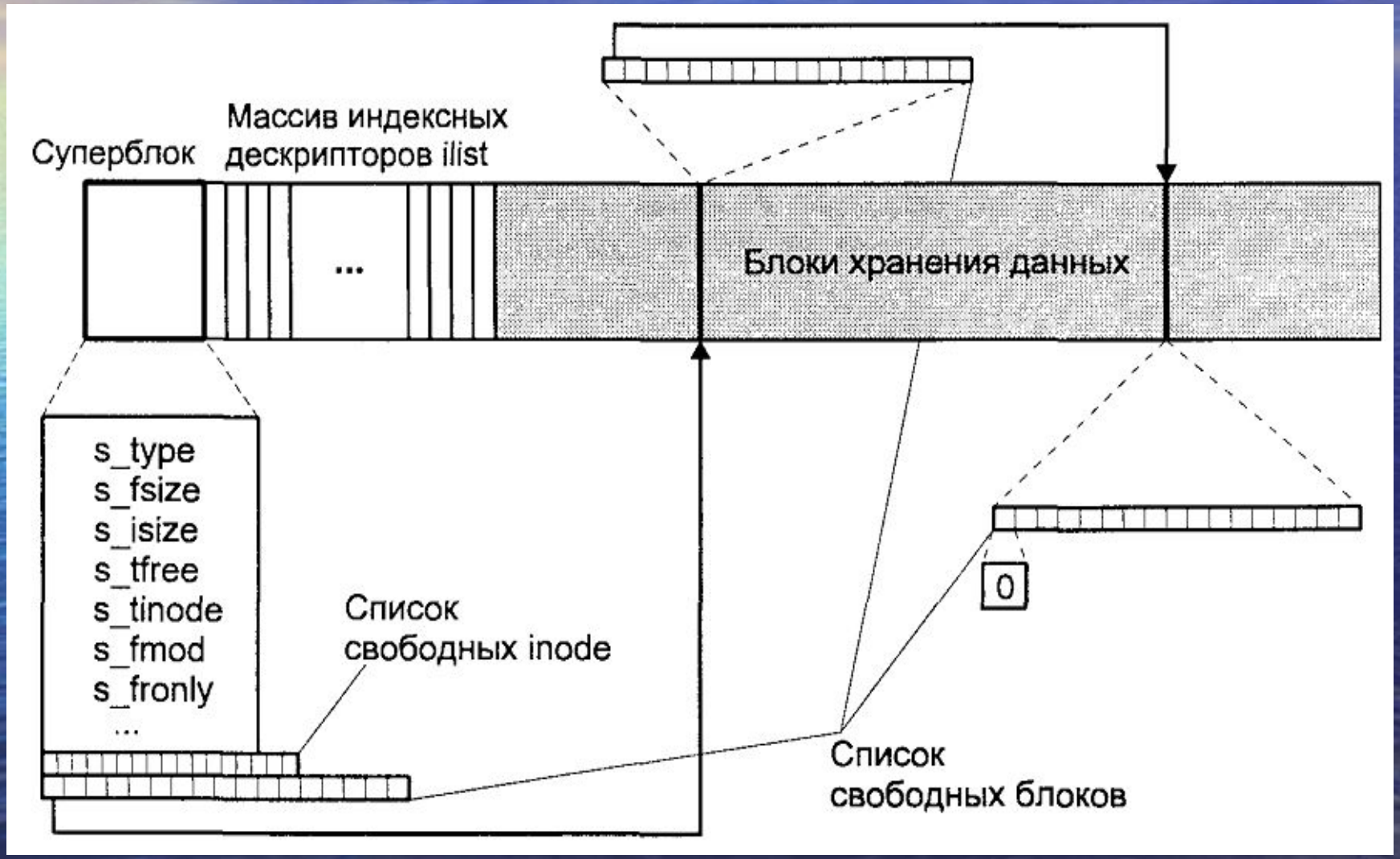
Файловая система ОС UNIX

Базовая файловая система svfs

- Занимает один раздел жесткого диска
- Размер логического блока 1К
- Три области на диске

Структура базовой файловой системы



Суперблок (1)

Содержит общую информацию о файловой системе (ФС), необходимую для монтирования и управления ФС (размещением файлов).
Суперблок считывается в память при монтировании ФС и хранится там постоянно

Суперблок (2)

- Тип файловой системы
- Размер файловой системы в логических блоках (суперблок, массив индексных дескрипторов, область данных)
- Размер массива индексных дескрипторов

Суперблок (3)

- Количество свободных блоков, доступных для размещения
- Количество свободных индексных дескрипторов, доступных для размещения
- Флаги (режимы монтирования)
- Размер логического блока (512, 1024, 2048)

Суперблок (4)

- Список номеров свободных индексных дескрипторов (inode)
- Список номеров свободных блоков

Хранятся особым образом

Массив индексных дескрипторов (inode)

Содержит метаданные всех файлов.

Имеет ограниченный фиксированный размер.

Количество элементов определяет максимальное количество файлов

inode (1)

Содержит метаданные файла (все кроме имени файла и его содержимого):

- di_mode – тип файла, основные и расширенные права доступа
- di_nlinks – количество ссылок (жестких связей, имен)

inode (2)

- di_uid, di_gid – числовые идентификаторы владельца файла и группы
- di_size – размер файла в байтах (для специальных файлов – старший и младший номера устройств)

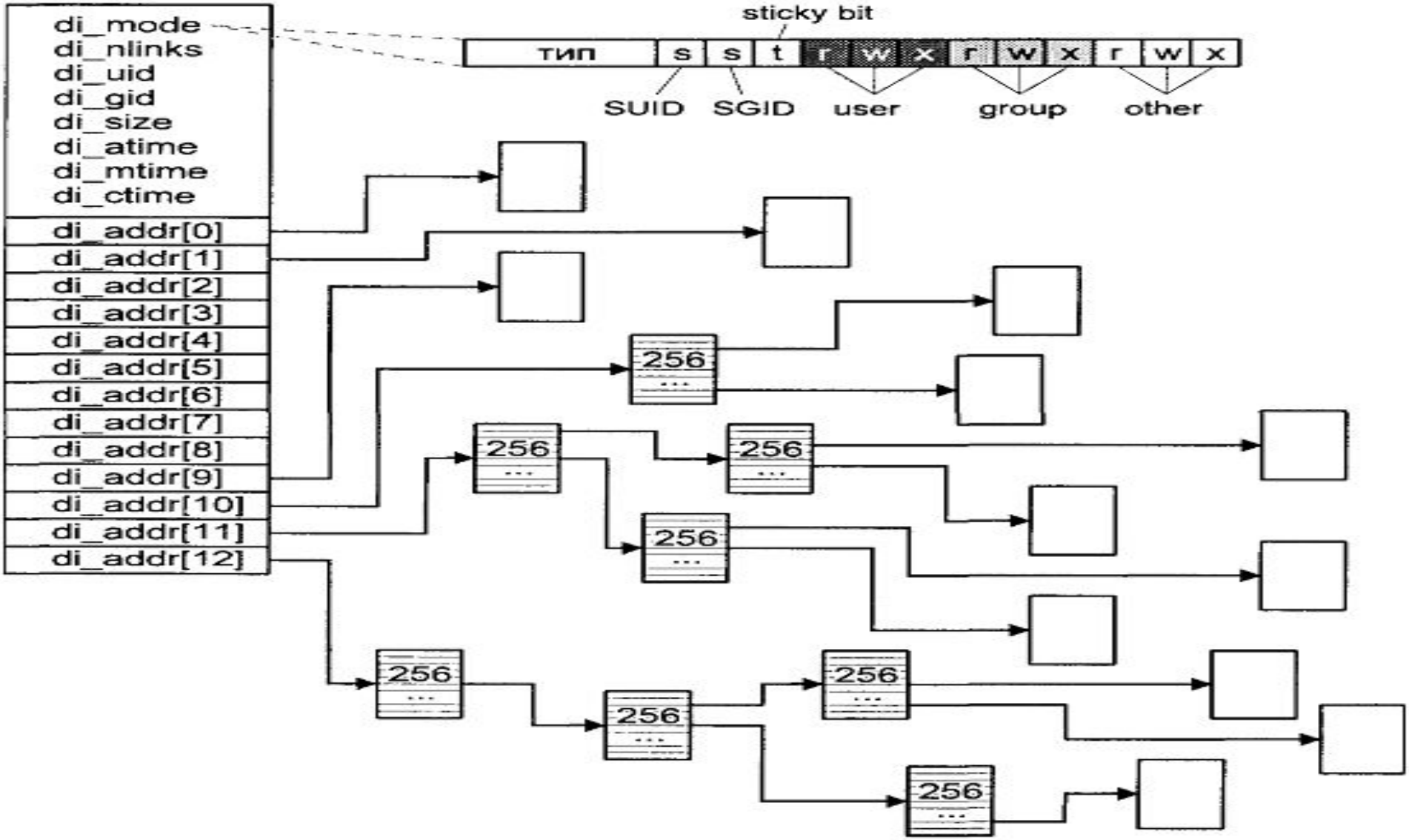
inode (3)

- di_atime – время последнего доступа к файлу
- di_mtime – время последней модификации файла
- di_ctime – время последней модификации метаданных
- di_addr[13] – массив адресов блоков данных

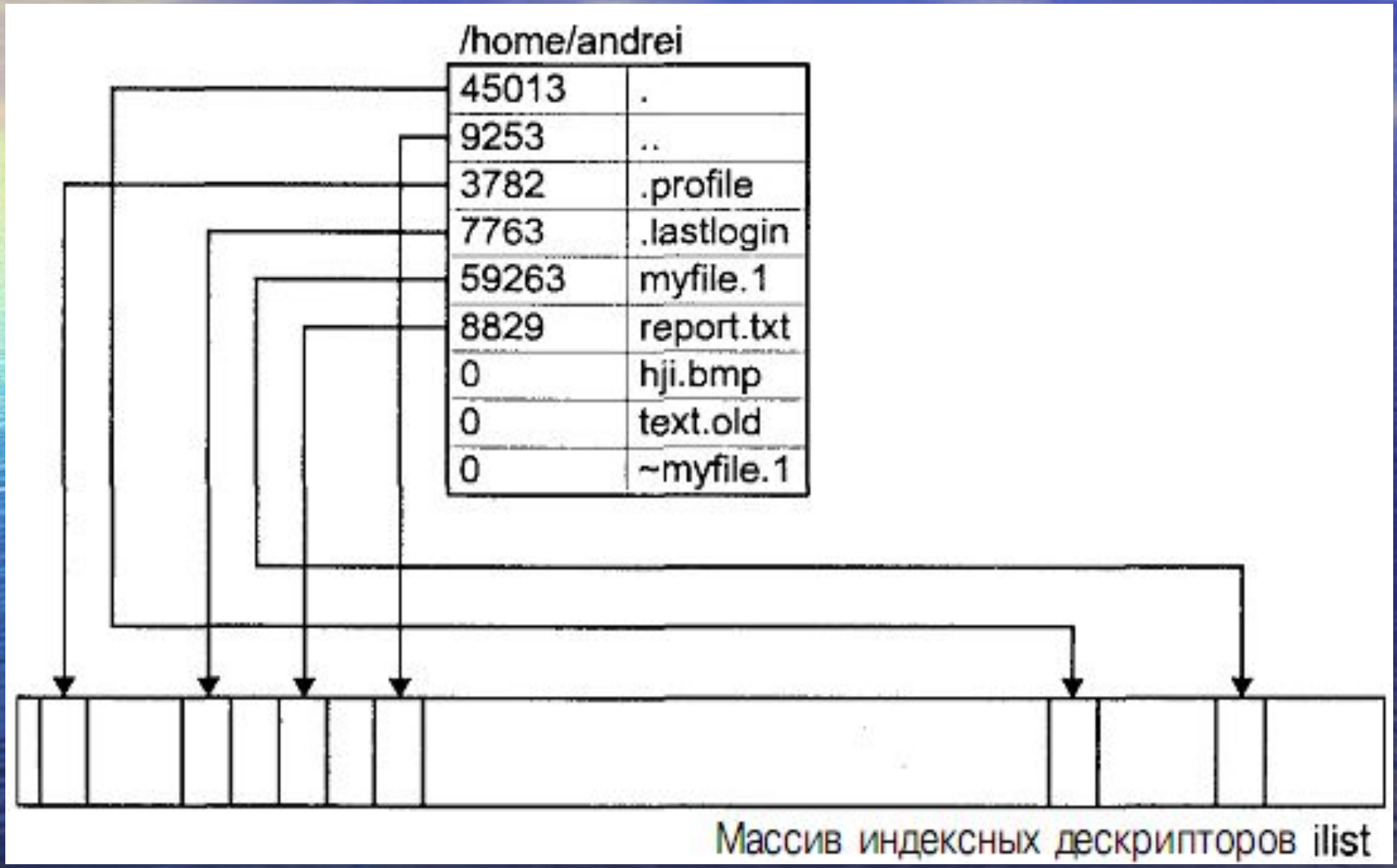
Типы файлов

- Обычный файл -
- Каталог (директория) d
- Символьное устройство c
- Блочное устройство b
- Именованный канал (FIFO) p
- Сокет s
- Символическая ссылка l

Структура inode



Структура каталога



Трансляция имени файла

Абсолютное имя (начинается с ”/”) – относительно корневого каталога с номером inode 2 (или в u-area, если процесс поменял корневой каталог)

Относительное имя – относительно текущего каталога (u-area)

Недостатки svfs (1)

- Суперблок хранится в единственном экземпляре – потеря суперблока – потеря всей ФС
- Метаданные файла располагаются далеко от его данных (медленно + возможна фрагментация)

Недостатки svfs (2)

- Для повышения производительности необходимо использовать блоки больших размеров, но при этом в среднем теряется половина блока
- Фиксированный размер массива inode

Недостатки svfs (3)

- Фиксированный размер имени файла – 14 символов
- Фиксированное количество inode – 65536

Berkeley Fast File System (FFS)

- Появилась в 4.2BSD
- Подверглась существенным улучшениям в 4.3BSD (производительность и надежность)
- Обладает полным функционалом svfs

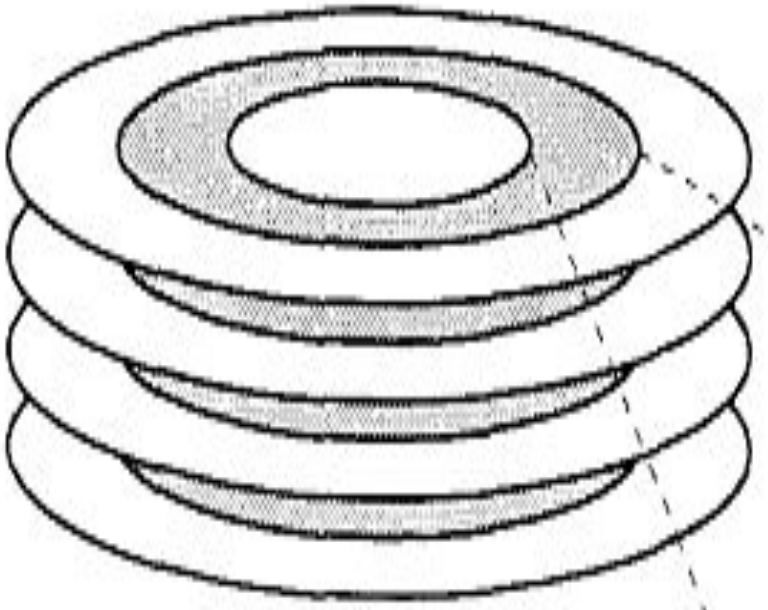
Улучшения в FFS (1)

- Суперблок хранит только неизменяемую часть (нет списков свободных блоков и inode) и хранится в нескольких копиях в разных частях диска

Улучшения в FFS (2)

- Новые принципы размещения информации на диске с учетом его геометрии
- Новая структура каталога

Структура FFS



Группа цилиндров

Для каждой группы цилиндров выделяется место под определенное количество i-node

Обычно 1 inode на 2К дискового пространства (не преодолено ограничение svfs, но повышена надежность, и производительность)

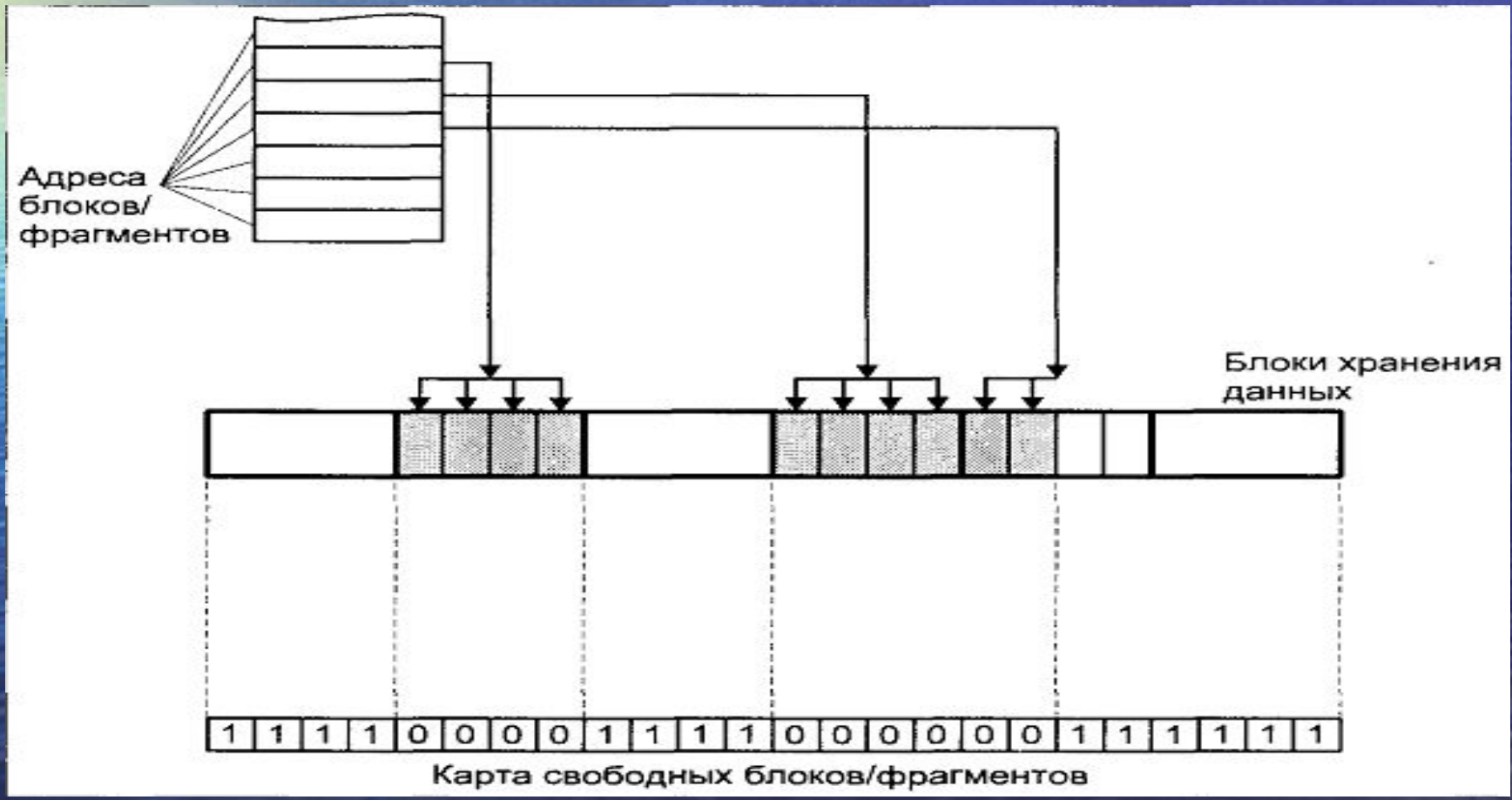
Блоки, состоящие из суперблока, массива inode и блоков данных носит название кластера

Каждый кластер начинается с некоторым смещением от начала цилиндра по сравнению с предыдущим

FFS поддерживает размер блока до 64К

Для повышения эффективности использования вводится битовая карта блоков, позволяющая использовать ½ блока и так до размера физического сектора

Карта свободных блоков



Принципы размещения файлов (1)

- Файл размещается в блоках хранения данных, принадлежащих кластеру, где находятся его метаданные
- Все файлы каталога размещаются в одном кластере

Принципы размещения файлов (2)

- Каждый новый каталог помещается в группу цилиндров, отличную от группы родительского каталога
- Последовательные блоки на диске размещаются со смещением на некоторый угол

Принципы размещения файлов (3)

Удовлетворительные результаты по производительности достигаются, когда есть по крайней мере 10% свободного пространства

Каталог FFS

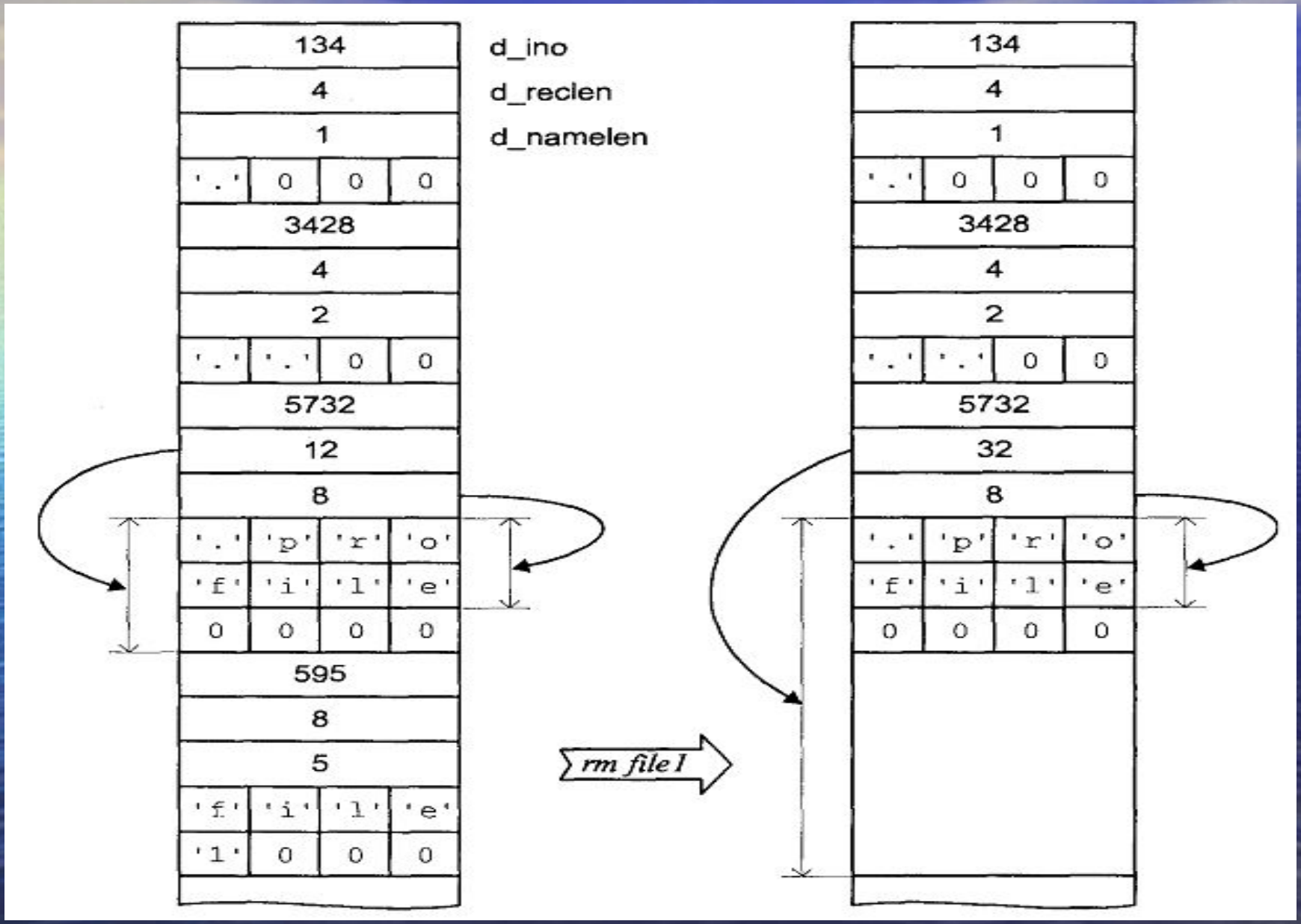
Поддерживает имя файла переменной длины до 255 символов

Запись каталога

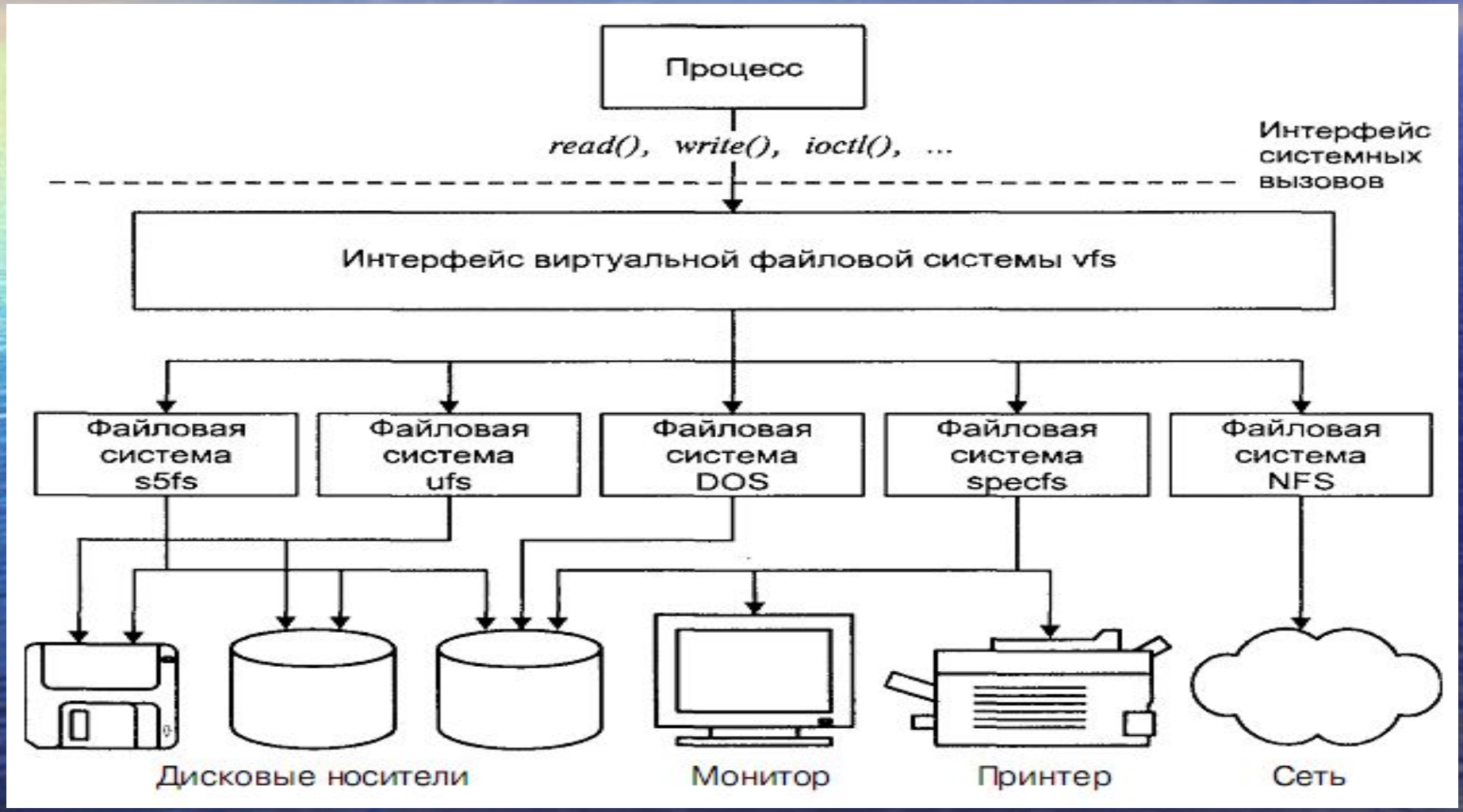
- d_ino – номер inode
- d_reclen – длина записи
- d_namlen – длина имени
- d_name[] – имя файла

Имя файла дополняется нулями до 4-х байтовой

Удаление файла



Архитектура виртуальной ФС



Виртуальный inode (vnode)

Интерфейс разработан в 1984 для ufs и NFS Sun Microsystems

Позволяет работать с любыми типами файловых систем даже FAT16

Виртуальный inode (vnode)

Метаданные всех открытых файлов представлены в памяти в виде in-core inode (vnode)

Структура vnode одинакова для всех файлов, независимо от типа ФС

Виртуальный inode (vnode)

Поле		Описание
u short	vflag	Флаги vnode
u short	v count	Число ссылок на vnode
struct filock	*v filocks	Блокировки файла
struct vfs	*v_vfsmountedhere	Указатель на подключенную файловую систему, если vnode является точкой монтирования
struct vfs	*v_vfsp	Указатель на файловую систему, в которой находится файл
enum vtype	v_type	Тип vnode: обычный файл, каталог, специальный файл устройства, символическая связь, сокет
caddr t	v data	Указатель на данные, относящиеся к реальной файловой системе
struct vnodeops	*v op	Операции vnode

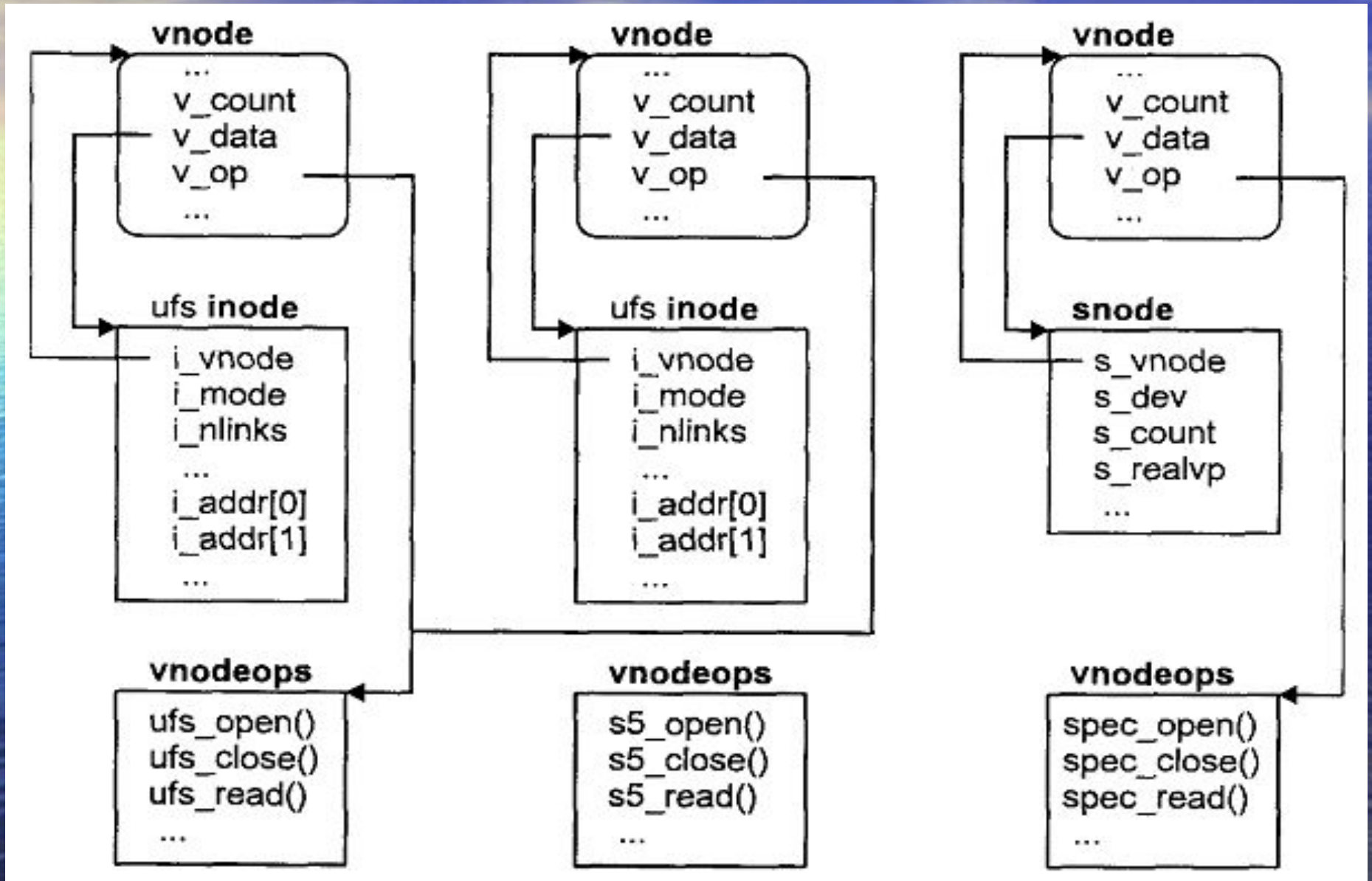
Операции с vnode (1)

int	(*vn_open) ()	Открыть vnode. Если операция предусматривает создание клона (размножение), то в результате будет размещен новый vnode. Обычно операции такого типа характерны для специальных файлов устройств.
int	*vn close) ()	Закрыть vnode.
int	(*vn read) ( )	Чтение данных файла, адресованного vnode.
int	(*vn write) ()	Запись в файл, адресованный vnode.
int	(*vn ioctl) ()	Задание управляющей команды.
int	*vn getaddr) ()	Получить атрибуты vnode: тип vnode, права доступа, владелец-пользователь, владелец-группа, идентификатор файловой системы, номер inode, число связей, размер файла, оптимальный размер блока для операций ввода/вывода, время последнего доступа, время последней модификации, время последней модификации vnode, число занимаемых блоков.
int	(*vn setaddr) ()	Установить атрибуты vnode. Могут быть изменены UID, GID, размер файла и времена доступа и модификации.
int	(*vn	Проверить права доступа к файлу, адресованному vnode. При этом производится отображение между атрибутами доступа файлов UNIX и атрибутами реальной файловой системы (например, DOS).
int	(*vn_lookup) ()	Произвести трансляцию имени файла в соответствующий ему vnode.
int	(*vn create) ()	Создать новый файл и соответствующий ему vnode.

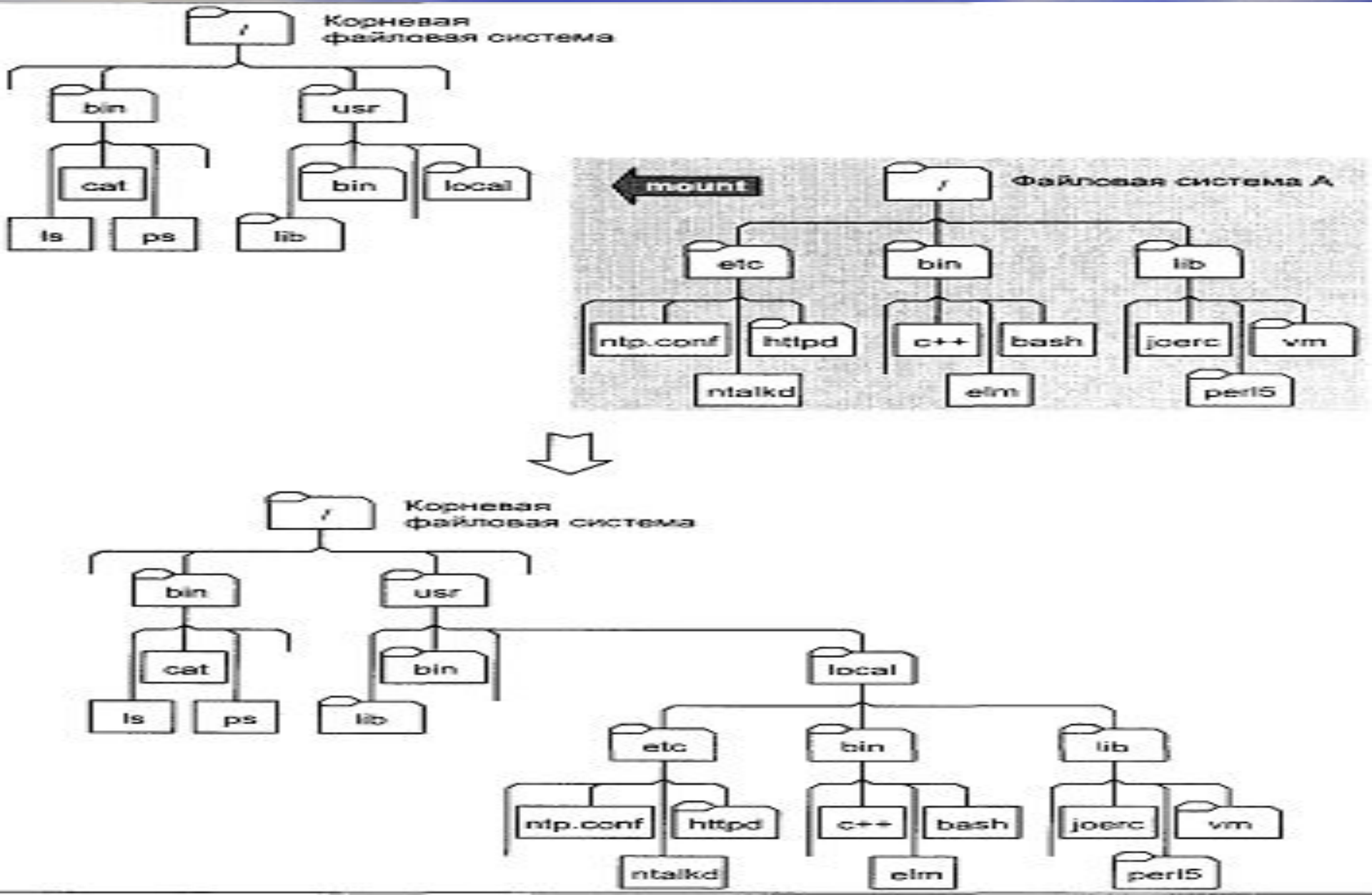
Операции с vnode (2)

int	(*vn_link)()	Создать жесткую связь между именем файла и vnode.
int	(*vn mkdir)()	Создать новый каталог в указанном vnode каталоге.
int	(*vn rmdir)()	Удалить каталог.
int	(*vn readdir)()	Считать записи каталога, адресованного vnode.
int	(*vn symlink)()	Создать символическую связь между новым именем и именем файла, расположенном в указанном vnode каталоге.
int	(*vn readlink)()	Чтение файла — символической связи.
int	(*vn_fsync)()	Синхронизировать содержимое файла — записать все кэшированные данные.
int	(*vn inactive)()	Разрешить удаление vnode, т. к. число ссылок на vnode из виртуальной файловой системы стало равным нулю.

Метаданные файла VFS



Монтирование ФС (подключение)



Структура VFS

struct vfs	*vfs next	Следующая файловая система в списке монтирования.
struct vfsops	*vfs_op	Операции файловой системы.
struct vnode	*vfs vnodecovered	vnode, перекрываемый файловой системой.
int	vfs_flag	Флаги: только для чтения, запрещен бит SUID и т. д.
int	vfs bsize	Размер блока файловой системы.
caddr t	vfs data	Указатель на специфические данные, относящиеся к реальной файловой системе.

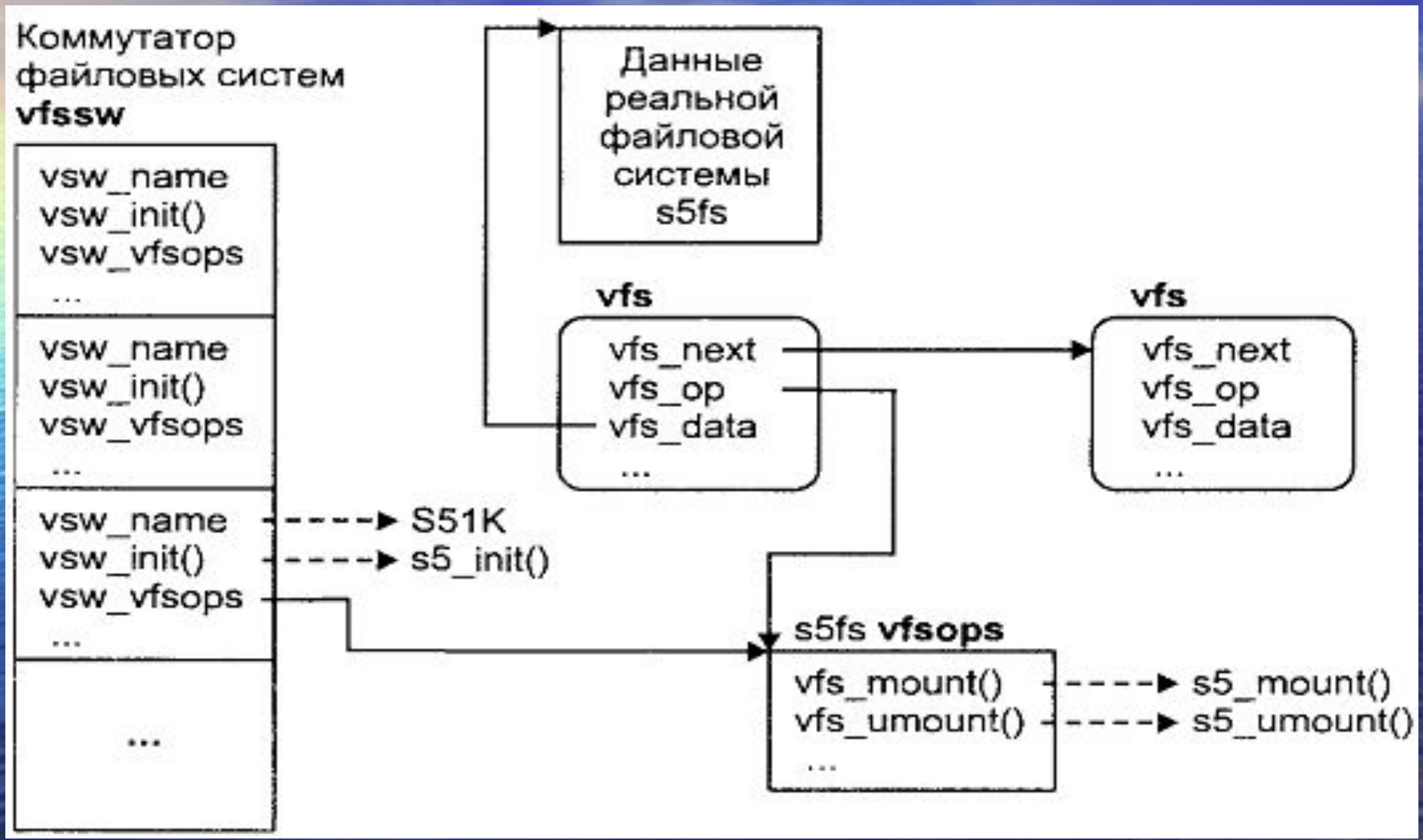
Операции ФС

int	*vfs mount)	Подключает файловую систему. Обычно операция включает размещение суперблока в памяти и инициализацию записи в таблице монтирования.
int	(*vfs unmount)()	Отключает файловую систему. Операция включает актуализацию данных файловой системы на накопителе (например, синхронизацию дискового суперблока и его образа в памяти).
int	(*vfs_root)()	Возвращает корневой vnode файловой системы.
int	(*vfs statfs)()	Возвращает общую информацию о файловой системе, в частности: размер блока хранения данных, число блоков, число свободных блоков, число inode.
int	(*vfs_sync)()	Актуализирует все кэшированные данные файловой системы.
int	(*vfs_fid)()	Возвращает *файловый идентификатор* (fid — file Identifier), однозначно адресующий файл в данной файловой системе. В качестве fid может, например, выступать номер inode реальной файловой системы.
int	(*vfs_vget)()	Возвращает указатель на vnode для файла данной файловой системы, адресованного fid.

Коммутатор ФС vfssw

char	*vsw name	Имя типа файловой системы
int	(*vsw init) ()	Адрес процедуры инициализации
struct vfsops	*vsw vfsops	Указатель на вектор операций файловой системы
long	vsw flag	Флаги

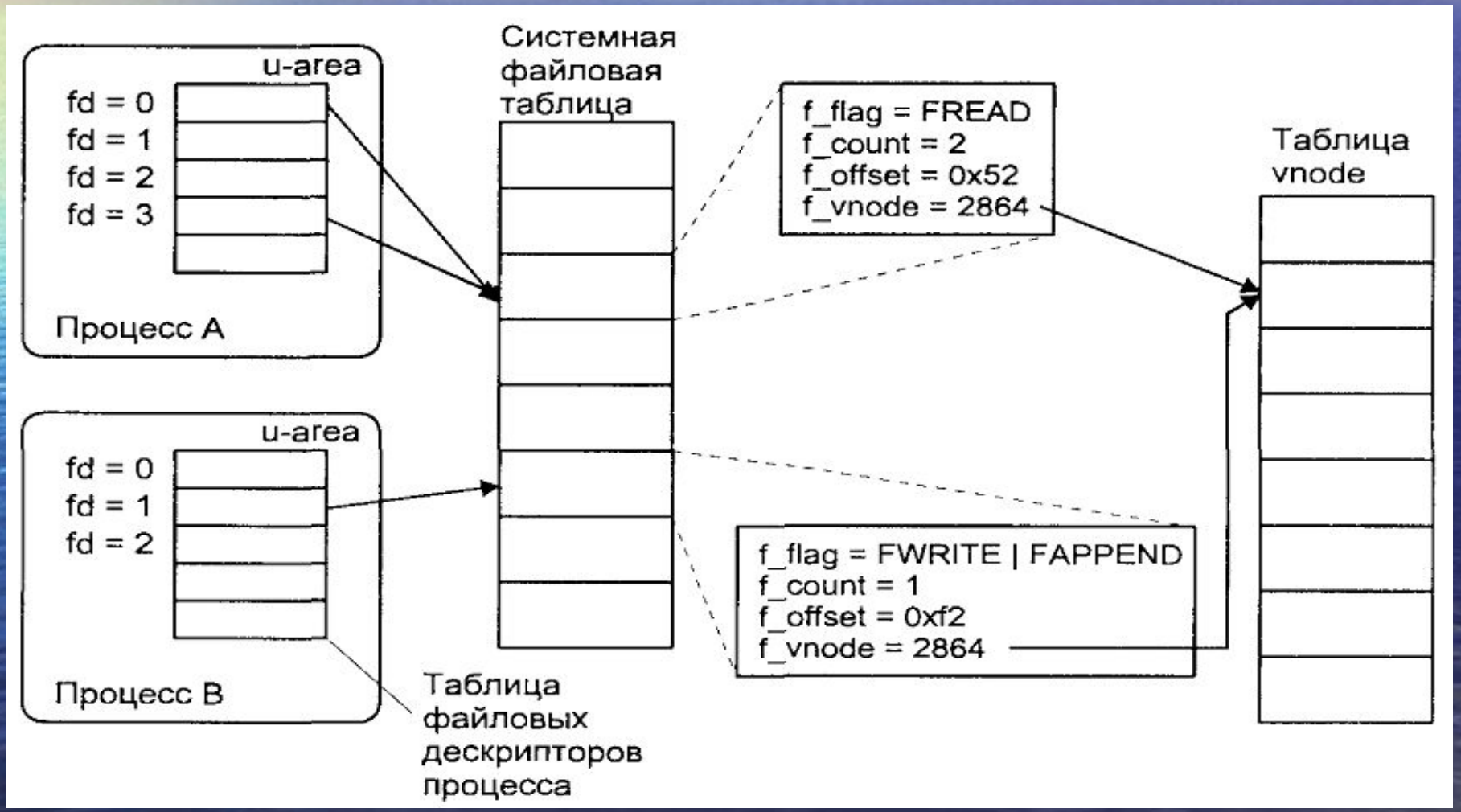
Структуры данных vfs



Действия при монтировании ФС

- Проверка прав монтирования
- Размещение и инициализация специфических данных конкретной ФС (vfs_data)
- Размещение vnode для корневого каталога ФС (vfs_root)

Доступ к файловой системе



Файловый дескриптор

Таблица дескрипторов хранится в u-area. С дескриптором связан указатель на элемент системной таблицы открытых файлов и флаг CLOSE_ON_EXEC. Размер таблицы определяет максимальное число открытых данным процессом файлов

Стандартные дескрипторы

- 0 – стандартный ввод
- 1 – стандартные вывод
- 2 – стандартный протокол

Системная таблица открытых файлов

При каждом новом открытии файла (open(), creat()) выделяется новый элемент. Размер таблицы определяет максимальное число открытых файлов в системе

Элемент таблицы

- Количество ссылок
- Флаги (режим открытия, SYNC и т.п.)
- Файловый указатель
- Ссылка на vnode данного файла (vnode всегда хранится в единственном экземпляре)

Открытие (создание) файла

Системные вызовы:

int open(const char *, int, int);

int creat(const char *, int);

Обязательно проверка прав доступа

Флаги (не все)

- O_RDONLY
- O_WROLY
- O_RDWR
- O_CREAT
- O_TRUNC
- O_EXCL
- O_NONBLOCK

Права доступа к новому файлу

Третий параметр open (второй параметр creat) & ! umask

0777 & ! 022 = 0755

0777 & ! 0777 = 0000

0777 & ! 0 = 0777

Параметры creat

Системный вызов creat эквивалентен open с флагами:

O_WRONLY | O_CREAT | O_TRUNC

Стандартные операции (1)

int read(int, char *, int);

чтение из файла

int write(int, const char *, int);

запись в файл

Права доступа не проверяются

Стандартные операции (2)

int lseek(int, int, int);

позиционирование файлового указателя

3-й параметр

SEEK_SET

SEEK_CUR

SEEK_END

Стандартные операции (3)

int close(int);

закрытие файлового дескриптора

Последовательно проходим по ссылкам и освобождаем только те элементы, где число ссылок станет равным нулю

Стандартные операции (4)

int dup(int);

int dup2(int, int);

дублирование файлового дескриптора

Стандартные операции (5)

int fcntl(int, int, …);

различные нестандартные операции с файловым дескриптором

Временные параметры

- atime
- mtime
- ctime

Разреженный файл (файл с дырой)

Последовательность:

- write
- lseek
- write

Дыра логически представлена массивом нулей, физически блоки не выделены

Буферный кэш

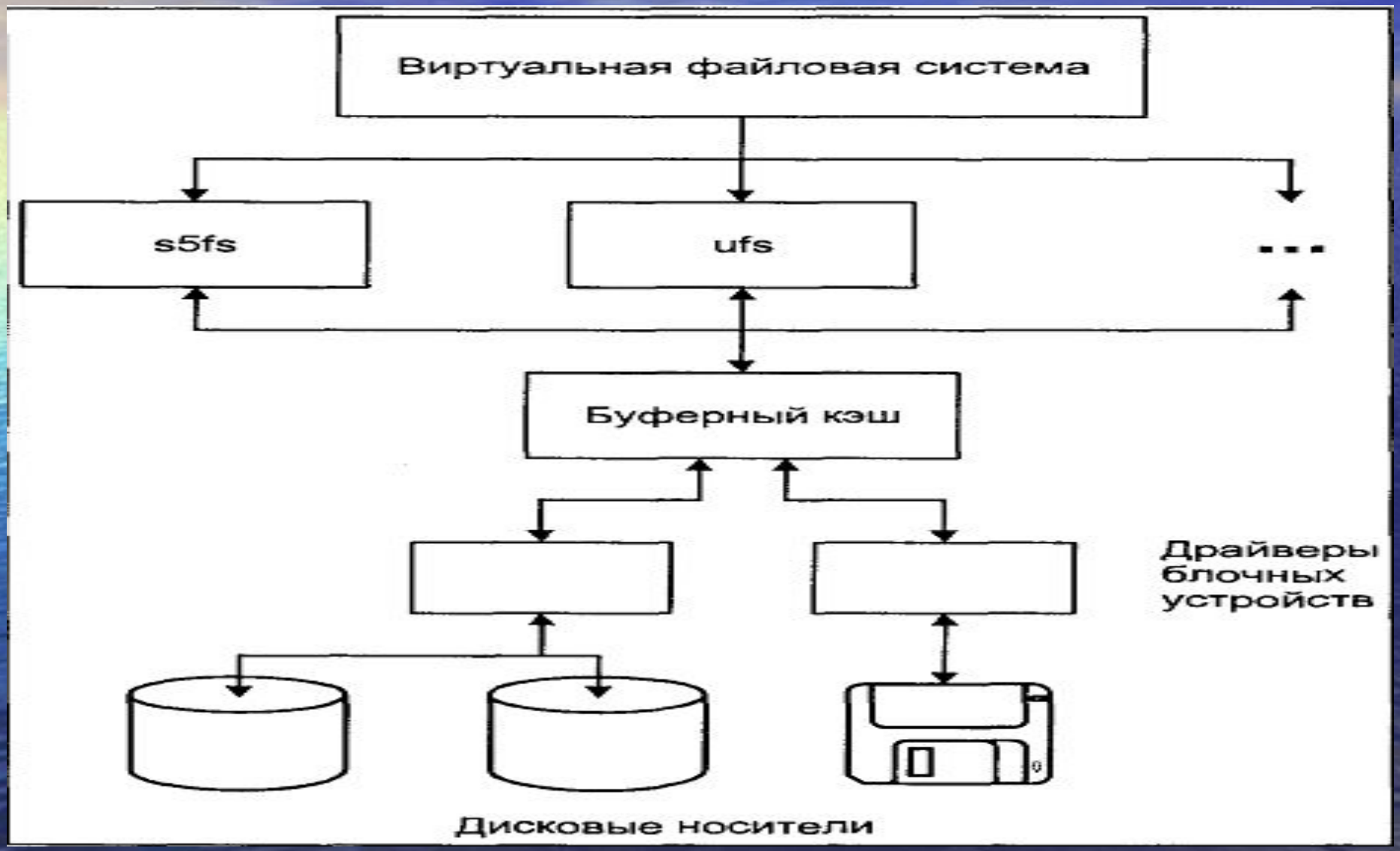
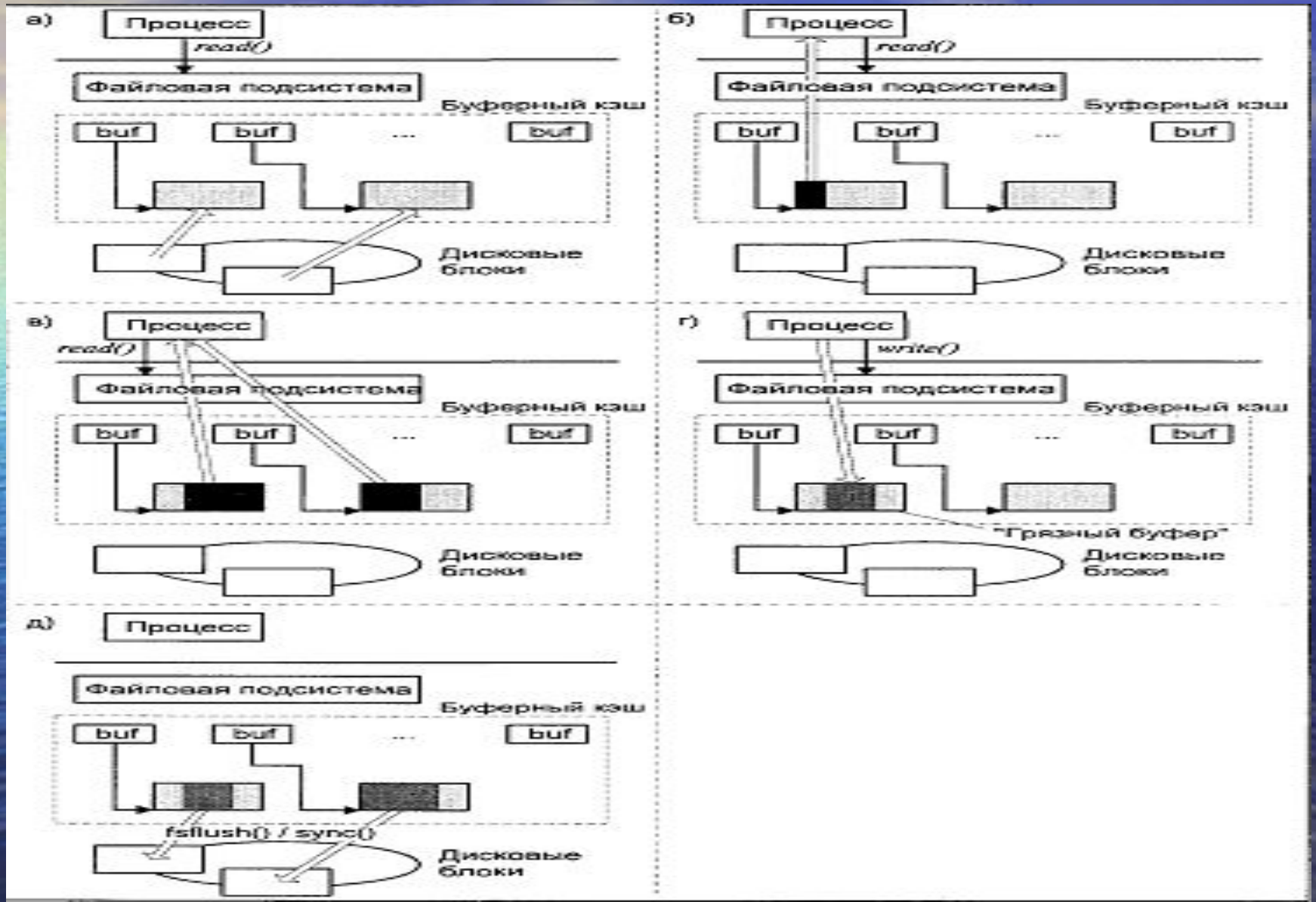


Схема работы



Обновление «грязных» буферов

- Системный вызов sync()
- Команда sync
- Последнее закрытие файла
- Диспетчер буферного кэша

В некоторых версиях UNIX не используется буферный кэш. Для оптимизации работы файловой подсистемы все открытые файлы отображаются в адресное пространство процесса. Подсистема управления памятью обрабатывает страничные ошибки

Целостность файловой системы

Нарушения, связанные с

- содержимым файла могут быть опасны для пользователей и для системы (если системный файл)
- метаданными файла всегда опасны для системы

Возможные ошибки ФС