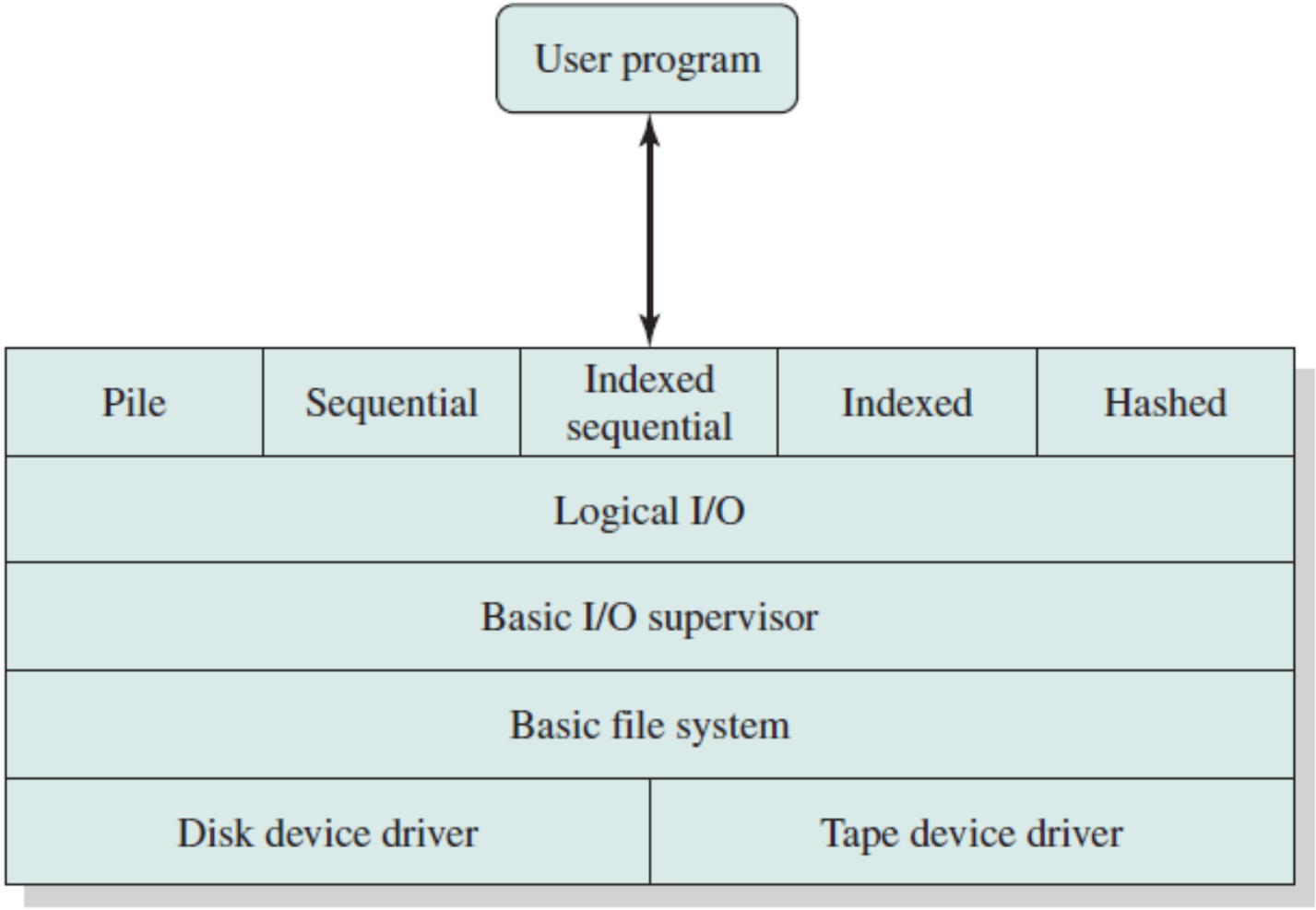
Файлова система

доц. д-р инж. Христо Вълчанов

http://cs.tu-varna.bg

Файлова система - архитектура



Файлове – основни концепции

- **Файл** - логическа последователност от байтове.
- Съдържание:
  - Данни;
  - Програми.
- Структура в зависимост от типа:
  - Текстов файл;
  - Сорс файл;
  - Изпълним файл.

Атрибут на файл

- Name
- Identifier
- Type
- Location
- Size
- Protection
- Time, date, user identification

Съхраняват се в директорна структура.

Операции върху файл

- Create
- Write
- Read
- Reposition within file
- Delete
- Truncate

Пример: операции върху файл

```
/* File copy program. Error checking and reporting is minimal. */

#include <sys/types.h>                  /* include necessary header files */
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[]);       /* ANSI prototype */

#define BUF_SIZE 4096                   /* use a buffer size of 4096 bytes */
#define OUTPUT_MODE 0700                /* protection bits for output file */

int main(int argc, char *argv[])
{
        int in_fd, out_fd, rd_count, wt_count;
        char buffer[BUF_SIZE];

        if (argc != 3) exit(1);         /* syntax error if argc is not 3 */

        /* Open the input file and create the output file */
        in_fd = open(argv[1], O_RDONLY);        /* open the source file */
        if (in_fd < 0) exit(2);                 /* if it cannot be opened, exit */
        out_fd = creat(argv[2], OUTPUT_MODE);   /* create the destination file */
        if (out_fd < 0) exit(3);                /* if it cannot be created, exit */
```

Пример: операции върху файл (2)

```
    /* Copy loop */
    while (TRUE) {
        rd_count = read(in_fd, buffer, BUF_SIZE); /* read a block of data */
    if (rd_count <= 0) break;                     /* if end of file or error, exit loop */
        wt_count = write(out_fd, buffer, rd_count); /* write data */
        if (wt_count <= 0) exit(4);               /* wt_count <= 0 is an error */
    }

    /* Close the files */
    close(in_fd);
    close(out_fd);
    if (rd_count == 0)                            /* no error on last read */
        exit(0);
    else
        exit(5);                                  /* error on last read */
}
```

Отваряне на файл

- **Таблица на отворени файлове** – съдържа информация за всички отворени файлове

```
Open   . . .   Close
```

Реализация в Unix:

- Системна таблица – местоположение, дати, размер и др.
- Таблица за процес – всички файлове, отворени от процеса

Отваряне на файл (2)

- **Файлов указател** – указател към последната позиция за четене/запис за процес, който е отворил файла
- **Брой отворени файлове** – брояч на броя пъти отваряне на файл (позволява премахване на данни от таблицата при затварянето й от последния процес)
- **Местоположение на файла на диска**
- **Права на достъп** – права на достъп до файла за процес

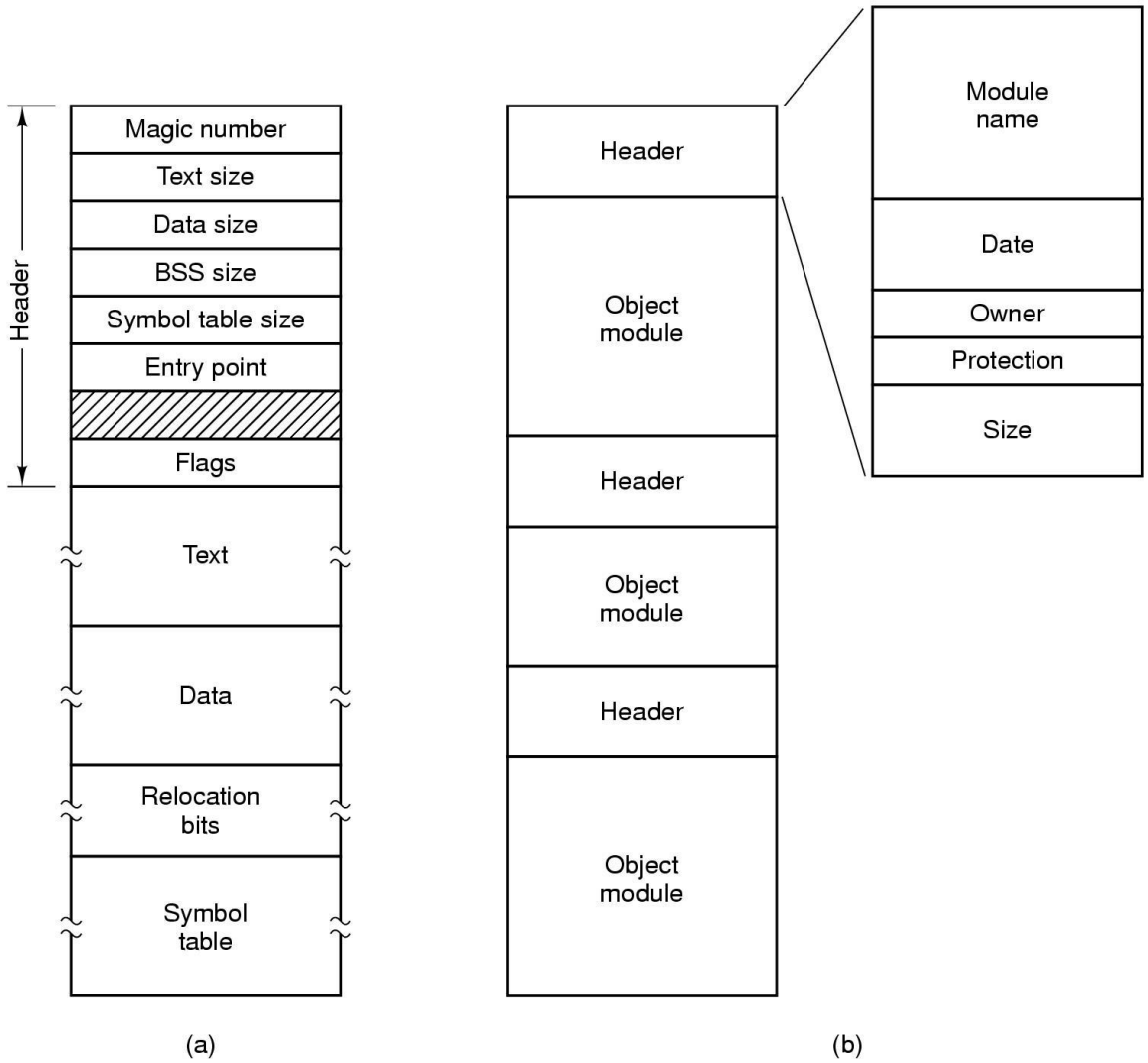
Типове файлове

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Структура на файл

- Без структура – последователност от байтове
- Проста структура - запис
  - Редове
  - Фиксирана дължина
  - Променлива дължина
- Сложна структура
  - Форматиран документ
- Кой определя структурата
  - Операционната система
  - Приложенията

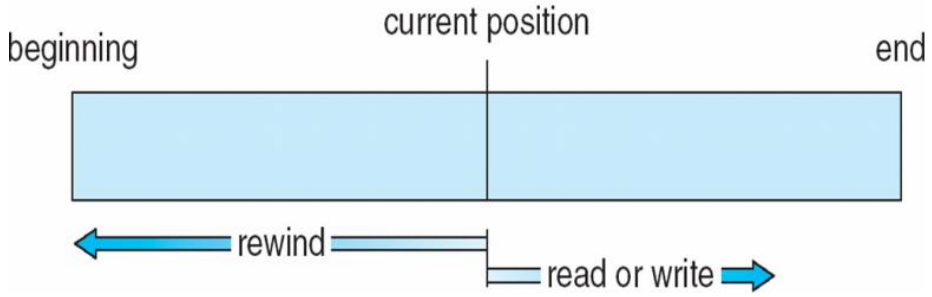
Структура на файл (2)



Методи на достъп

Последователен достъп

- Обща структура



- Операции:
  - **read_next ()** – чете следваща част от файла и автоматично увеличава файловия указател.
  - **write_next () –** добавя в края на файла **и** увеличава файловия указател до края на файла.
  - **reset()** – връщане към началото на файла.

Методи на достъп

Директен достъп

- Файлът се състои от записи с фиксиран размер.
- Файлът се разглежда като номерирана последователност от блокове записи.
- Операции:
  - **read(n)** – чете относителен блок номер n
  - **write(n)** - записва в относителен блок n
- Относителните блокове позволяват на ОС да определи къде те да бъдат разположени
- Удобен начин за непосредствен достъп до големи обеми информация (бази данни)

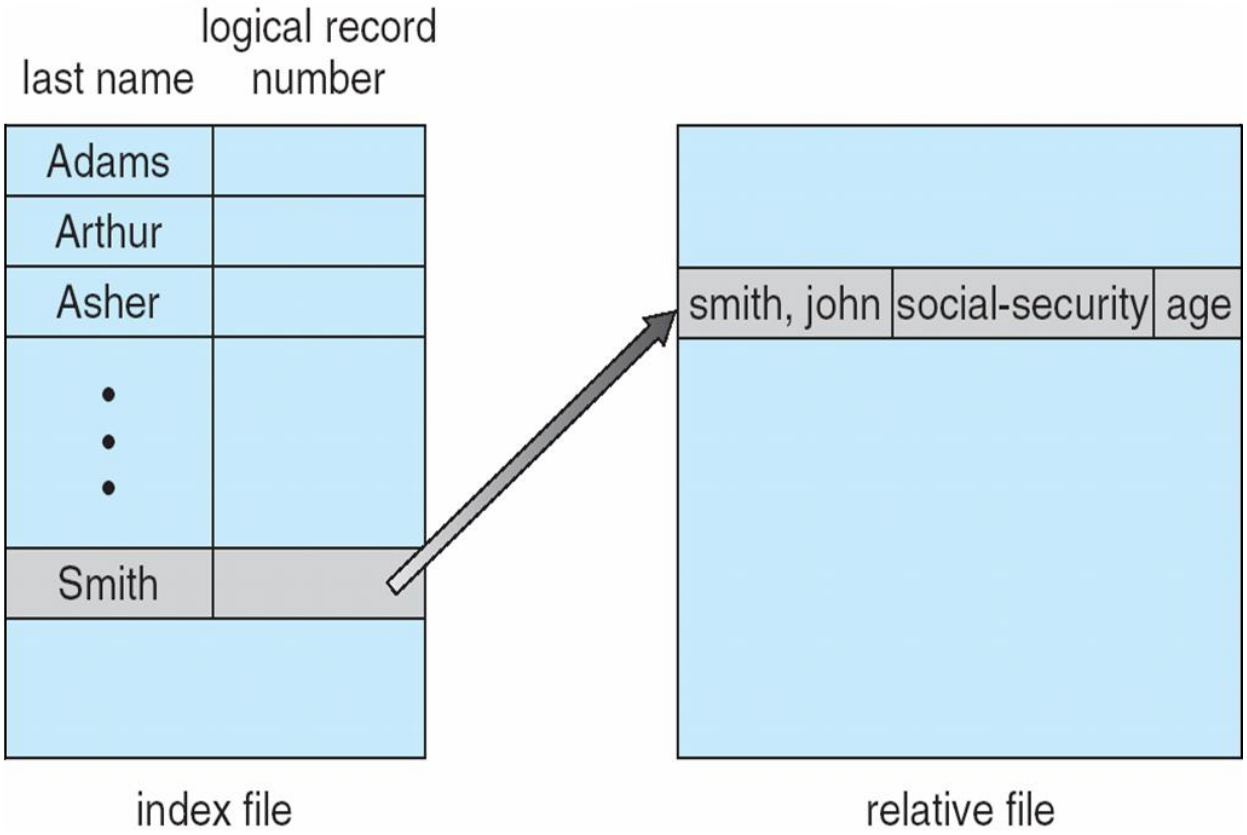
Симулиране на последователен достъп при файл с директен достъп

sequential access	implementation for direct access
reset	$cp = 0;$
read next	*read cp*; $cp = cp + 1;$
write next	*write cp*; $cp = cp + 1;$

Други методи на достъп

- Могат да се изградят върху базовите методи
- Най-общо изискват създаване на индекс за файла
- Индексът се съхранява в паметта
- Ако е прекалено голям, в паметта се съхранява индекс на главния индекс (на диска)
- IBM индексиран последователен метод за достъп (ISAM)
  - Малък главен индекс, сочещ към дискови блокове на вторичния индекс
  - Файлът се съхранява сортиран по ключ
  - Реализира се от ОС

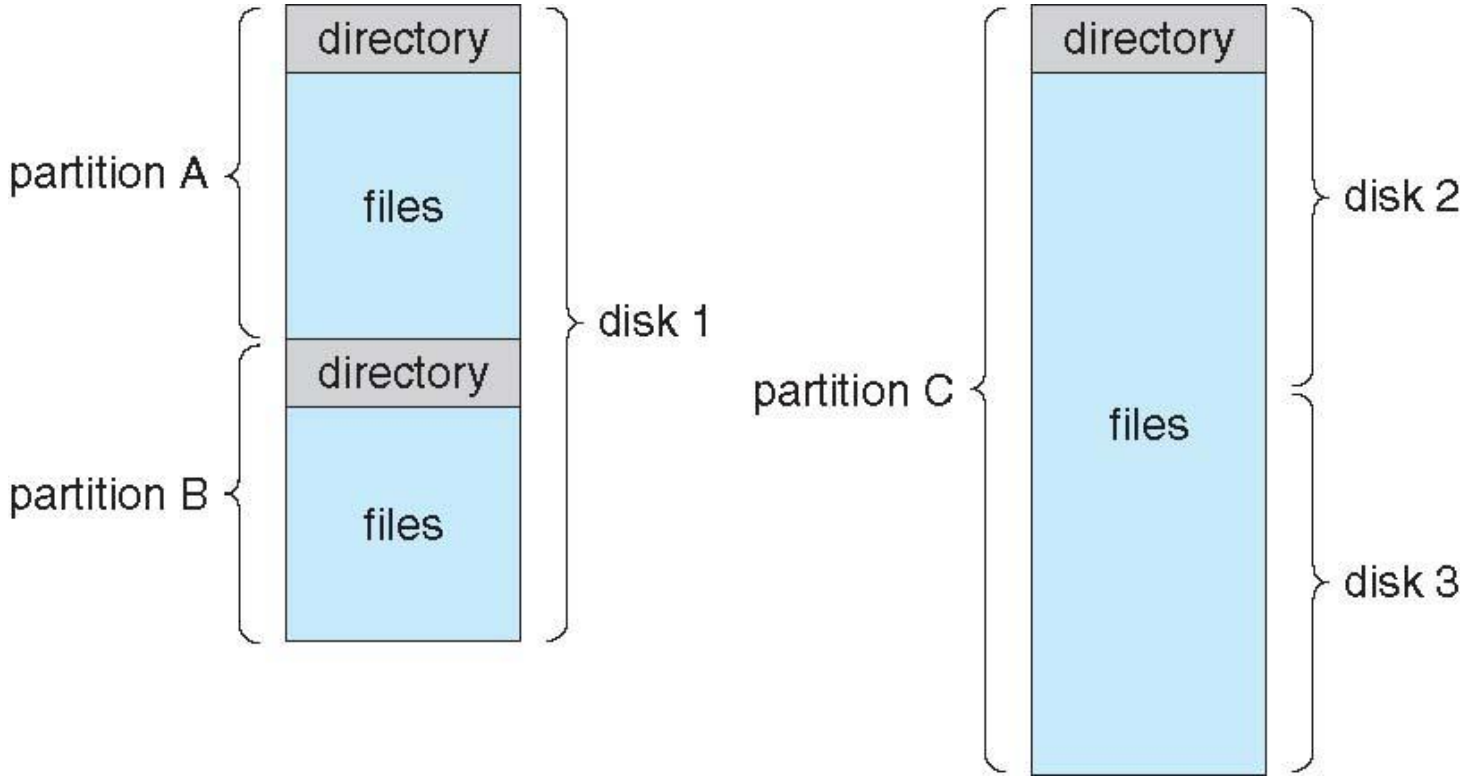
Пример за индекс



Структура на диска

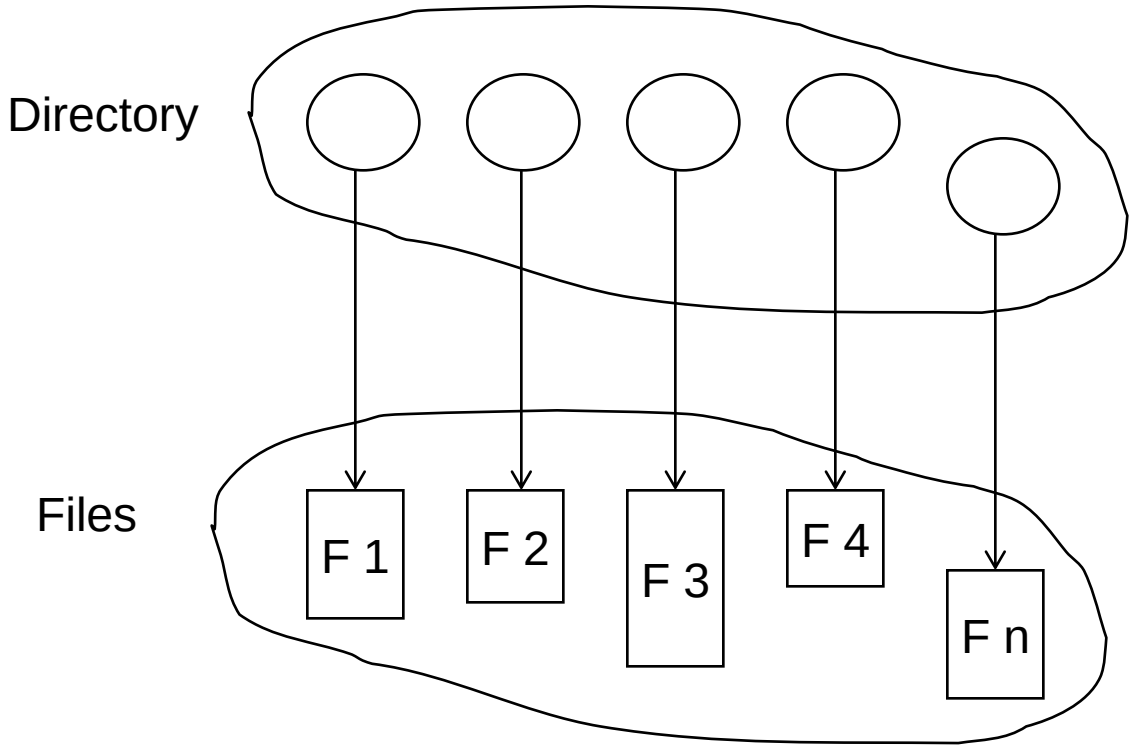
- Дискът може да се раздели на **partitions (дялове)**
- Диск или дял могат да се използват **raw** – без файлова система или **formatted** с файлова система
- Единицата, съдържаща файлова система - **volume**
- Всеки том съдържа информация за файловете в него, съхранена в **device directory** или **volume table of contents**
- Директорията съдържа информация като: име, местоположение, размер и др. за всички файлове в тома

Типична организация на ФС



Структура на директориите

Директория – съвкупност от възли, съдържаща информация за всички файлове



Операции върху директории

- Търсене на файл
- Създаване на файл
- Изтриване на файл
- Извеждане списък на директория
- Преименуване на файл
- Достъп до всяко място

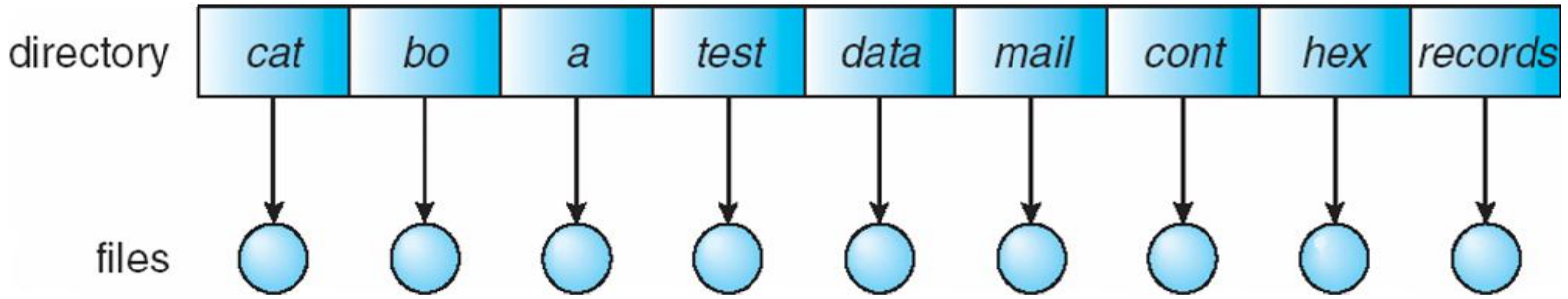
Организация на директории

Директорията се организира логически за:

- Ефикасност – бързо откриване на файл
- Именуване – удобство за потребителите
  - Двама потребители могат да използва едно и също име на файл
  - Един файл може да има няколко различни имена
- Групиране – логическо групиране на файлове (игри и др.)

Директория с единично ниво

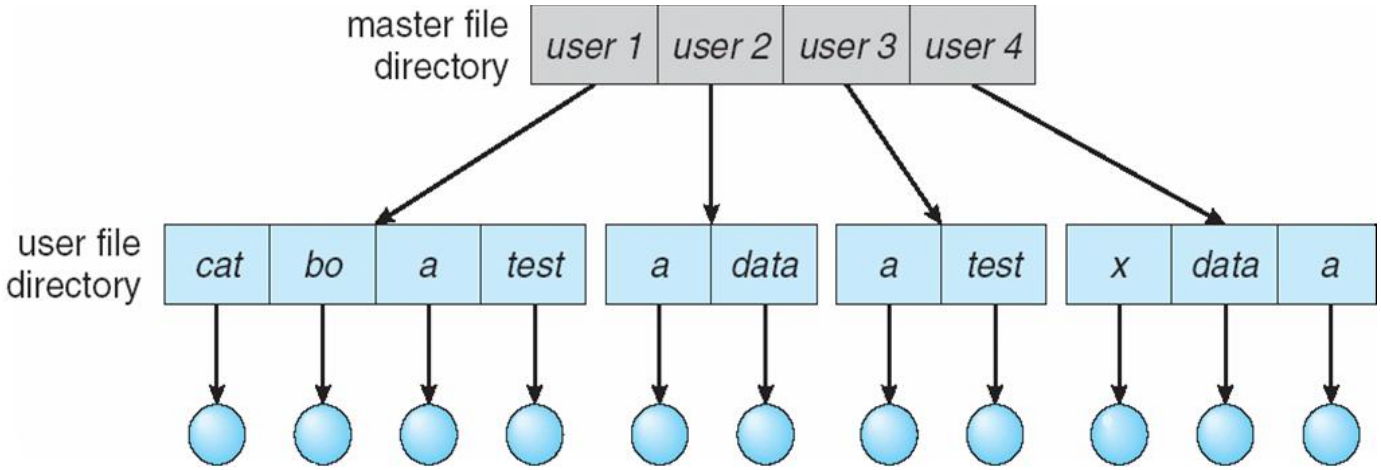
- Единична директория за всички потребители



- Проблем с именуването
- Проблем с групирането

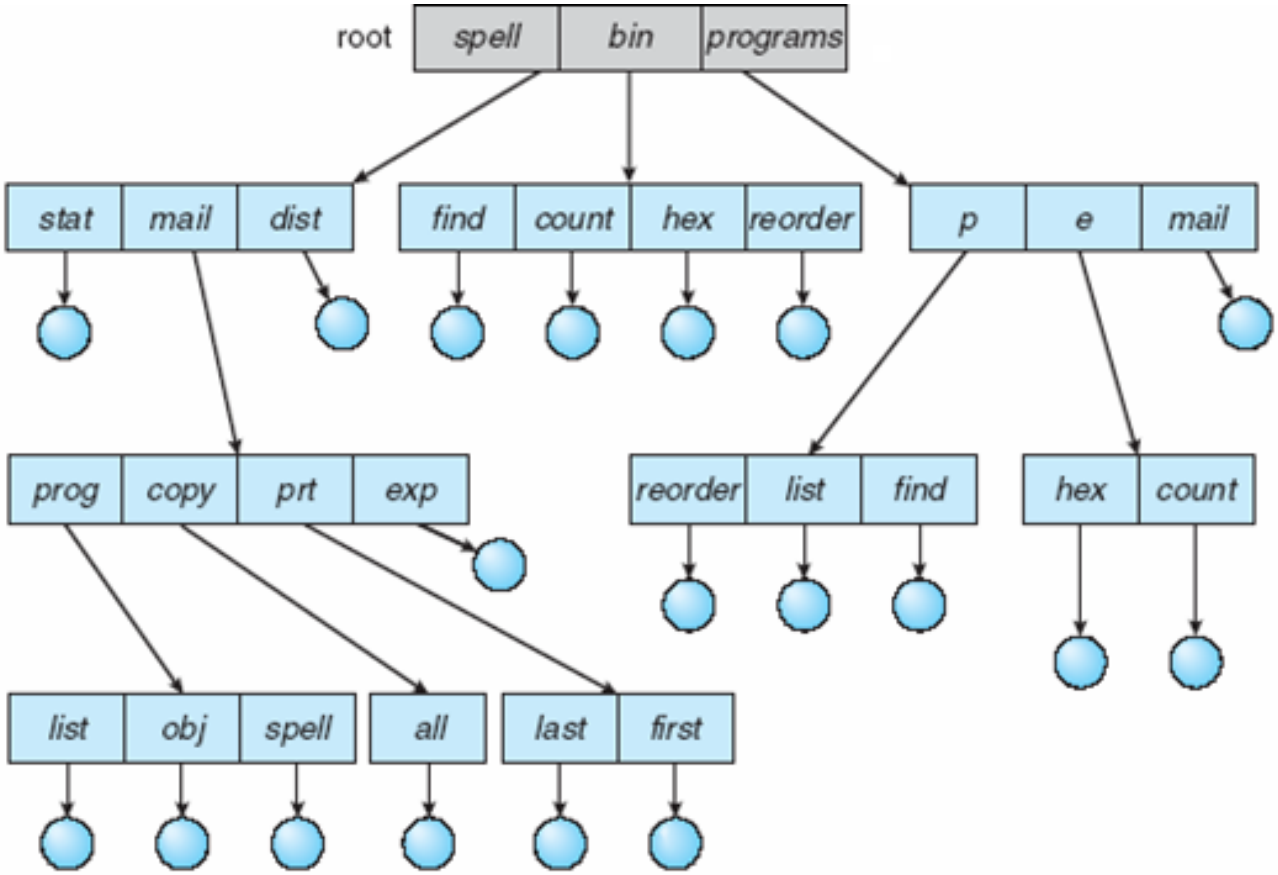
Директория на две нива

- Отделна директори за всеки потребител



- Име на път
- Еднакви имена за различни потребители
- Ефикасно търсене
- Няма възможност за групиране

Дървовидна структура



Дървовидна структура - особености

- Ефикасно търсене
- Възможност за групиране
- **Абсолютно** или **относително** име на път
- Създаването на файл става в текущата директория
- Изтриване на файл

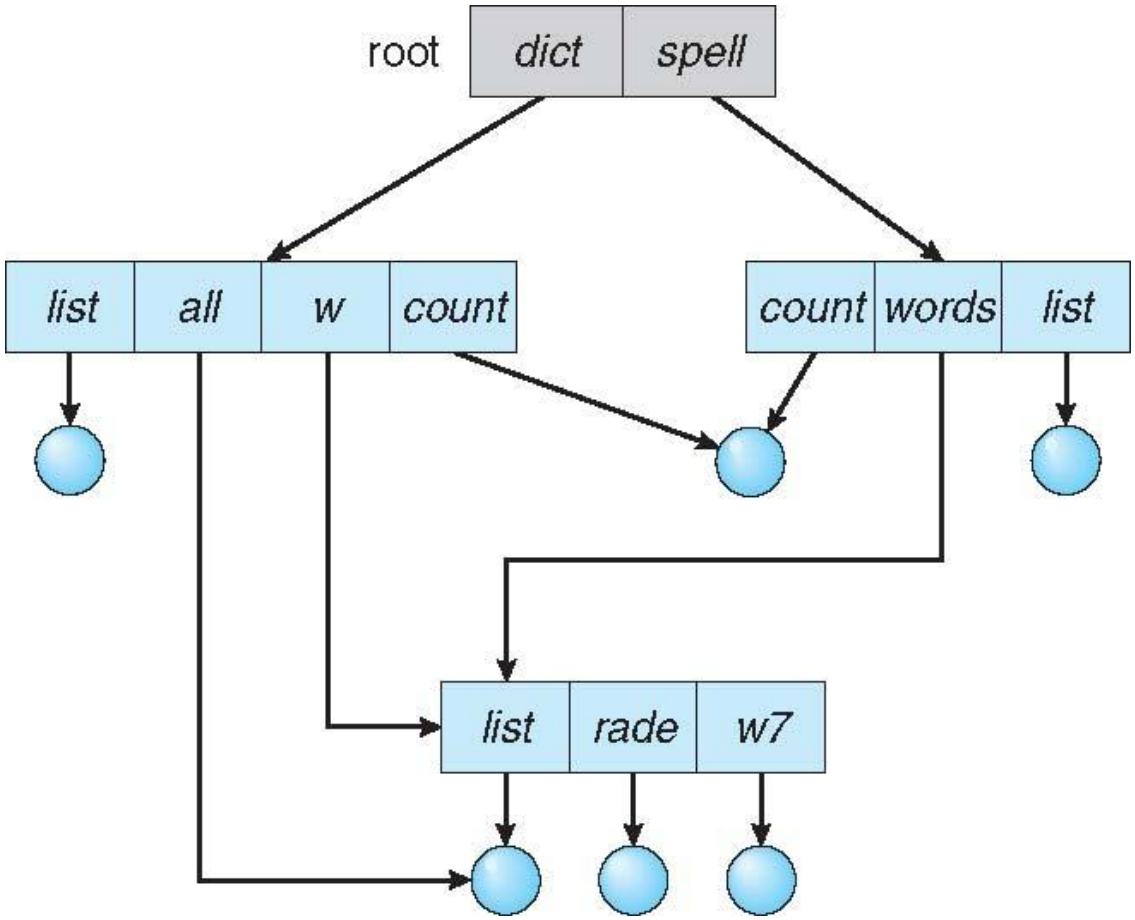
```
rm <file-name>
```

- Създаването на нова поддиректория е в текущата

```
mkdir <dir-name>
```

- Изтриване на директория рефлектира в изтриване на цялото ѝ поддърво

Структура ацикличен граф



- Възможност за споделяне

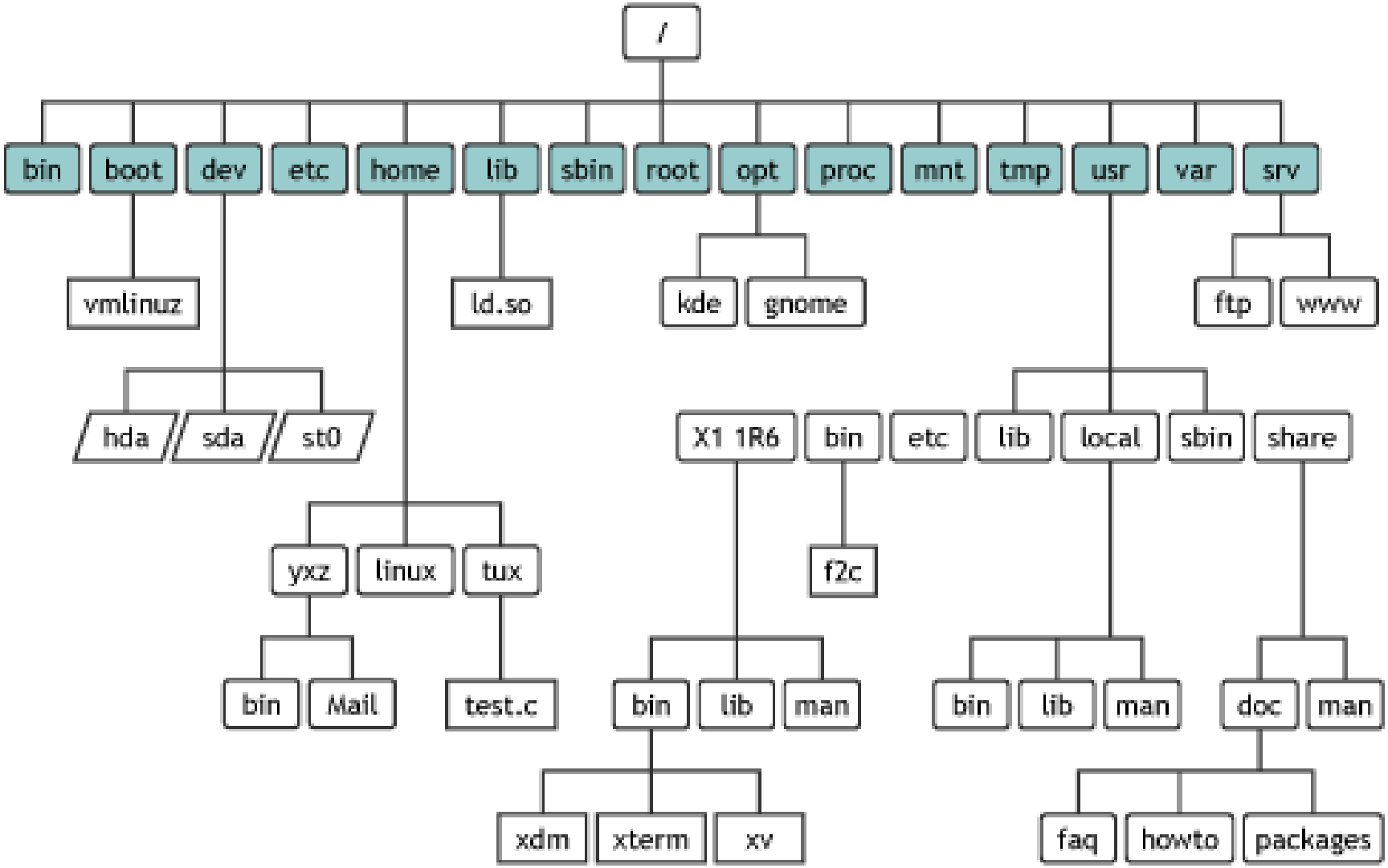
Структура ацикличен граф (2)

- Две различни имена (псевдоними)
- Реализиране на споделените файлове и директории?
  - общ подход (Unix) е използване на линкове (**link**).
  - Link – указател към файл или поддиректория
  - реализира се като абсолютно или относително име

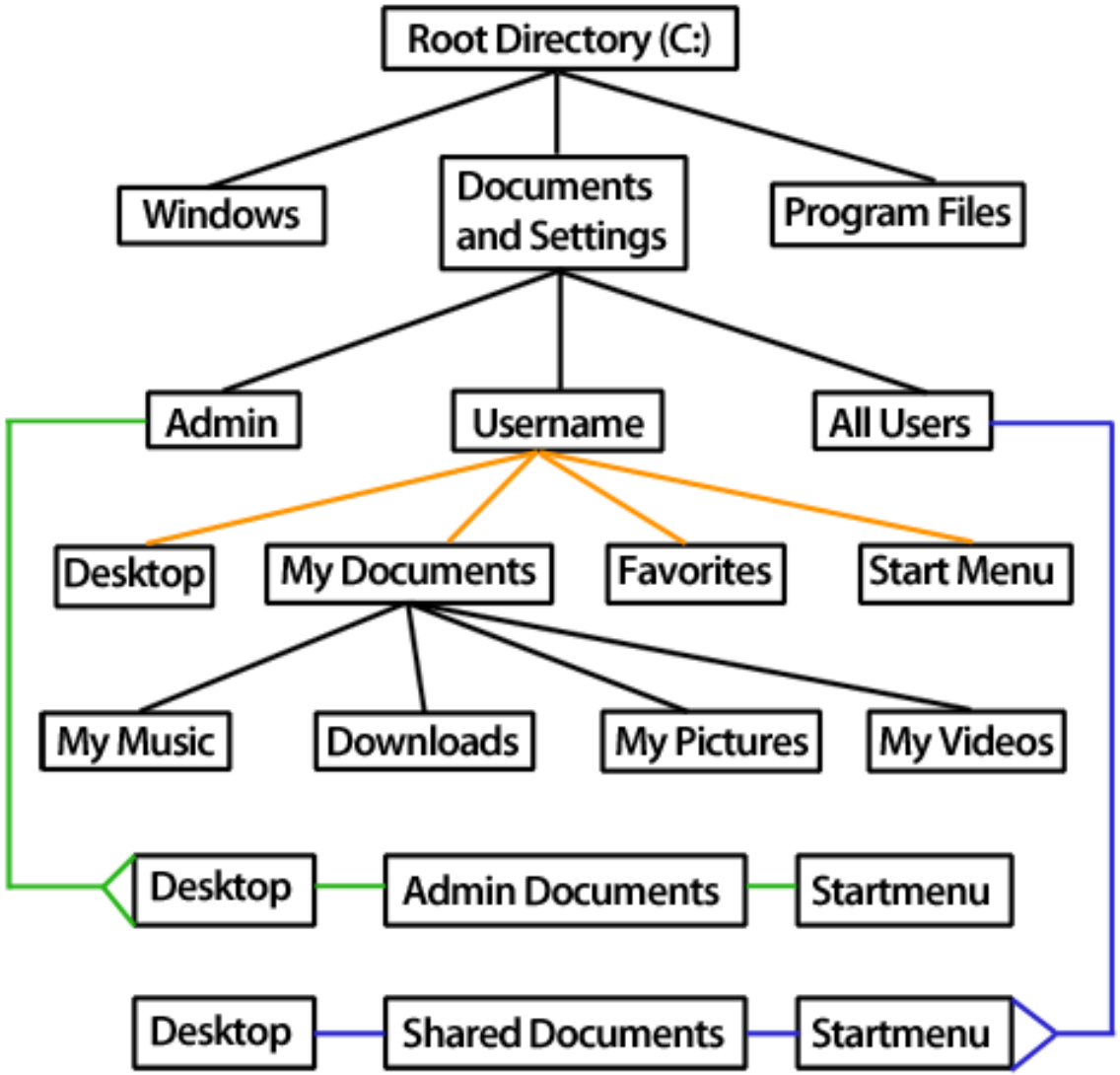
Линкове в Unix

- **Hard link** – множество директорни записи сочат към един и същи файл. Валидни са за обикновени файлове в същата файлова система
- **Symbolic link** – специален тип файл, съдържащ информация къде да се открие свързан файл. Могат да се използват за свързване на файлове и директории и от други файлови системи

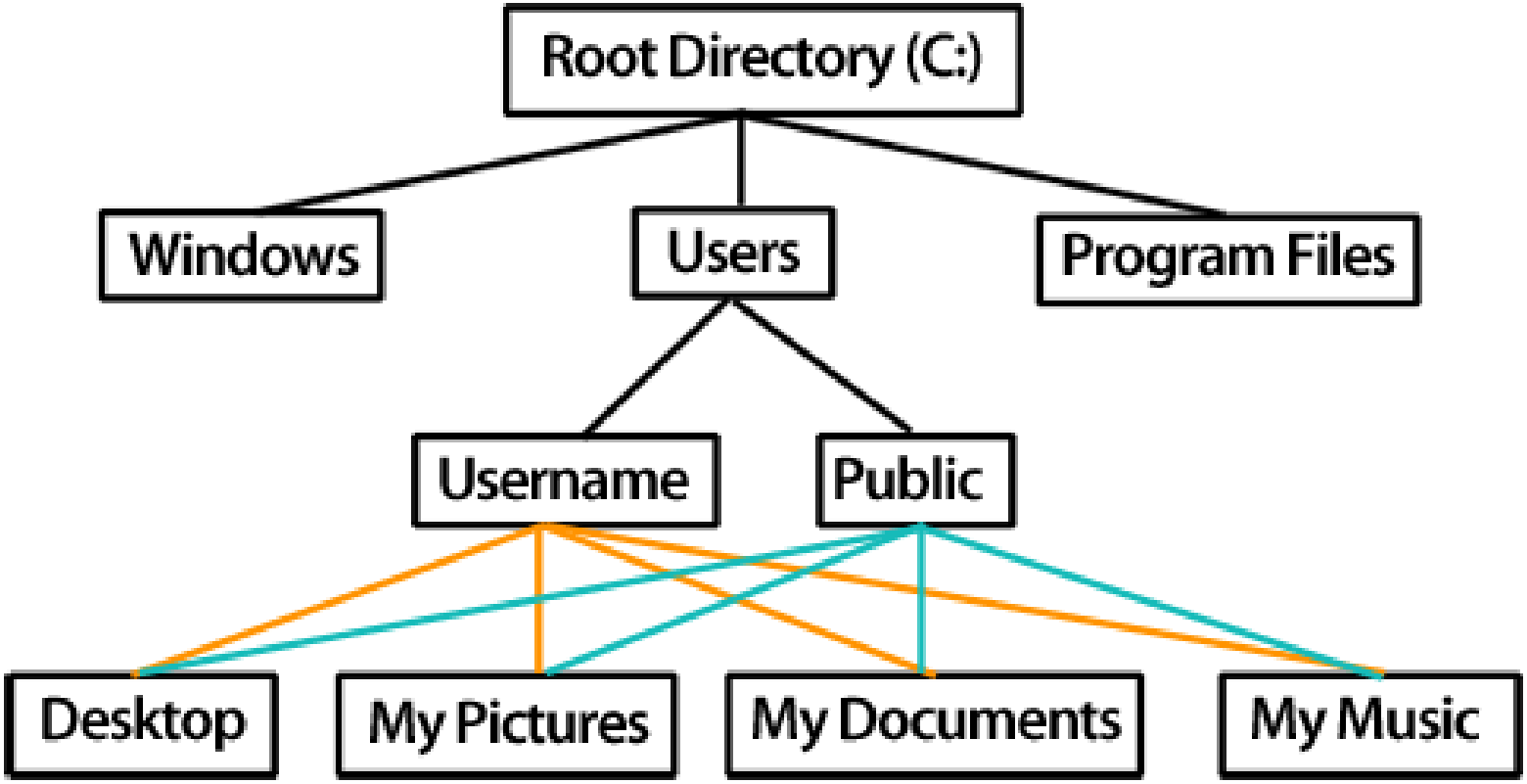
Директорна структура Linux



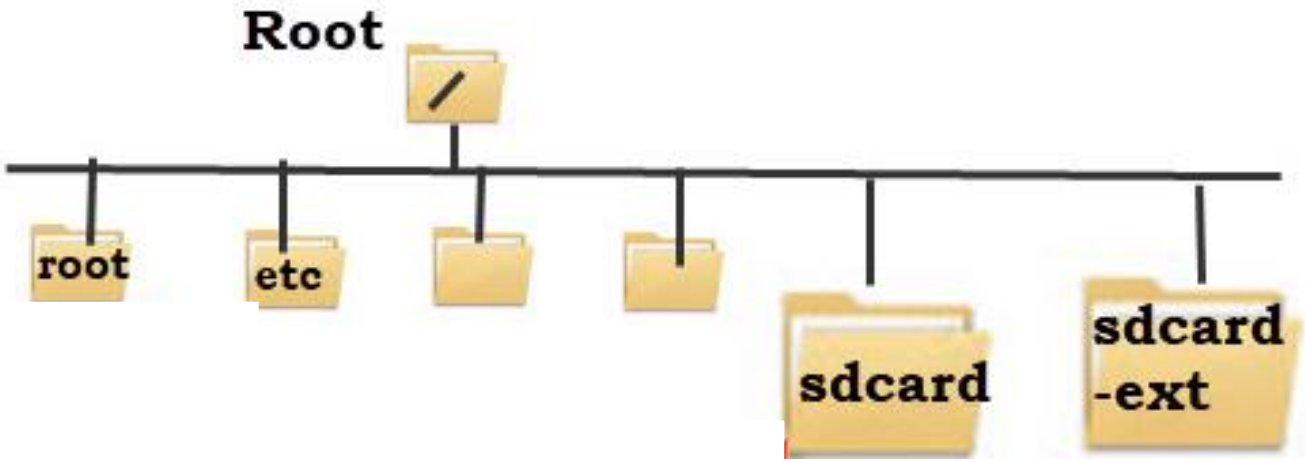
Директорна структура Windows XP



Директорна структура Windows 7



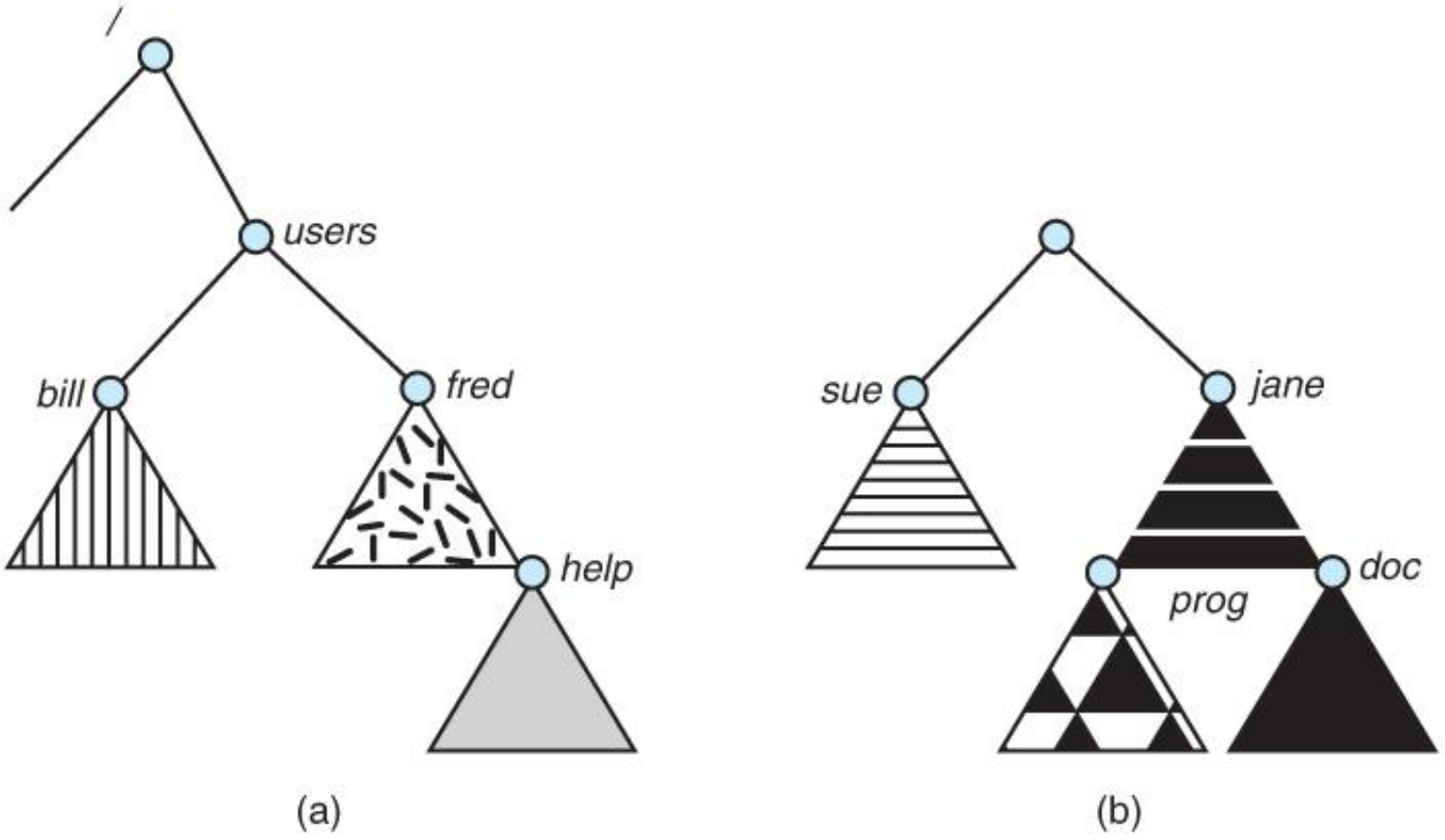
Директорна структура Android



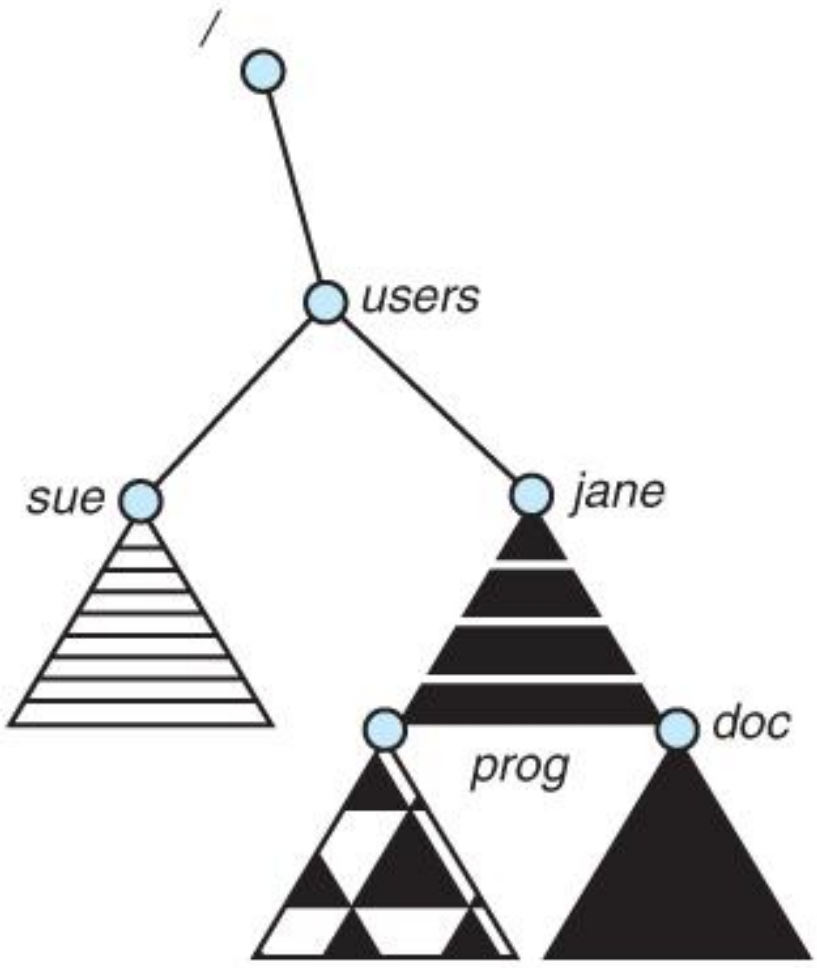
Монтиране на файлова система

• Основна идея – комбиниране на множество файлови системи в една голяма дървовидна структура
• Изисква се *точка на монтиране* (директория)
• Всички файлове в точката на монтиране не са достъпни от монтираната файлова система

Монтиране на файлова система (2)



Монтиране на файлова система (3)



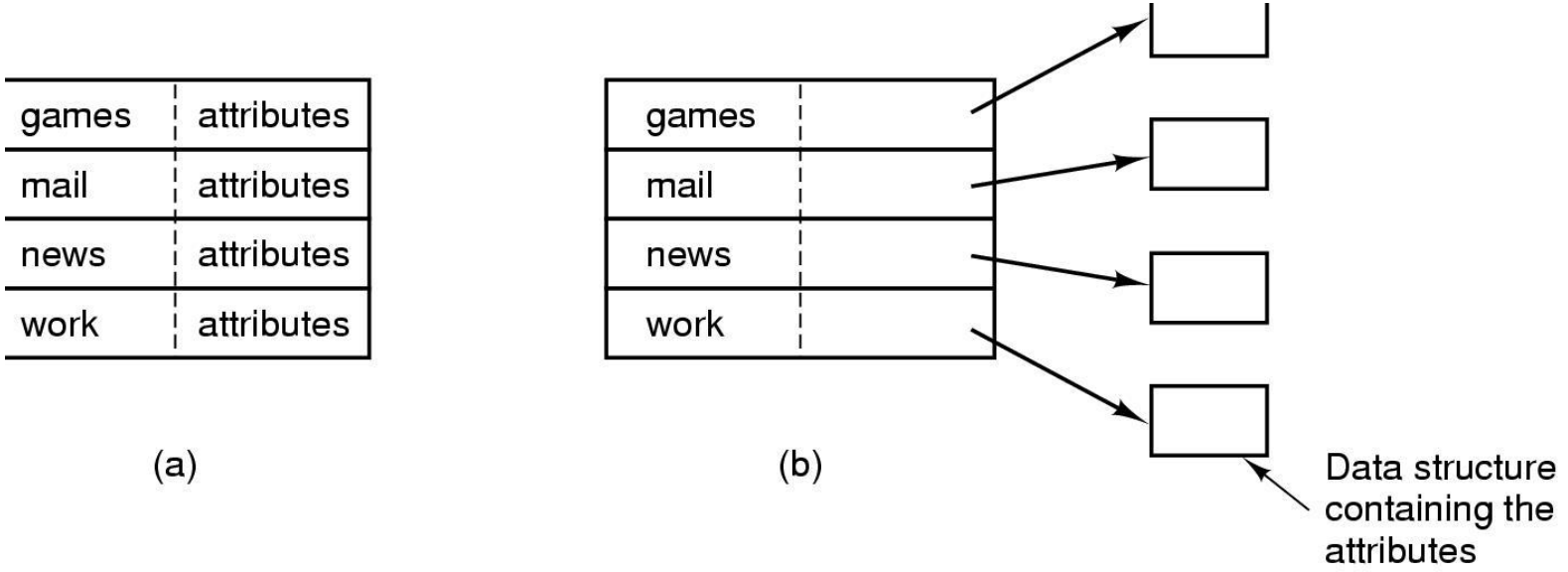
Контролни структури на диск

- Boot control block (boot block, partition boot sector);
- Partition control block (superblock, Master File Table);
- Директорна структура;
- File Control Block

Контролни структури (2)

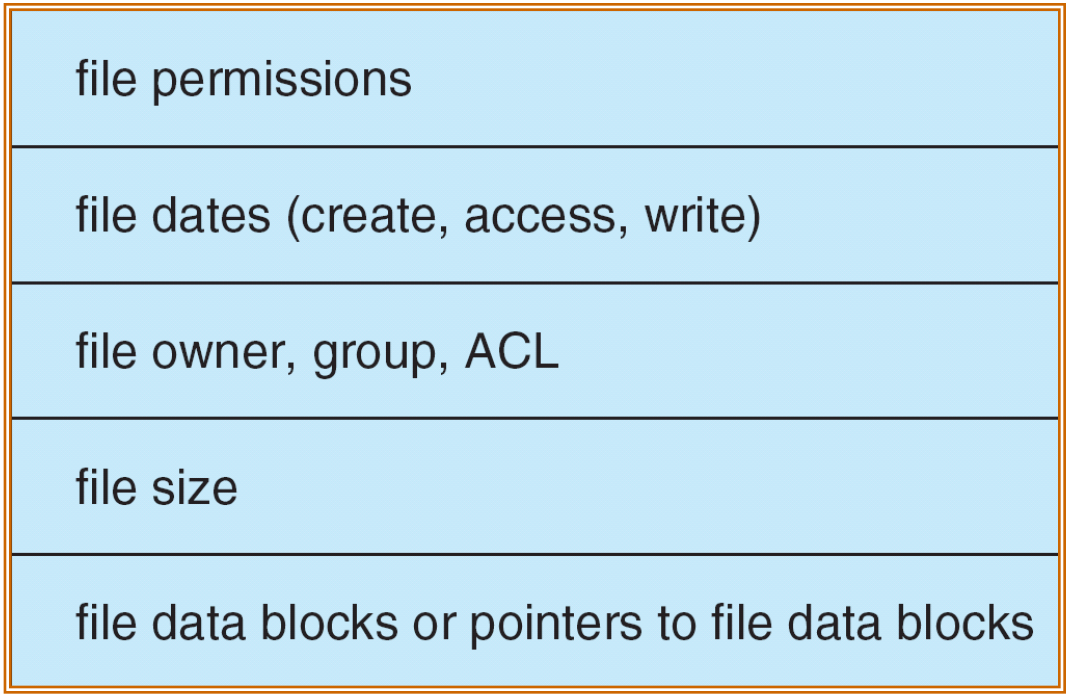
Съхранение на информацията:

- Всеки елемент съдържа всичката информация
- Свързан е към i-node

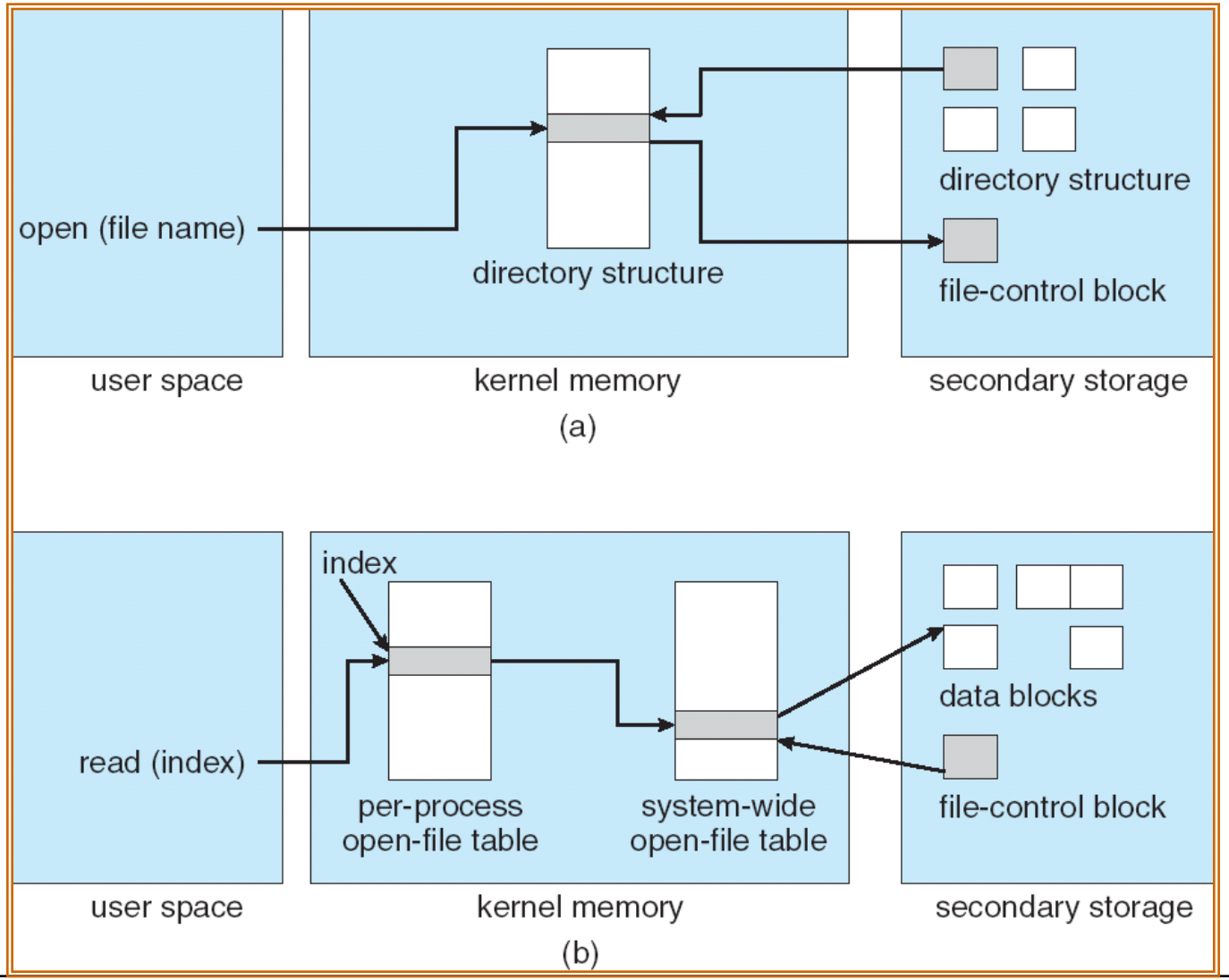


Контролни структури (3)

- File Control Block - съдържа цялостна информация за файл (**i-node**)



Отваряне и четене от файл



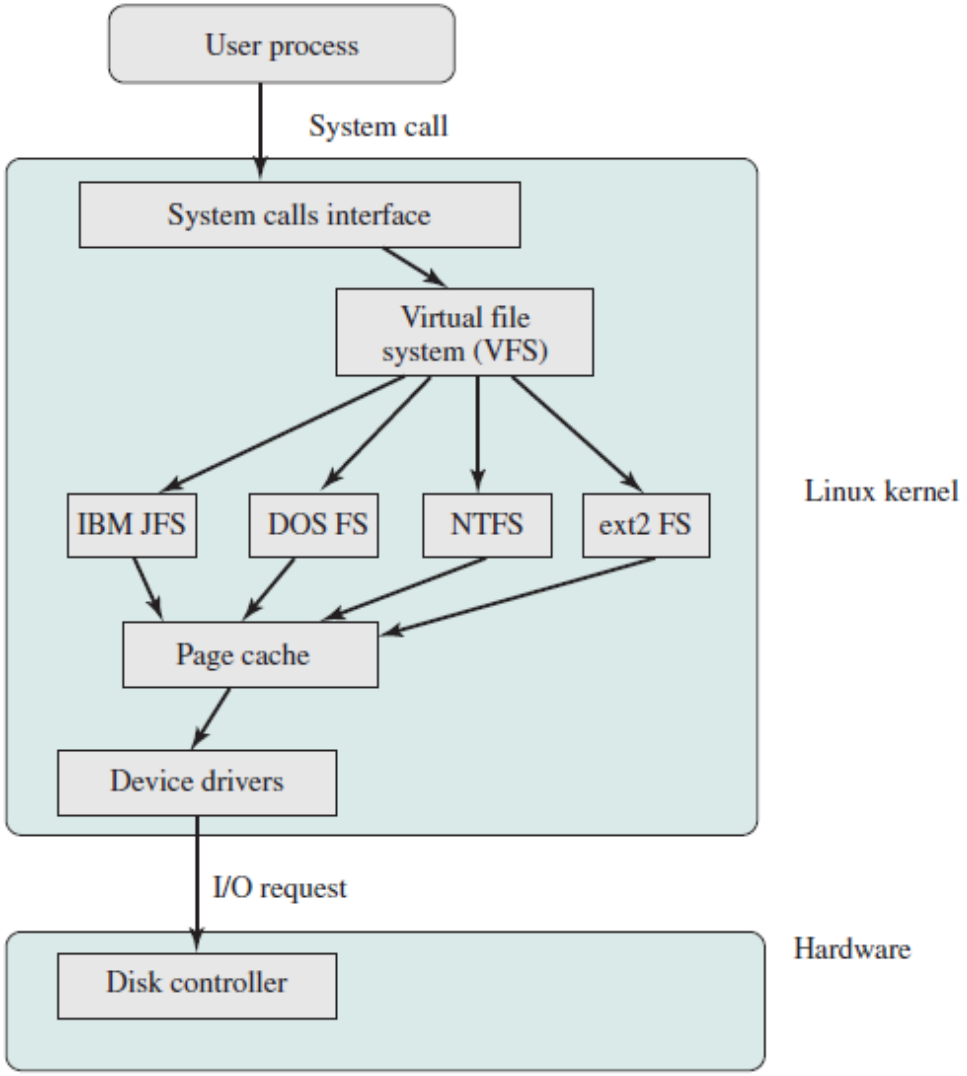
Търсене в директория

- Линейно търсене
- Алтернативи:
  - Hash таблица за директория
  - Кеш за директория

Виртуална файлова система

- **Virtual File System** – предоставя обектно-ориентиран начин за реализиране на файловата система
- Позволява един и същи интерфейс (API) да се използва за различни типове системи
- API се използва за VFS интерфейс

Виртуална файлова система (2)



Защита

- Собственикът (създател) на файла трябва да може да контролира:
  - Какво може да бъде направено
  - От кого
- Типове достъп
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Контрол на достъп в Linux

- Режими на достъп: **read, write, execute**
- Три класа потребители в Unix / Linux:
  - **owner**
  - **group**
  - **public**
- За всеки файл и поддиректория има 9 защитни бита: 3 за owner, 3 за group, 3 за public.

Листинг на директория в Linux

```
drwxr-xr-x  2 webmonster users 4096 Jul  4  2012 htdig/
drwxr-xr-x  2 root       root  4096 Dec 17  2015 magento/
drwxr-xr-x 14 webmonster users 4096 Aug  6  2013 manual/
drwxr-xr-x  3 webmonster users 4096 Apr 24  2015 msacademy/
drwxr-xr-x  4 webmonster users 4096 Apr 24  2015 msdnaa/
drwxr-xr-x  2 webmonster users 4096 Apr 24  2015 music/
drwxr-xr-x  8 zarko      users 4096 Jul 16  2015 newhealth/
drwxr-xr-x  2 webmonster users 4096 Apr 24  2015 software/
drwxr-xr-x  5 webmonster users 4096 Apr 24  2015 stpract/
```

Промяна на разрешения

- За всеки отделен файл или поддиректория се дефинира подходящ достъп
- Задаване на защитата - ***chmod***
- Компоненти:
  - Потребител – [***ugoa***] u – собственик, g – група, o – други, a – всички
  - Добавяне/премахване – [**+ -**]
  - Разрешения – **r, w, e**

```
# chmod u+x test.c
# chmod +w test.c
# chmod g-x test.c
```

Промяна на разрешения (2)

- Прилагане на специфично множество от разрешения
- Използва се числово представяне

Стойност	rwe
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Промяна на разрешения (3)

```
# chmod 751 test.c
# chmod 240 test.c
```

Промяна на разрешения за директории

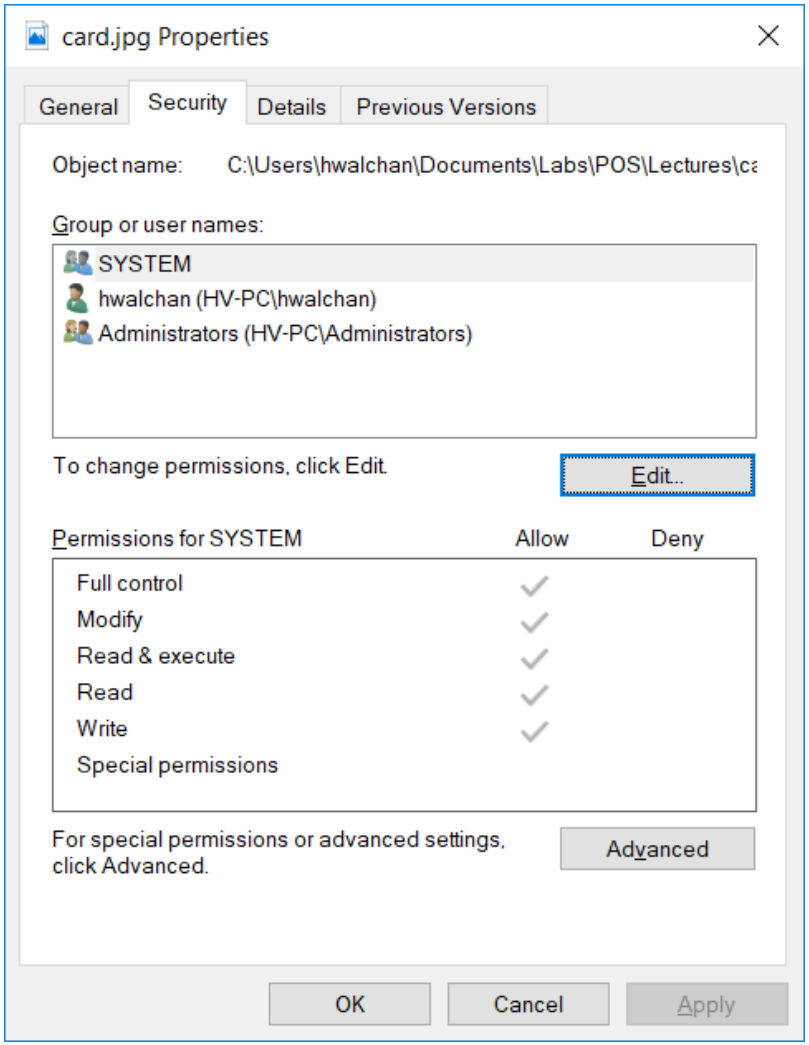
- **R** – четене съдържанието на директорията
- **W** – създаване на файлове и поддиректории в нея
- **X** – възможност за влизане в директорията

Промяна на собственост на файл

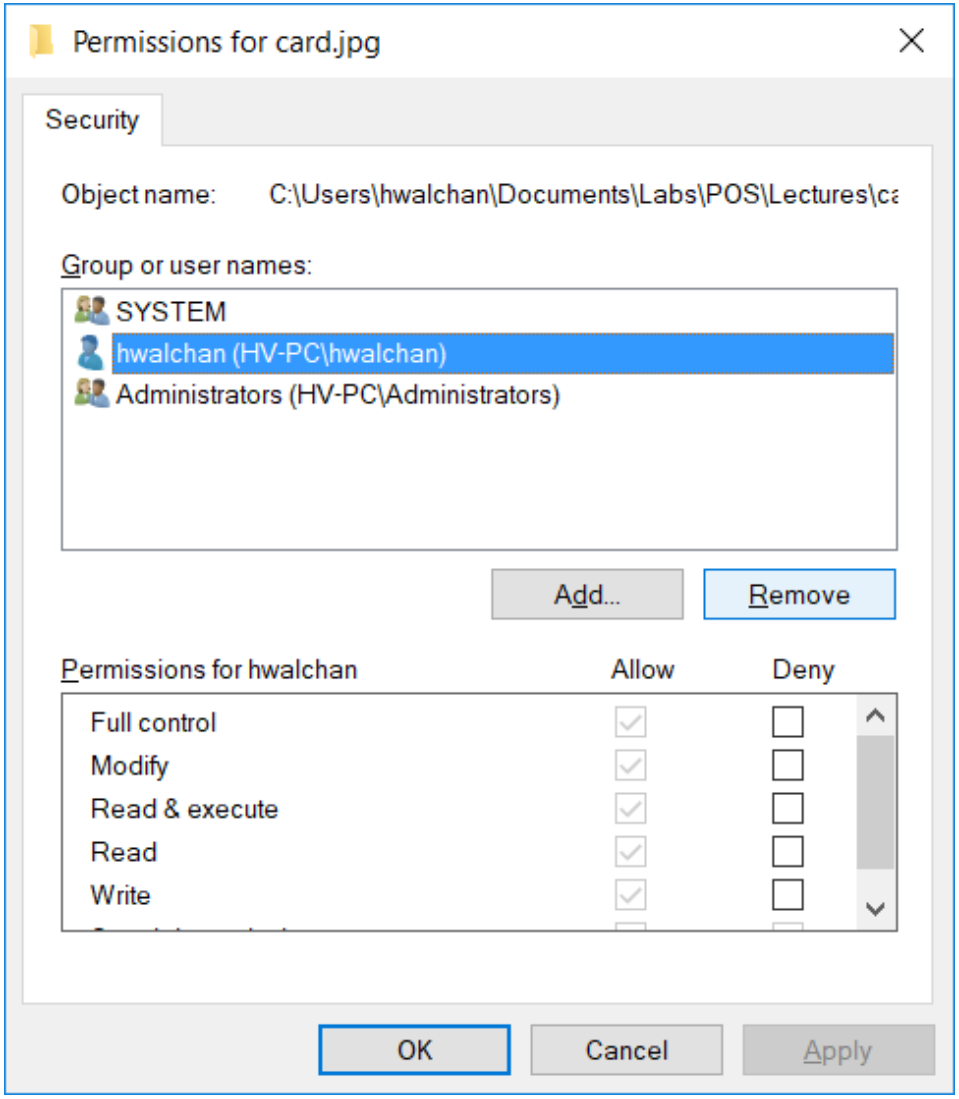
```
# chown <user>:<group> <file>

# chown ivan:users test.c
```

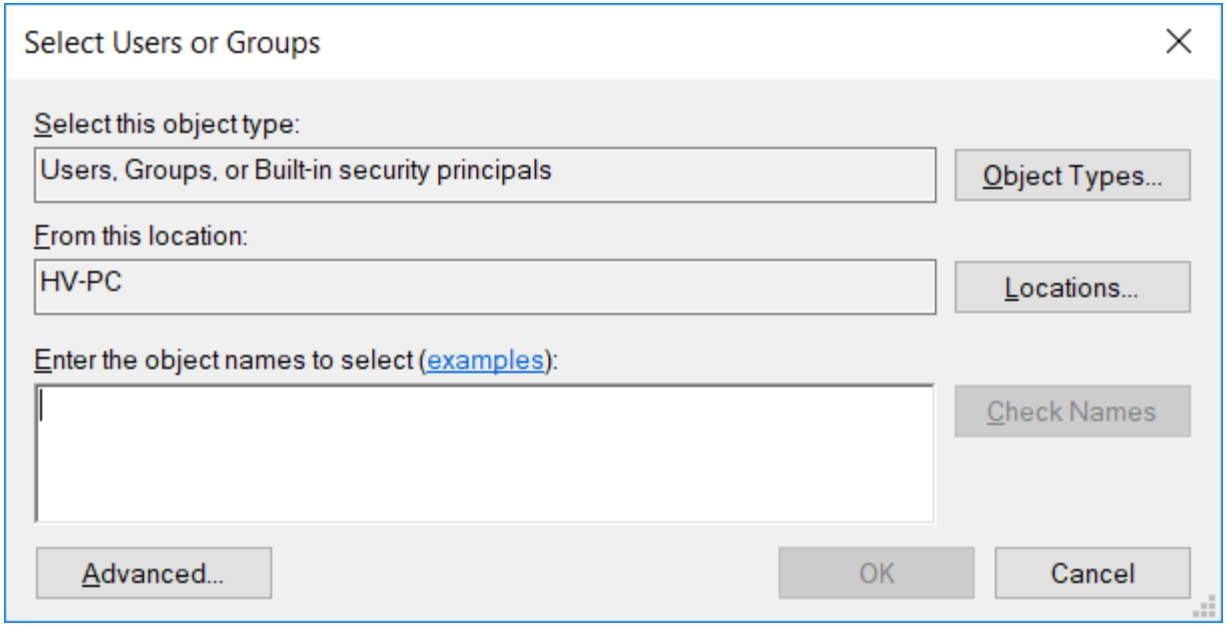
Управление на достъп в Windows



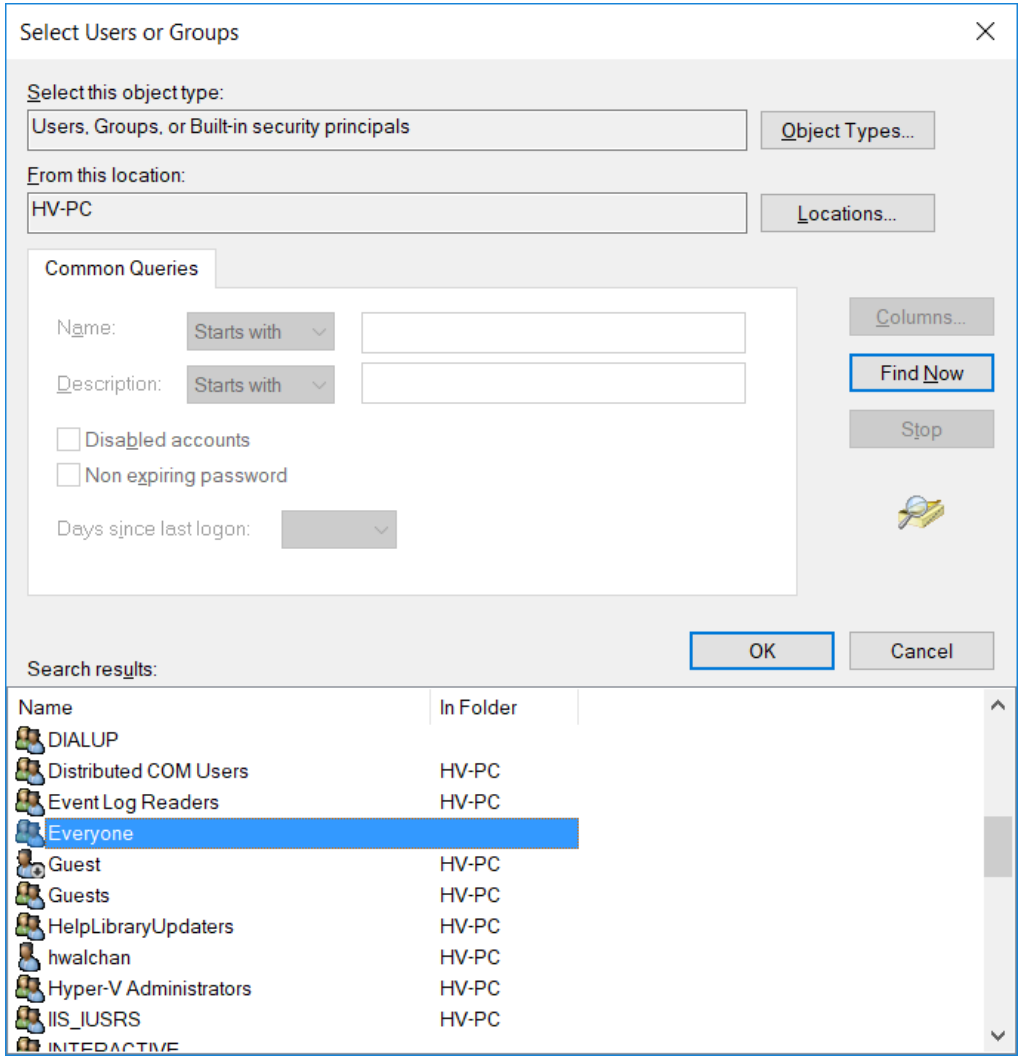
Управление на достъп в Windows



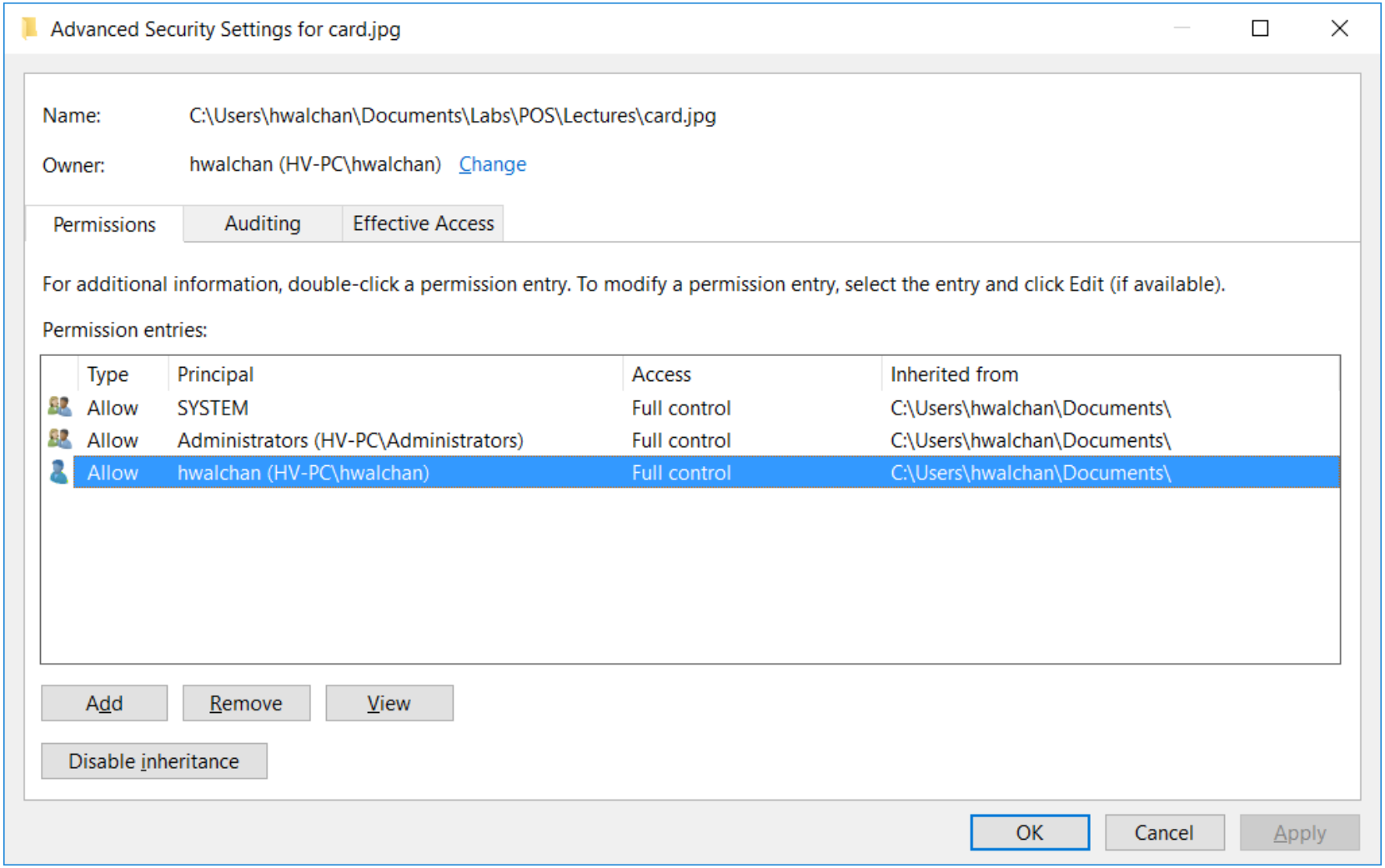
Управление на достъп в Windows



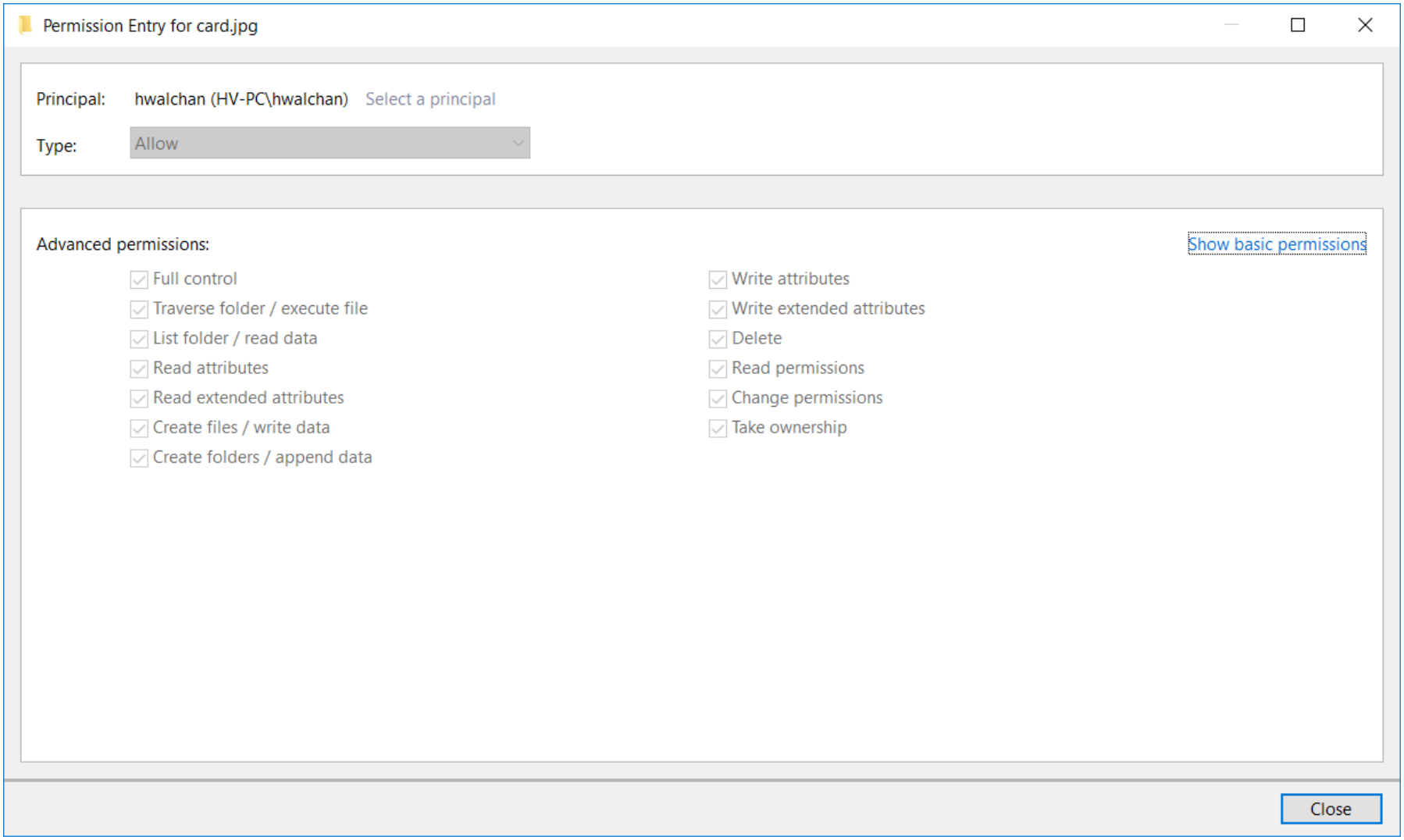
Управление на достъп в Windows



Управление на достъп в Windows



Управление на достъп в Windows



Въпроси ?