

ЛЕКЦИЯ 9

ФАЙЛОВА СИСТЕМА НА ОС

 **Определение за файл**

 **Операции с файлове**

 **Функции на ФС**

 **Файлова организация**

 **Разпределение на ВП**

 **Описател на файл**

ОПРЕДЕЛЕНИЯ

Файл се нарича **организирана съвкупност** от **данни**, на която е дадено **име**. Файловете **обикновено** са **във външна памет (ВП)**.

Операциите с файлове (**като цяло**) **са**:

- ① **откриване** (**open**) – **подготовка** за работа;
- ② **закриване** (**close**) – **край** на работата;
- ③ **създаване** (**create**) – **формиране** на нов файл;
- ④ **унищожаване** (**destroy**) – **разрушаване** на файла;
- ⑤ **копиране** (**copy**) – създаване на **нов екземпляр**;
- ⑥ **преименуване** (**rename**) – **смяна на името**;
- ⑦ **показване** (**list**) – **извеждане на екран** (**печат**).

ОПЕРАЦИИ СЪС ЗАПИСИ

Данните в един файл **се** разделят **на** индивидуални **елементи**, които могат да се поберат в ОП. **Обработката** е **на равнище елемент (запис)**. **Операциите с елементи** са:

- ❶ **четене** (**read**) – **вход** на елемент **от файл** в процес;
- ❷ **запис** (**write**) – **изход** на елемент **във файл**;
- ❸ **актуализация** (**update**) – **модифициране (замяна)** на **съществуващ** във файла **елемент** с данни;
- ❹ **вмъкване** (**insert**) – **добавяне** на нов елемент;
- ❺ **изтриване** (**delete**) – **изключване** на елемент.

Файловата система отговаря за управлението на файловете, разположени във ВП.

ФУНКЦИИ НА ФС НА ОС

ФС реализира **следните функции** на ОС:

- ❶ предоставя на потребителите **възможност да създават, модифицират и унищожават файлове**;
- ❷ предоставя на потребителите **възможност да разделят под контрол файловете помежду си**;
- ❸ **механизмът за разделяне** на файл (колективно използване) трябва да предвижда **различен вид контролиран достъп**: четене, запис, изпълнение;
- ❹ предоставя на потребителите **възможност за задаване на удобна структура** на файловете;
- ❺ предоставя на потребителите **възможност да управляват предаването на данни**;

ФУНКЦИИ НА ФС (прод.)

ФС реализира **още** и следните **функции**:

- ⑥ трябва да предвижда **средства за съхраняване и възстановяване** на файловете;
- ⑦ предоставя на потребителите **възможност за обръщение** към файл **чрез символично име**, а не **чрез име на физ. у-во** (независимост от ПУ);
- ⑧ в системи със секретна информация трябва да предоставя **възможности за шифриране и дешифриране на данните** във файловете;
- ⑨ трябва да има **дружелюбно отношение към потребителя**: той да **работи с логическо**, а не с **физическо представяне** на данни и устройства.

ЙЕРАРХИЯ НА ДАННИТЕ

Всички данни се формират **от битове** (**0** и **1**).

Байт – група **битове**, кодиращи един **знак**.

[**Полубайт**: ляв – **зонов** и десен – **цифров**.]

Поле – група **свързани байтове**: **буквено**, **цифрово**, **буквено-цифрово** и **смесено**.

Запис (**логически**) – група **свързани полета**.

Файл – група **взаимосвързани записи**.

Файловете днес биват **двоични** (**всяко поле** е **двоично число**) и **текстови** (**всеки байт** се третира като **определен знак**).

ВИДОВЕ ЗАПИСИ

Физически запис (блок): единица данни, която **реално се обменя с дадено ПУ.**

Логически запис: съвкупност от данни, която потребителят **възприема като единно цяло.**

Между двата вида записи **няма връзка.**

При съответствие в 1 физически 1 логически се говори за **неблокувани записи.**

При съответствие в 1 физически няколко логически се говори за **блокувани записи.**

Записите могат да бъдат: с фиксирана, с променлива и с неопределена дължина.

ПРИМЕР: ВИДОВЕ ЗАПИСИ

фиксирана дължина, неблокувани



АМ ключ данни

фиксирана дължина, блокувани



АМ ключ данни

променлива дължина, неблокувани



АМ ключ данни

променлива дължина, блокувани



АМ ключ данни
ос-09

8/41

БУФЕРИРАНЕ НА ЗАПИСИ

Буферирането позволява **изчисленията** да се извършват **едновременно с В/И** операции.

Буфер е място в ОП, в което може да се разположат **един или няколко** физически или логически (при блокиране) **записа**.

Най-разпространена е схемата с **двойна буферизация**: докато **в единия** буфер се реализира **В/И** на данни с ПУ, **ЦП** може да **работи с** данните в **другия** буфер. **Буферите се сменят след** приключване на **обработката**.

ОРГАНИЗАЦИЯ НА ФАЙЛ

Под **организация на файл** се разбира **начинът за разполагане на записите на файла във ВП**. Бива **четири основни вида**:

- ① последователна;
- ② индексно-последователна;
- ③ пряка;
- ④ библиотечна.

Не всички ЗУ дават възможност за **удобно реализиране на всяка организация**. Затова за **ЗУ** се говори като за **последователни (ленти)** и за **преки (дискове) устройства**.





ХАРАКТЕРИСТИКИ НА ФАЙЛ

Файловете имат (и ФС трябва да ги поддържа) **следните важни характеристики:**

- ① изменяемост:** честота на включване на нов запис или промяна на съществуващ. При малка файлът се нарича статичен, а при голяма – динамичен или изменяем;
- ② активност:** % от записите на файла, които се обработват при дадено изпълнение;
- ③ размер:** количество на данните, които се съхраняват в този файл.

ФАЙЛОВА СИСТЕМА

Файловата система е важна част от ОС. Като правило тя **съдържа следните средства**:

-  **методи за достъп**, определящи конкретната организация на достъп до данните във файла;
-  **средства за управление на файловете**, които осигуряват съхранение на, обръщение към, колективно използване и защита на данните;
-  **средства за управление на ВП**, осигуряващи разпределението на ВП за запис на файлове;
-  **средства за осигуряване на цялостност на файловете**: гарантира съхранимост на файла, т. е. това, което трябва, е там и друго няма.

СПРАВОЧНИЦИ

За да реализира своите функции по разполагане на файловете във ВП и за осигуряване на достъпа до съхранените в тях данни, **ФС се нуждае от работно място на всеки носител**, където да си води бележки.

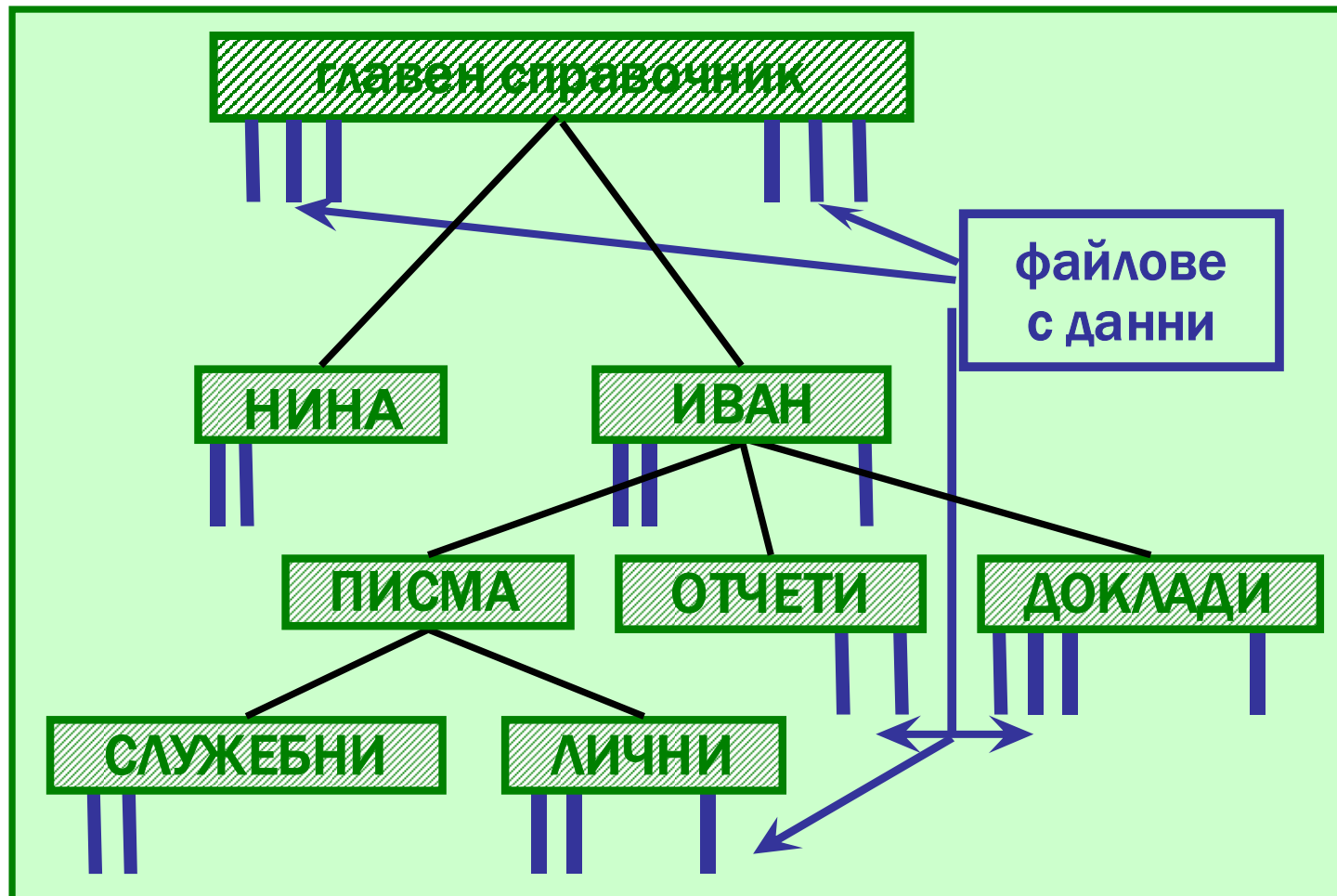
Това място е **известно като справочник на файловете** за този носител – **таблица на ОС.**

В справочника се съхраняват **имената** на файловете, **сведения за достъпа** до данните, **защитата на файла** и др. **Изгодна стратегия е справочниците да бъдат йерархични.**

ЙЕРАРХИЧНИ СПРАВОЧНИЦИ

- 💡 **Голямо** количество файлове в носител.
- 💡 **Трудности при** (едновременно) работа на повече от един човек.
- 💡 **Имитация на нормалната човешка организация**, използвана **за борба със сложността**: листовете ⇒ пликите ⇒ папки ⇒ чекмеджета ...
- 💡 **Пълно име** на файл.
- 💡 **Текущи** справочник и диск.

ПРИМЕР



ОС-09

15/41

РАЗПРЕДЕЛЕНИЕ НА ВП

Проблемите по раздаване и освобождаване на ВП за файлове са подобни на проблемите по разпределяне на ОП при много програми.

Разликата е, че програмите не могат да се местят в ОП, но данните във ВП могат.

Разпределението на ВП може да бъде свързано и разпокъсано. При първото (доста по-просто) с течение на времето дисковото пространство се фрагментира. Събирането на боклука се реализира в неработно време за да не се пречи на потребителите.

СВЪРЗАНО РАЗПРЕДЕЛЕНИЕ

За всеки файл се отделя непрекъсната област от дискова памет (изисква линеаризация на диска).

Нов файл не може да се създава, когато във ВП няма достатъчно голям свободен участък.

Добра страна е, че последователните логически записи обикновено се реализират във ВП чрез последователни физически записи.

Справочникът е изключително прост: име, начален дисков адрес и размер на файла.

Недостатъците са подобни на променливия брой раздели при многопрограмен режим. При промяна на размера такива схема не е рационална.

СПИСЪК ОТ СЕКТОРИ

Дисковата памет се разделя на **физ. записи** с еднаква дължина, наречени **сектори**.

Всеки **файл** се реализира като **списък**.

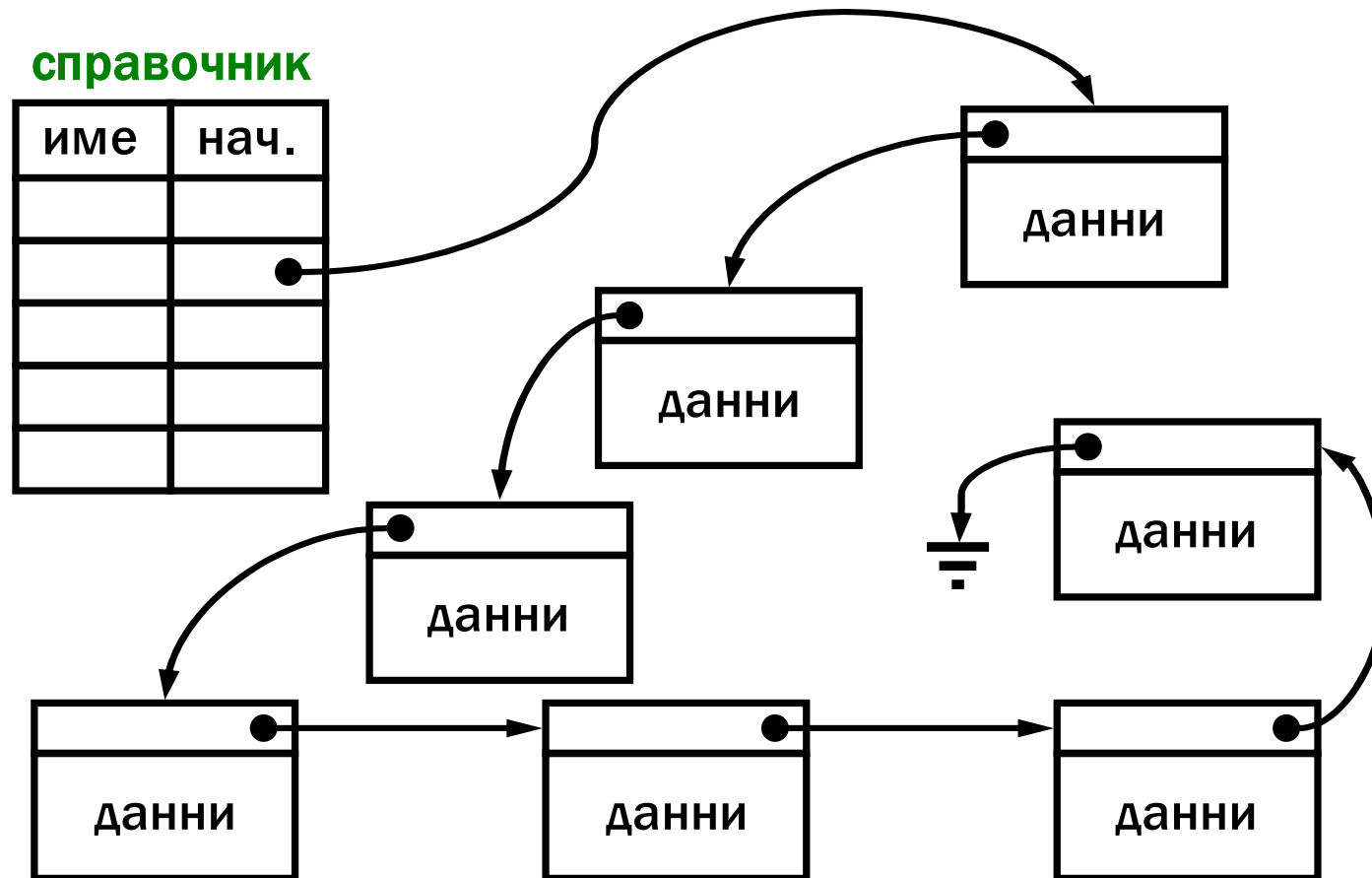
Свободното (и повреденото) дисково пространство може да **се реализира** като **отделен списък** или като **битова карта**.

В справочника се посочва **първия сектор**.

Наличието на указатели **намалява полезния размер на диска** и **води до объркване**.

Достъпът до данните е последователен.

ПРИМЕР: СПИСЪК ОТ СЕКТОРИ



ОС-09

19/41

ХАРАКТЕРНИ ЧЕРТИ

- ☺ Дискът е разделен на сектори (256, 512) с последователни номера, които се трансформират в адреси «ц, п, з» от ФС.
- ☺ промяната на размера на файл е проста;
- ☺ вмъкването и изтриването на данни в средата на файл, също е просто;
- ☺ не е необходимо уплътняване на диска.
- ☹ голяма част от диска се губи за указатели;
- ☹ смесването на указатели и данни в един сектор може да доведе до грешки;
- ☹ достигането до желания сектор е бавно.

СВОБОДНА ПАМЕТ

Свободните (все още не използвани) **сектори** могат да бъдат **организирани като списък на свободните сектори** (като специален файл).

Когато за съществуващ файл **е необходим нов сектор** се използва **първия сектор от списъка на свободните сектори**.

Когато съществуващ файл **освобождава сектор**, то **този сектор се включва в списъка на свободните**. **По-лесно е** това да става **в началото**, но **по-добро решение е** включване **в края**, т. е. да се реализира **кръгов списък**.

БЛОКОВИ РАЗПРЕДЕЛЕНИЯ

Блоковото разпределение съчетава свързаното и несвързаното разпределение.

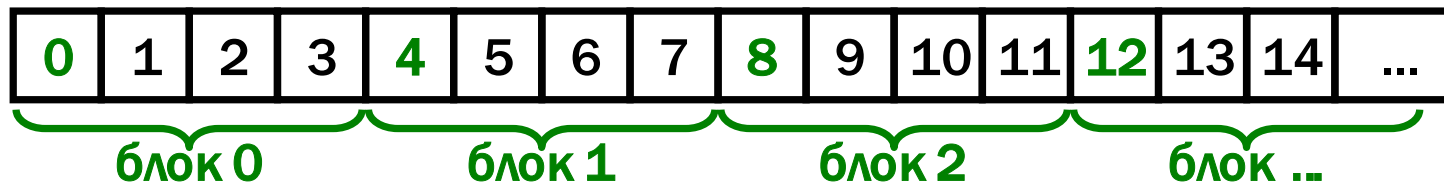
ВП се дели на блокове, като всеки съдържа последователни сектори (екстенти).

Всички блокове могат да бъдат с равен брой сектори, но секторите в различните блокове може и да варират – по-сложна схема.

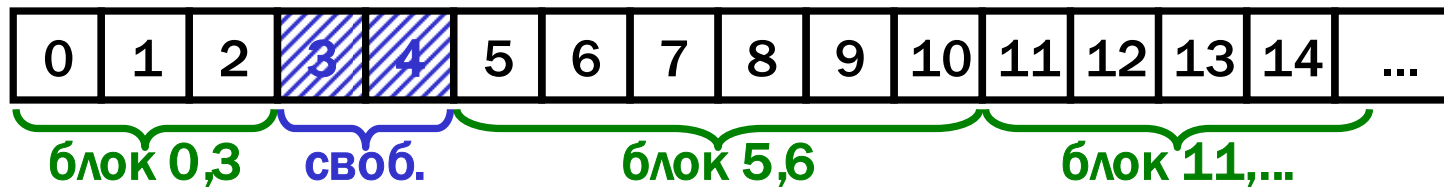
Блоковите разпределения могат да бъдат: списък от блокове, списък от индексни блокове и таблица за разположението.

ПРИМЕР: БЛОКОВЕ

диск, разделен на сектори



същият диск, разделен на **блокове с фиксиран размер** от 4 сектора: за блок с номер n не се посочва размер (фиксиран) и начален сектор



разделяне на **блокове с различен размер**:
всеки блок **се посочва** чрез **начало и размер**

СПИСЪК ОТ БЛОКОВЕ

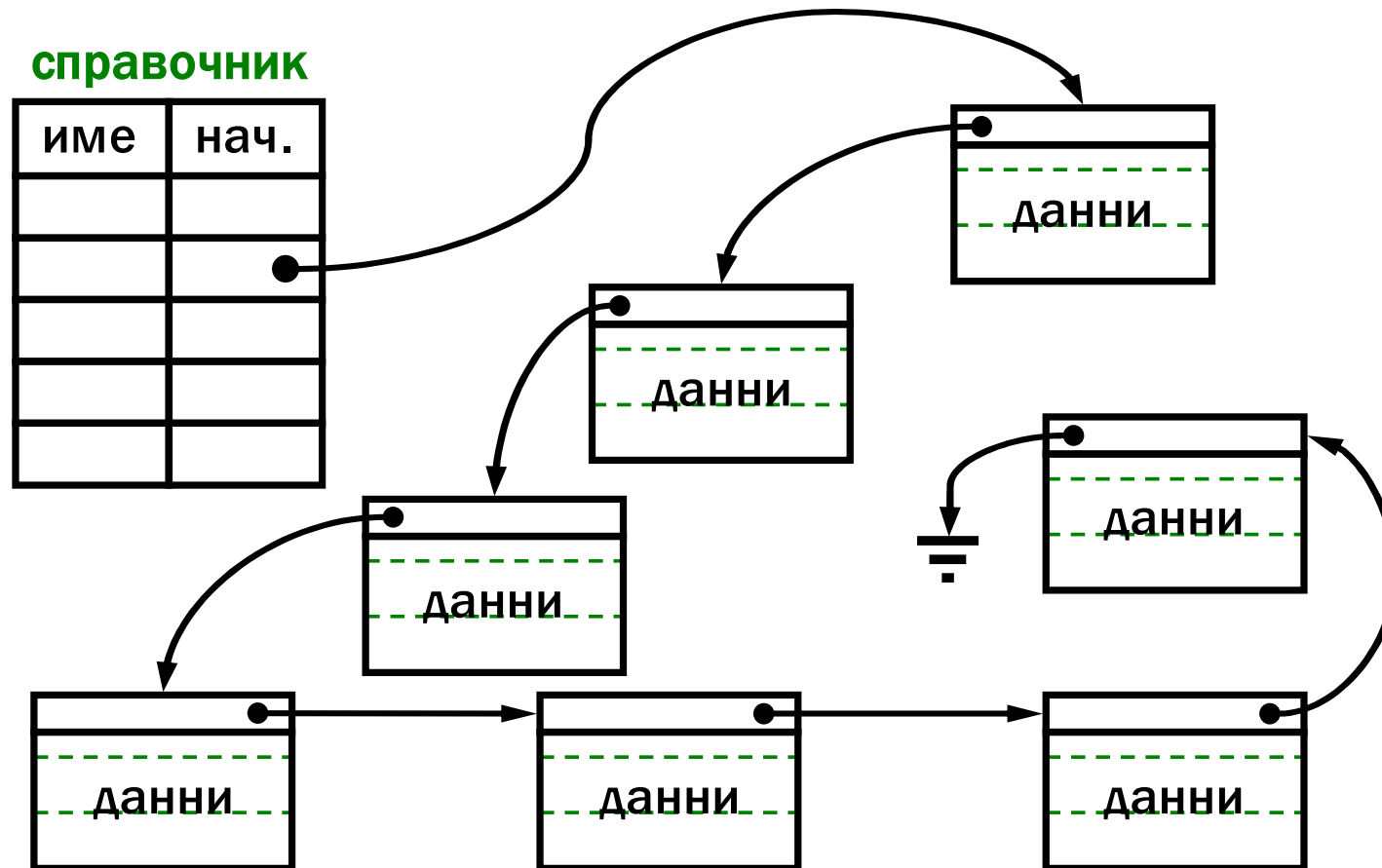
В справочника се посочва първият блок на файла и всеки блок (без последния) съдържа указател към своя наследник.

Добро решение е блокът да обединява секторите в една писта от диска.

Достигането до определен блок от даден файл може да доведе до голямо местене на главите, когато блоковете на един файл са разпръснати по диска.

При блокове с еднакъв размер има малко предимство пред списък от сектори.

ПРИМЕР: СПИСЪК ОТ БЛОКОВЕ



ОС-09

25/41

СРАВНЕНИЕ СЪС СПИСЪК ОТ СЕКТОРИ

При блокове с фиксиран размер тази организация наподобява списък от сектори, но със следните предимства:

- ① размерът на указателя намалява (поне с 1 бит);
- ② указатели има само в първите сектори на блока (повече място за данни);
- ③ при търсене на необходимия блок може да се чете само първият сектор на блоковете.

При блокове с променлив размер:

- 😊 потенциално по-бързо намиране на данните.
- 😞 значително по-сложна организация.

ИНДЕКСНИ БЛОКОВЕ

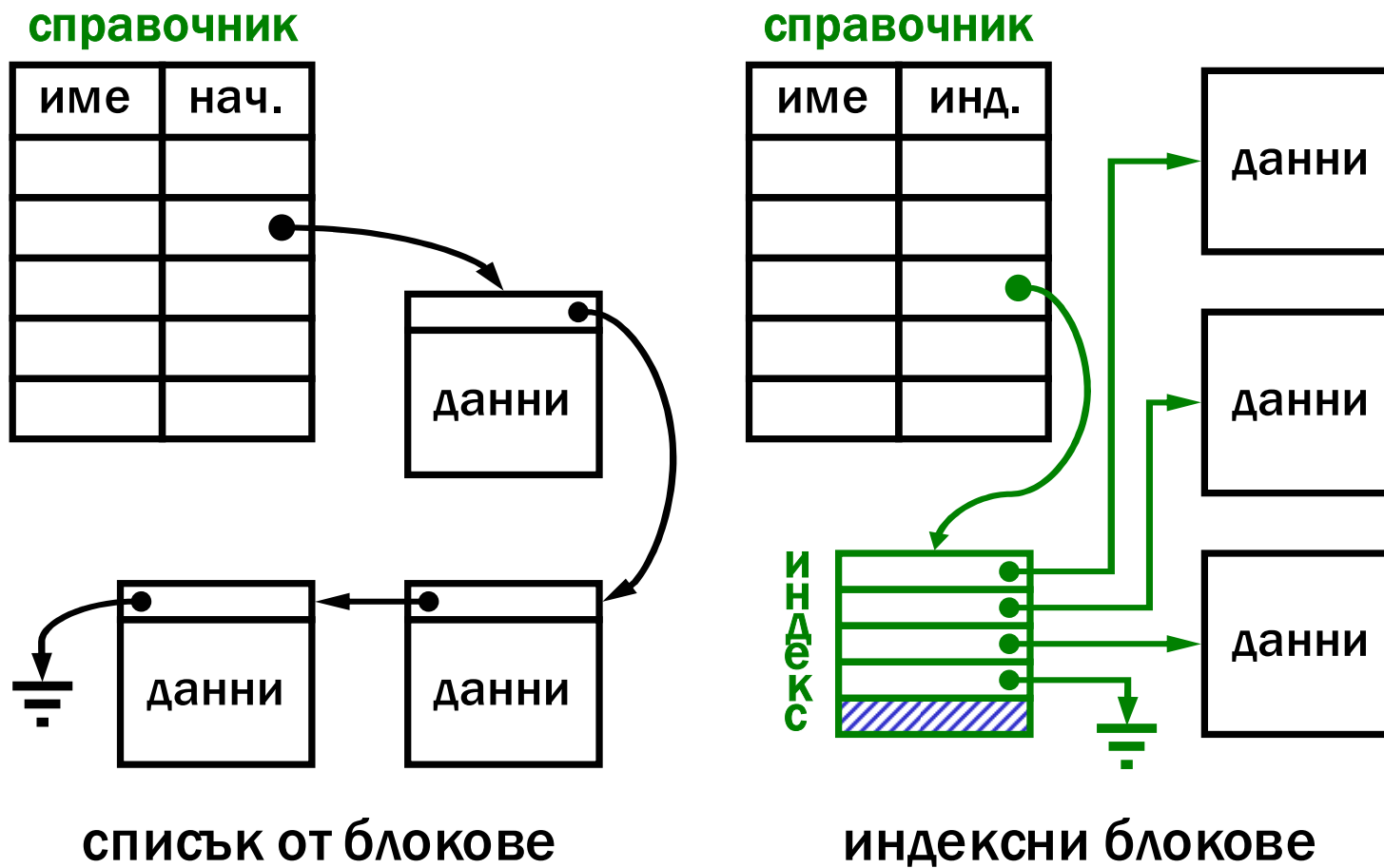
По-добро блоково решение е указателите да се преместят от блоковете с данни в отделен блок, наречен индексен блок.

Така се елиминира разнородността на битовете в рамките на един блок.

Индексният блок за достъп до данните на един файл може да бъде постоянно в ОП. Наличието на индексни блокове ускорява достъпа до данните блокове на файла.

Интересно решение има в ОС UNIX.

ПРИМЕР: ИНДЕКСЕН БЛОК



OC-09

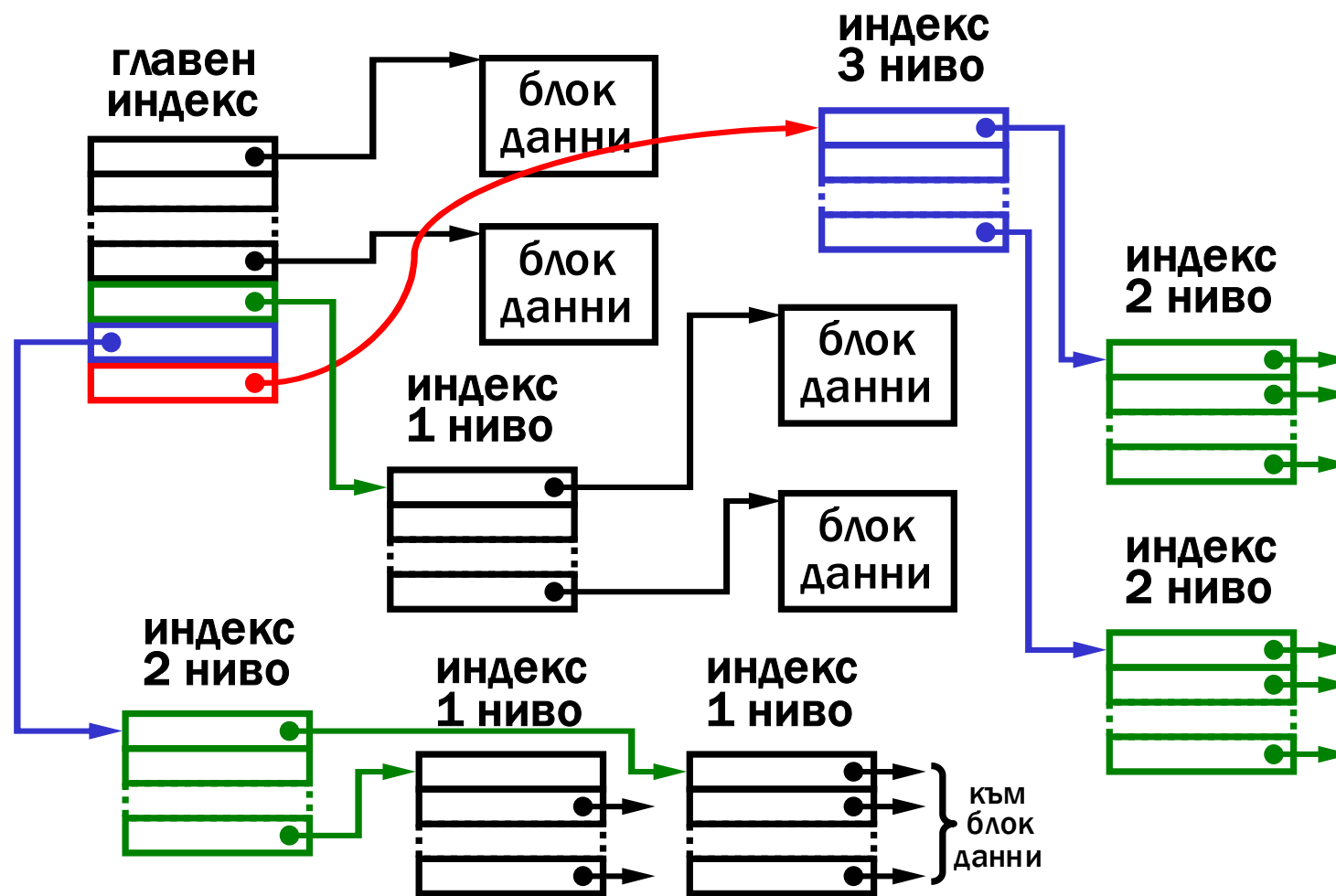
28/41

СРАВНЕНИЕ

В сравнение **със списък от блокове**:

- ☺ **еднородна информация** в 2 вида блокове;
- ☺ **индексният блок** може да се държи **в ОП**;
- ☺ **достигането до** желаните **данни е бързо**;
- ☺ **лесно** се прилага **при променлива дължина**.
- ☹ **ограничава размера** на файла;
- ☹ **вмъкването и изтриването** на данни **в средата на файл е малко по-сложно**;
- ☹ **не е подходящо свободната памет** **да се организира като отделен файл**.

ИНДЕКСИТЕ НА UNIX



ОС-09

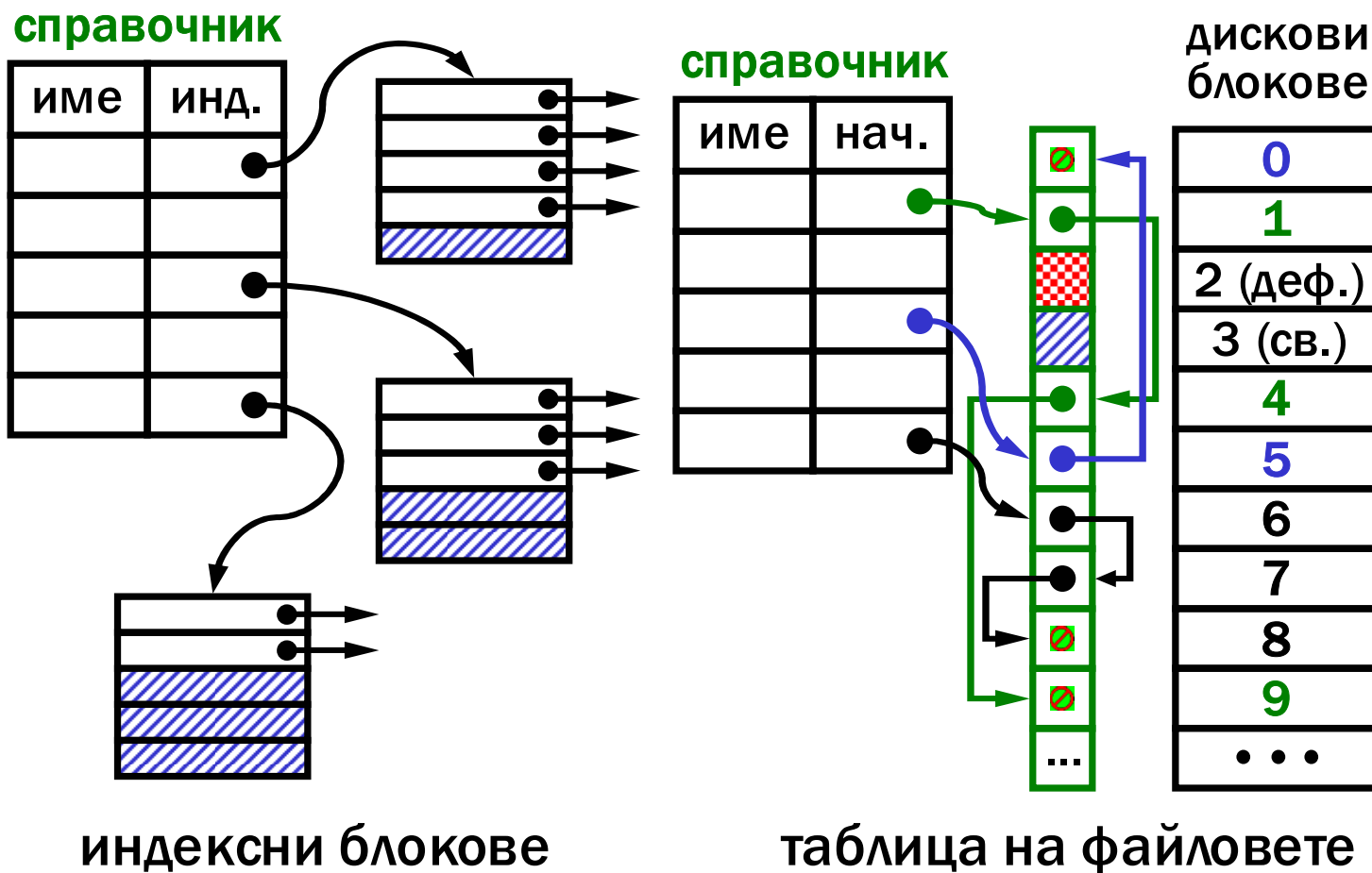
30/41

ТАБЛИЦА НА ФАЙЛОВЕТЕ

Индексните блокове на всички файлове се обединяват в една **обща таблица** за диска. Въвеждането на тази таблица в ОП **ускорява достъпа до всички файлове.**

Смесването на индексните блокове затруднява идентификацията им. Таблицата **повтаря на равнище индекс списъка на блоковете: т. е. всеки** **неин елемент отговаря за съответния блок с данни в диска** и съдържа **индекса на следващия го блок за същия файл** или сигнал, че няма такъв.

ПРИМЕР: ТАБЛИЦА



ос-09

32/41

СВОБОДНА ДИСКОВА ПАМЕТ

При **списък от блокове с еднаква дължина** свободното пространство може да бъде **специален системен файл**. По-добре е той да бъде организиран като **кръгов списък**.

При **индексни блокове с еднаква дължина** свободната памет **също** се организира като **обикновен или кръгов списък**.

При **индексни блокове с различна дължина** е по-добре да има **битова карта на секторите**.

При **таблица на файловете** свободните блокове имат **особен номер в таблицата**.

ОПАСНИ МОМЕНТИ

При списък от сектори или блокове загубата на указател унищожава остатъка от файла.












При индексни блокове повредата на блок за данни е неприятна, но повредата на индексен блок е фатална – файлът се губи.

При таблица за разположение на файловете повредата на таблицата е изключително опасна, защото се унищожават всички данни.

При всяка организация грешки при водене на свободната памет могат да доведат до поява на изгубени сектори/блокове.

ОПИСАТЕЛ НА ФАЙЛ

Описател (дескриптор) на файл или **блок за управление** на файл, е управляващ блок, **съдържащ информация, необходима на ОС за извършване на различните операции с този файл. Във ВП съдържа:**

-  **символично име** на файла;
-  **разположение** на файла **във ВП**;
-  **организация** на файла (**последователна, пряка, ...**);
-  **тип на устройството**;
-  данни за **управление на достъпа** (**права** за достъп);
-  **тип** на файла (**текст**, обектен **код**, **програма** на ... и др.);
-  **характер** на файла (**постоянен, временен, работен**);
-  дата и време на **създаване**;
-  дата за **унищожаване**;
-  дата и време на **последната модификация**;
-  **активност** (**брой** на **четенията, записванията** и др.).

ПРАВА ЗА ДОСТЪП

Различните потребители имат различни права за достъп до данните на файловете.

ФС е отговорна за строгото съблюдаване на потребителските права като забранява несанкционирания достъп.

Правата за достъп бяха разгледани при сегментната организация на виртуалната памет. Те биват: четене, запис (изменение, вкл. унищожаване), изпълнение (когато съдържа машинен код) и разширяване (включване на нови записи във файла).

МАТРИЦА НА ПРАВАТА

Един от възможните начини за управляване на правата за достъп до даден файл е поддържането на матрица на достъпа.

В нейните редове са изброени файловете, а в колоните – потребителите.

Всеки елемент на матрицата е 1 (да) или 0.

Поради различните видове права за достъп матриците трябва да бъдат толкова, колкото видове права поддържа системата (една за четене, втора за запис и т. н.).

ПРИМЕР: МАТРИЦА С ПРАВА ЗА ДОСТЪП

		файлове							
		1	2	3	4	5	6	7	8
потребители	I	да	не	не	не	да	да	не	не
	II	не	не	не	не	да	да	да	да
	III	да	да	да	да	да	да	да	да
	IV	не	не	да	не	не	да	не	не
	V	не	не	не	не	не	да	не	не
	VI	не	да	да	не	не	да	да	не

Определяне на **правото за четене** на файл

ос-09

38/41

ПРАВА ЧРЕЗ КЛАСОВЕ

Матрицата обикновено е **толкова голяма**,
че практически **не** може да **се реализира**.

Поради това **достъпът** до файловете често **се**
основава на класифициране на различните
потребители. **Обичайните класове са:**

- 🕯 **собственик** – като правило **създателят** на файла;
- 🕯 **приятел** – **собственикът** сам **определя** този клас;
- 🕯 **група** – потребители, които **работят съвместно**;
- 🕯 **всички останали** потребители – **общ** достъп.

Разпределението на потребителите **по групи**
(проекти) е дело **на администратора на ОС**.

БЕЗОПАСНОСТ НА ДАННИТЕ




Животът е пълен с изненади!

За предотвратяване на **някои неприятности (пожар)** съществуват **физически средства**.

Най-разпространеният начин за **защита на данните** е тяхното **копиране**. Освен това може **действията** с един **файл** да **се регистрират в друг файл** – скъпа, но оправдана излишна дейност.

Абсолютна безопасност не съществува!

Недостатъци на периодичното копиране са:

-  **закриване на системата** за потребителите;
-  **продължително време** при голям размер на ФС;
-  **при отказ** се губят действията след последното копиране.

Стъпковото копиране често е по-добро решение.

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В
СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ЗАЩИТАТА НА ОС**