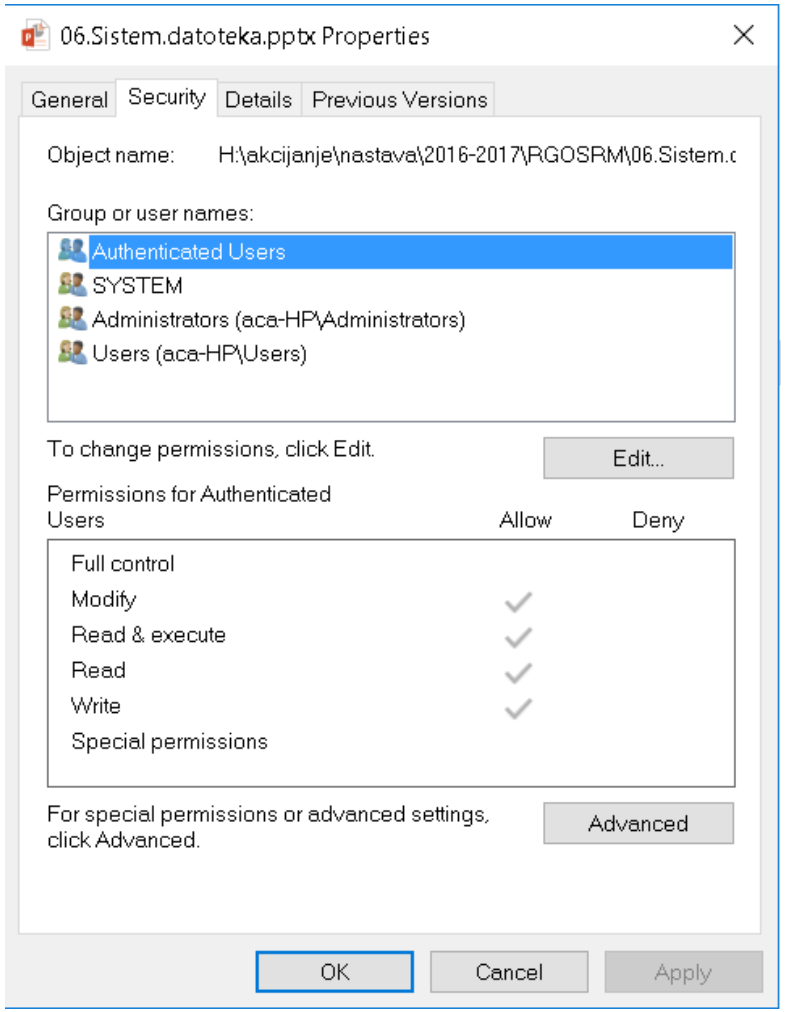


Права приступа

- Код једнокорисничких система релативно једноставно
- Код вишекорисничких
  - Није решење да свако приступа само својим датотека
  - Мора бити флексибилније решено
- Обично се за вишекорисничке користе системи дозвола:
  - Дозвола за читање
  - Дозвола за писање
  - Дозвола за извршавање
- Обично се омогућавају дозволе на нивоу појединачних корисника
- Али и група више корисника ради лакшег управљања

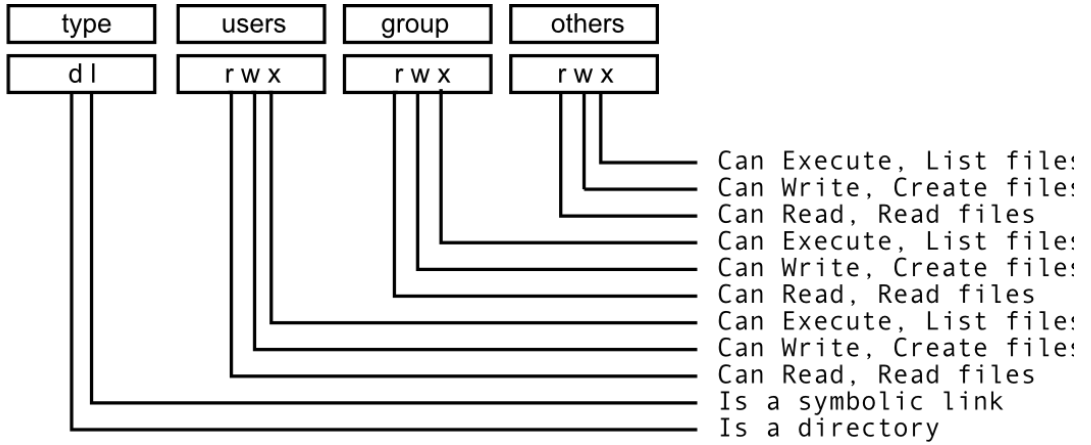
Права приступа (2)

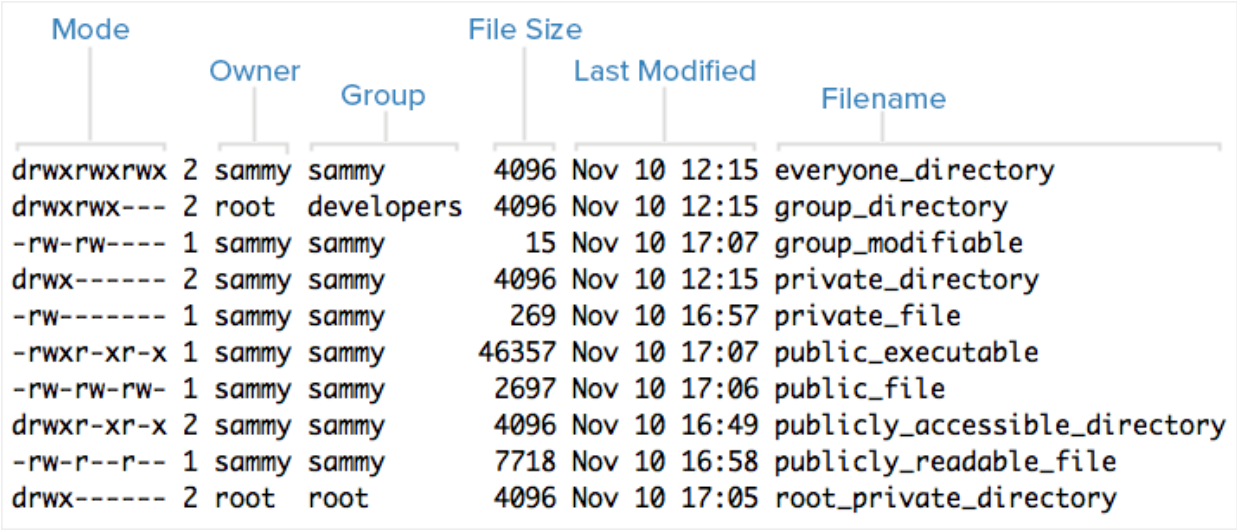
- Различито значење над обичним датотекама и директоријумима
- Код датотека је јасно
- Код директоријума:
  - Читање подразумева право листања
  - Писање подразумева додавање и брисање датотека
  - Извршавање подразумева улаз у директоријум и приступ датотекама



Права приступа (3)

- Под Linux-ом су дозволе базиране на 3 тробитна броја (максимум је 777)
- [3 типа корисника] x [3 типа дозвола]
- Типови корисника су задати:
  - По конкретном кориснику
  - По групи корисника
  - Сви остали
- Нпр. Директоријум **group_directory** може да мења, чита и извршава или корисник **root** или било ко из групе корисника **developers**, а остали корисници немају никакве дозволе.
  Дозвола је, дакле, 770.





Имплементација Система датотека

- Систем датотека се смешта на секундарну меморију
  - Звaћемо је диск (без обзира да ли је у питању HDD, SSD, USB, ...)
  - Дискови су обично подељени на партиције (одвојене логичке просторе)
- Први сектор диска се назива MBR (master boot record)
  - Подаци записани ту се први учитавају при покретању рачунара
  - MBR садржи табелу партиција са почетном и крајњом локацијом сваке
  - Једна од партиција је у MBR означена као активна
  - Први корак у извршавању ОС је проналажење активне партиције и учитавање њеног првог блока (Boot блок)
  - Boot блок садржи процедуру за покретање ОС

Имплементација датотека

- Најмања јединица адресирања у систему датотека се зове **блок**
- Магнетни дискови обично користе блокове од 512B или 4KB
- Свака датотека је представљена скупом придружених блокова
- Кључни проблеми:

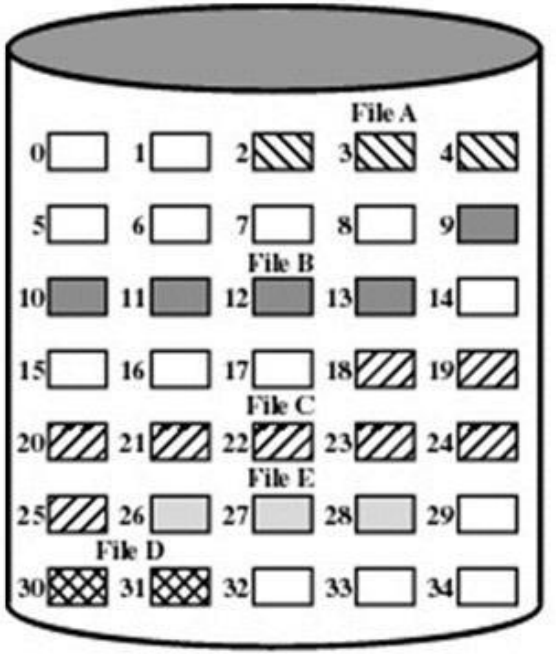
1. *Начин на који се алоцирају слободни блокови*
2. *Памћење блокова који припадају некој датотеци*

Приступ 1 – непрекидна алокација

- Најједноставнији приступ
- Датотеку чини непрекидни низ блокова
- Нпр. датотека има величину од N блокова и почиње на локацији B
  - Онда она заузима блокове: B, B+1, ..., B+N-1
  - Следећа датотека се смешта после ове
- Добре стране:
  - Убрзава читање, јер су датотеке организоване секвенцијално
    - Глава хард-диска се минимално помера при секвенцијалном приступу
    - Директан приступ такође брз, јер се зна унапред где почиње сваки део датетеке, на основу њеног почетка: i-ти блок се налази на B+i локацији

Приступ 1 – непрекидна алокација (2)

- Лоше стране:
  - Приликом брисања се прави рупа → долази до екстерне фрагментације
  - Пошто су блокови фиксне величине → долази и до интерне фрагментације
  - Највећи проблем: раст датотеке!
    - Или оставити довољно простора на крају → још већа интерна фрагментација
    - Или премештање → кошта, јер је хард-диск спор



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

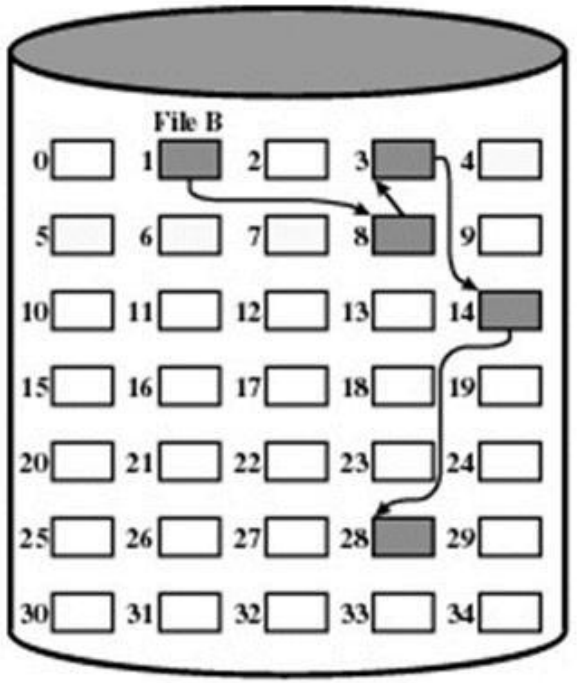
Figure 12.7 Contiguous File Allocation

Приступ 2 – преко повезане листе

- Датотека представљена као повезана листа блокова
- Блок садржи показивач на следећи блок, тј. адресу следећег блока
- Директоријум за сваку датотеку садржи само адресу првог блока
- Нова датотека се креира тако што се у директоријум дода нови показивач на први блок те датотеке
- Добре стране:
  - Нови блок може бити било где, а не на узастопним локацијама
  - Блокови датотеке разбацани свуда по диску
  - Дакле, нема екстерне фрагментације

Приступ 2 – преко повезане листе (2)

- Лоше стране:
  - Јако неефикасан директан приступ неком делу датотеке
  - Да би се нашао i-ти блок, морају се проћи свих i-1 блокова
  - Додатно успорење, јер померања главе хард-диска кошта
  - Лоша поузданост – шта ако се деси грешка на неком показивачу



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	1	5
...	...	...

Figure 12.9 Chained Allocation

Приступ 3 – индексирана алокација

- Проблем код претходног приступа је што су показивачи разбацани
- Индексирана алокација држи све показиваче на једном месту, тзв. блоку индекса
- Свака датотека садржи свој блок индекса, који садржи низ адреса блокова који чине датотеку
- Ако треба да се прочита i-ти блок датотеке, узима се i-ту адресу
- Добре стране:
  - Нема екстерне фрагментација, као и код повезаних листи
- Лоше стране:
  - Доста простора се троши на показиваче, чак иако је мала датотека

Приступ 3 – индексирана алокација (2)

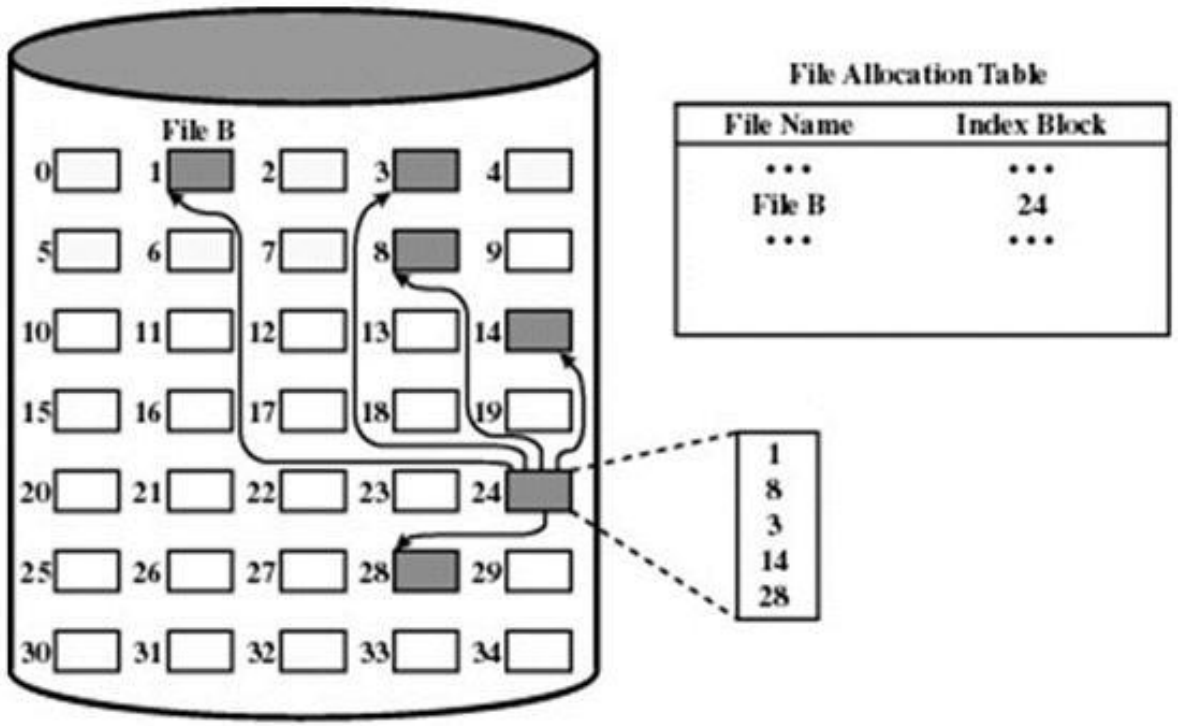


Figure 12.11 Indexed Allocation with Block Portions

Имплементација директоријума

- Да би ОС нашао нпр. датотеку C:/user1/doc/text/dat.txt:
  - Прво мора у директоријуму C:/ потражити директоријум **user1**
  - Потом га отворити, и у њему наћи директоријум **doc**
  - Потом и њега отворити, и у њему наћи директоријум **text**
  - Потом и њега отворити, и учитати блокове који се односе на **dat.txt**

Приступ 1 – преко листе

- Директоријум представља као повезану листу имена датотека са придруженим показивачима на почетне блокове датотека
- При креирању нове датотеке, претражује се листа како би се утврдило да већ не постоји нека са истим именом
- Лоше стране:
  - Брисање или тражење датотеке захтева пролазак кроз целу листу имена
  - Побољшање:
    - Листу имена одржавати сортираном
    - Онда се може применити бинарна претрага зарад бржег проналажења

Приступ 2 – преко хеш табеле

- Уместо у повезану листу, имена датотека убачена у хеш табелу
- Ово омогућава проналажење датотеке скоро моментално
- Потребно је користити одговарајућу хеш функцију:
  - Нпр. збир ASCII вредности карактера из назива датотеке по модулу N
  - Где је N максимална величина хеш табеле
  - Ово ће сваки назив пресликати на одговарајућу позицију у низу
  - Нпр. Ако се датотека зове test.txt, а N=100, датотека ће се сместити на:
    - 4 x ascii(t)+ ascii(e) + ascii(s) + ascii(x) + ascii(.) = 4 x 116+101+115+46+120 = 846
    - 846 % 100 = 46
    - Дакле, ова реч би се ставила на позицију 46
  - Шта може бити потенцијални проблем?

Манипулисање слободним простором

- ОС мора да води евиденцију о **списку** слободних блокова
- При креирању датотеке, из списка се елиминишу узети блокови
- При брисању датотеке, адресе блокова се уписују у списак
- Први приступ користи бит-мапу:
  - Сваки блок се представља 0/1 ако је заузет/слободан
  - Нпр. ако су заузети блокови 3, 5, 7, 11, 12, онда је мапа 0010101000110 0...
  - Проблем је што за велики диск и бит-мапа мора бити велика
- Други приступ користи повезану листу:
  - Када се ослободи неки блок, само додамо показивач ка њему на крају листе
  - Овде није проблем директан приступ,
    јер када додељујемо блок, није битно који додељујемо