Лекция 3. Виртуальная файловая система POSIX

CS221. Архитектура компьютера и операционные системы

3 октября 2017 г.

Определения

Виртуальная ФС

Виртуальная файловая система: (Virtual file system) — абстрактный интерфейс прикладных программ над конкретными файловыми системами. Позволяет работать унифицированно с различными файловыми системами и другими наборами данных. SunOS (1985), System V Release 4, Windows NT, ...

Канал: (pipe) — средство межпроцессного взаимодействия для одностороннего обмена данными между двумя процессами (FIFO). В POSIX:

- именованные;
- неименованные.

Идентификатор пользователя/группы: (user ID, group ID) — целое число от 0 до 32 767 (изначально) для идентификации ядром пользователя/группы.

Определения

Виртуальная ФС

Виртуальная файловая система: (Virtual file system) — абстрактный интерфейс прикладных программ над конкретными файловыми системами. Позволяет работать унифицированно с различными файловыми системами и другими наборами данных. SunOS (1985), System V Release 4, Windows NT, ...

Канал: (pipe) — средство межпроцессного взаимодействия для одностороннего обмена данными между двумя процессами (FIFO). В POSIX:

- именованные;
- неименованные.

Идентификатор пользователя/группы: (user ID, group ID) — целое число от 0 до 32 767 (изначально) для идентификации ядром пользователя/группы.

Определения

Виртуальная ФС

Виртуальная файловая система: (Virtual file system) — абстрактный интерфейс прикладных программ над конкретными файловыми системами. Позволяет работать унифицированно с различными файловыми системами и другими наборами данных. SunOS (1985), System V Release 4, Windows NT, ...

Канал: (pipe) — средство межпроцессного взаимодействия для одностороннего обмена данными между двумя процессами (FIFO). В POSIX:

- именованные;
- неименованные.

Идентификатор пользователя/группы: (user ID, group ID) — целое число от 0 до 32 767 (изначально) для идентификации ядром пользователя/группы.

Файловая система POSIX

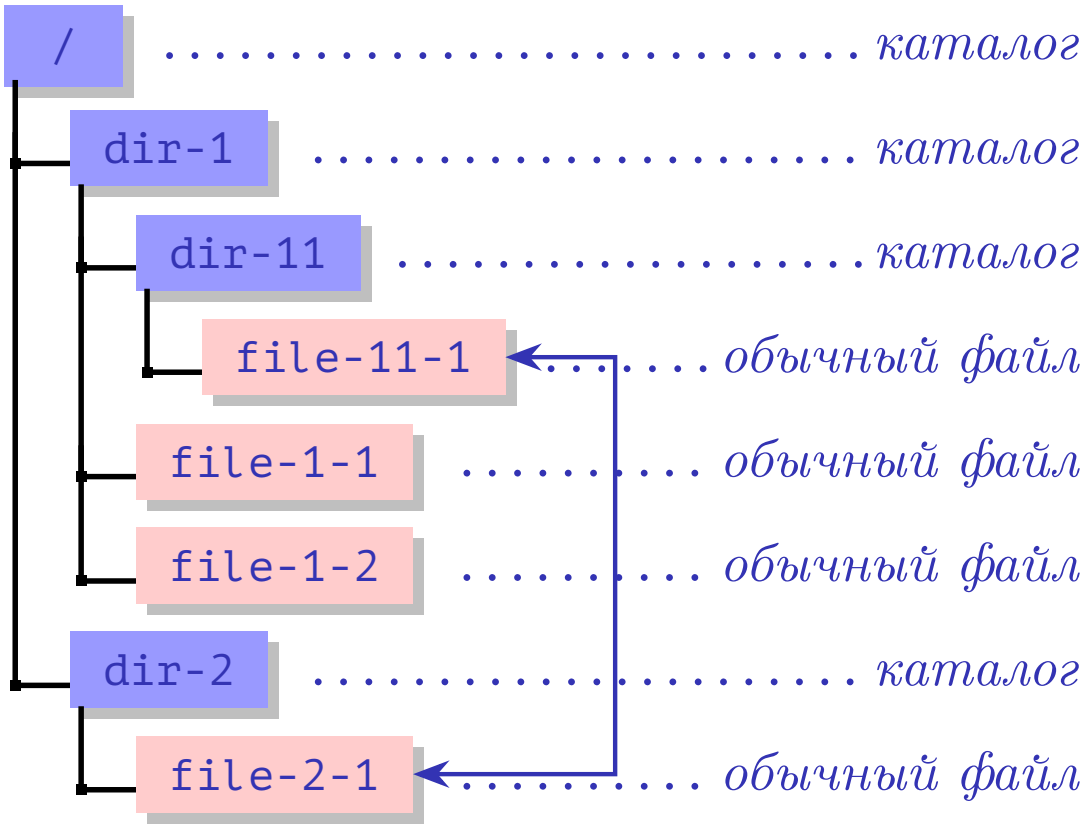

Рис. 1: структура POSIX-совместимой виртуальной файловой системы

Особенности файловой системы

Особенности виртуальной файловой системы POSIX

- Структура — ациклический граф.
- Единственный источник: /.
- Каталоги рассматриваются как специальные файлы.
- Имя файла может состоять из любых символов некоторого набора (например, Unicode) кроме “/” и “\0”.
- Имена файлов чувствительны к регистру букв.
- Специальные каталоги “.” (ссылка на текущий) и “..” (ссылка на каталог выше/корень) принадлежат каждому каталогу.

Представления путей

Определение

Абсолютный путь: (absolute path) — начинается с “/”.

Пример

`/home/student/.bash_profile`

Определение

Относительный путь: (relative path) — начинается с любого символа, кроме “/”.

Пример

`../user2/.ssh/id_rsa.pub`

Физические файловые системы

Некоторые часто используемые ФС

ext4: разработана специально для ядра Linux. Является потомком более старых версий ext3, ext2 и ext.

NTFS: используется в качестве основной для Windows NT. Имеет поддержку хранения дескрипторов безопасности.

FAT: использовалась в MS-DOS, позже — в Windows 3.11, Windows 95 и т. д. Используется в портативных устройствах и съёмных носителях. Не имеет поддержки разграничения прав доступа. Последней версией является FAT32, (размер файла $\leqslant$ 4 ГБ).

ISO 9660 (или CDFS): используется для хранения файлов на носителях CD-ROM, DVD-ROM и т. д. (размер файла $\leqslant$ 4 ГБ).

UDF: разработана для носителей CD-RW, DVD и т. д. В настояще время имеет поддержку жёстких дисков и носителей на Flash-памяти.

Монтирование физической файловой системы

Определения

Монтирование: (mounting) — процедура подготовки отображения данных физической файловой системы в каталог виртуальной.

Точка монтирования: (mount point) — каталог, используемый для отображения физической файловой системы.

Пример (Linux)

- Корневой каталог монтируется во время загрузки ОС.
- Некоторые другие монтируются позже во время загрузки на основе `/etc/fstab`.

Специальные файловые системы

Определение

Специальная файловая система: (special file system) — предоставляет доступ к данным, не содержащимся на каких-либо дисковых носителях.

Название	Точка	Описание
`pipefs`	—	Каналы.
`sockfs`	—	Сокеты.
`shm`	—	Общая память.
`devfs`	`/dev`	Виртуальный файл устройства.
`proc`	`/proc`	Информация о процессах и других структурах ядра.
`sysfs`	`/sys`	Информация о подсистемах ядра, устройствах и драйверах.

Таблица 1: примеры специальных файловых систем в ОС Linux

Ссылки

Определение

Ссылка: (жёсткая ссылка, hard link) — одно из имён файла, принадлежащее каталогу.

Команда POSIX

`ln` ⟨*существующий путь*⟩ ⟨*новый путь*⟩

Ограничения

- Нельзя создавать ссылки на каталог.
- Ссылки возможны в пределах одной файловой системы.

Ссылки (окончание)

Определение

Символьная ссылка: (гибкая ссылка, symbolic link) — файл, содержащий произвольный путь (в т. ч. несуществующий).

Команда POSIX

`ln -s` ⟨*упоминемый путь*⟩ ⟨*путь к ссылке*⟩

Типы файлов виртуальной файловой системы

Определения

Файл: (*обычный файл*) отражает содержимое физического файла.

Каталог: объединяет несколько файлов, включая другие каталоги.

Символьная ссылка: хранит путь к файлу.

Файл устройства: (*device file*) интерфейс для драйвера устройства (`devfs`).

Именованный канал: (*named pipe*) канал, который при открытии идентифицируется с помощью пути в виртуальной файловой системе (`pipefs`).

Сокет: (*socket*, *сокет домена Unix*) средство полнодуплексного межпроцессного взаимодействия (`sockfs`).

Виды файлов устройств

Определения

Символьные устройства: предоставляют небуферизованный доступ к физическим устройствам. Чтение/запись осуществляются порциями данных, кратными размерам блоков, которые поддерживает конкретное устройство.

Блочные устройства: предоставляют буферизованный доступ к устройствам, поддерживаемые размеры физических блоков могут быть не видны на уровне пользователя. Чтение/запись может выполняться порциями данных любых размеров, конкретный момент попадания данных на физическое устройство неизвестен, как и порядок операций записи.

Псевдо-устройства: не соответствуют каким-либо конкретным физическим устройствам. Предоставляют некоторые возможности операционной системы. (`/dev/null`, `/dev/random`).

Классы пользователей

Классы пользователей

Владелец: (owner) — владелец файла.

Группа: (group) — набор пользователей (владелец не обязательно $\in$ группе).

Остальные: (others) — $\forall$ пользователи, $\neq$ владельцу и $\notin$ группе.

Предупреждение

Разрешения для файлов/подкаталогов внутри каталогов не наследуются от родительских каталогов.

Права классов пользователей

Право	Файл	Каталог
Чтение	Чтение	Чтение имён файлов
Запись	Изменение содержимого	Переименование файлов, создание новых, удаление $\exists$ (если $\nexists$ исполнение $\Rightarrow$ право игнорируется).
Исполнение	Запуск исполняемого файла или файла сценария	Возможность выбора каталога в качестве текущего и получения доступа к файлам внутри.

Таблица 2: права классов пользователей

Атрибуты файлов

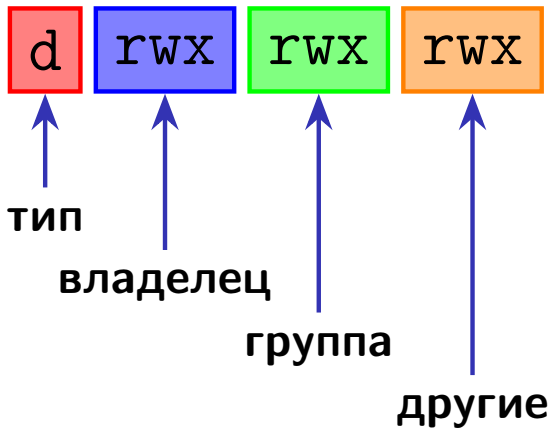

Рис. 2: значения атрибутов файла

Тип	Значение
-	обычный файл
d	каталог
l	ссылка
c	символьное устройство
b	блочное устройство
p	именованный канал
s	сокет

Таблица 3: значения атрибута типа файла

Примеры

Примеры

Атрибуты	Права
drwx------	можно всё.
dr-x------	
d-wx------	
d--x------	

Примеры

Примеры

Атрибуты	Права
`drwx------`	можно всё.
`dr-x------`	нельзя создавать, удалять, переименовывать файлы.
`d-wx------`	
`d--x------`	

Примеры

Примеры

Атрибуты	Права
`drwx------`	можно всё.
`dr-x------`	нельзя создавать, удалять, переименовывать файлы.
`d-wx------`	нельзя узнавать список файлов в каталоге.
`d--x------`	

Примеры

Примеры

Атрибуты	Права
`drwx------`	можно всё.
`dr-x------`	нельзя создавать, удалять, переименовывать файлы.
`d-wx------`	нельзя узнавать список файлов в каталоге.
`d--x------`	нельзя создавать удалять, переименовывать файлы, узнавать список файлов.

Запуск сценариев

Пример (script)

```
#! /bin/bash

echo "Hello world"
```

Пример

```
$ bash script
```

Пример

```
$ ./script
```

Файловые структуры

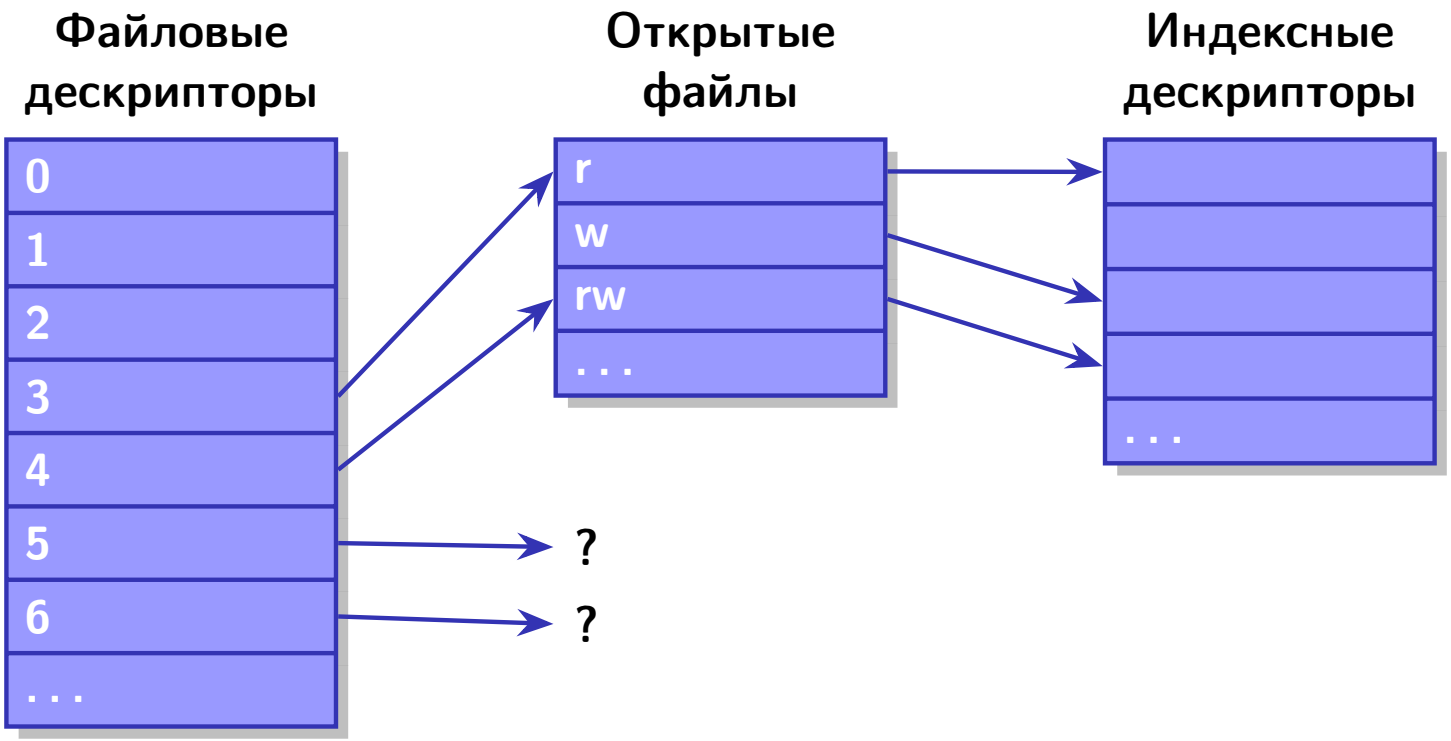

Рис. 3: таблицы файловых структур

Индексный дескриптор (inode)

Состав индексного дескриптора

- Идентификатор устройства;
- права доступа и тип (+ возможно, другие данные доступа);
- Количество жёстких ссылок;
- Идентификатор пользователя-владельца;
- Идентификатор группы-владельца;
- Длина в байтах;
- Отметки времени:
  - изменения состояния индексного дескриптора;
  - последнего обращения к файлу;
  - последнего изменения файла.

Открытый файл (file)

Состав открытого файла

- Режим;
- Флаги при открытии;
- Позиция;
- Ссылка на процесс-владелец;
- Ссылка на inode;
- Таблица файловых операций;
- ...

Открытие/создание файла

Функция open() (<sys/types.h>, <sys/stat.h>, <fcntl.h>)

```
int open(const char *pathname, int flags);    // nFD, -1
int open(const char *pathname, int flags, mode_t mode);
```

O_RDONLY	O_WRONLY	O_RDWR	O_TRUNC
O_APPEND	O_CREAT	O_EXCL	

Таблица 4: основные флаги режима открытия

S_IRUSR	S_IWUSR	S_IXUSR	S_IRWXU
S_IRGRP	S_IWGRP	S_IXGRP	S_IRWXG
S_IROTH	S_IWOTH	S_IXOTH	S_IRWXO

Таблица 5: флаги разрешений (mode_t, <sys/stat.h>)

Получение информации о файле

Функция fstat() (<sys/stat.h>)

```
int fstat(int nFD, struct stat *pBuf);    // 0, -1
```

Структура stat

```
struct stat
{
  dev_t     st_dev;
  ino_t     st_ino;
  mode_t    st_mode;
  nlink_t   st_nlink;
  uid_t     st_uid;
  gid_t     st_gid;
```

Структура stat (окончание)

```
  dev_t     st_rdev;
  off_t     st_size;
  blksize_t st_blksize;
  blkcnt_t  st_blocks;
  time_t    st_atim;
  time_t    st_mtim;
  time_t    st_ctim;
};
```

Операции с открытыми файлами

Функции read(), write(), close() (<unistd.h>)

```
ssize_t read(int nFD, void *pvBuf, size_t uCount);    // size, -1
ssize_t write(int nFD, const void *pcvBuf, size_t uCount);
int close(int nFD);    // 0, -1
```

Функция lseek() (<sys/types.h>, <unistd.h>)

```
off_t lseek(int nFD, off_t uOffset, int whence);    // off, (off_t) -1
```

SEEK_SET	SEEK_CUR	SEEK_END

Таблица 6: флаги направления смещения

Операции с индексными дескрипторами

Функция mkdir() (<sys/stat.h>, <sys/types.h>)

```
int mkdir(const char *pathname, mode_t mode);     // 0, -1
```

Функции rmdir(), link(), unlink(), rename(), symlink() (<unistd.h>)

```
int rmdir(const char *pathname);     // 0, -1
int link(const char *oldpath, const char *newpath);
int unlink(const char *pathname);
int rename(const char *oldpath, const char *newpath);
int symlink(const char *topath, const char *frompath);
```

Пример работы с файлами

Пример

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

#include <stdio.h>
#include <stdlib.h>
```

Пример (продолжение)

```
struct data
{
  int m_nNum;
  char m_szName[20];
};

int main()
{
  ssize_t nSize;
  off_t nOffset;
  struct data d1 = { 10, "Something" }, d2;
```

Пример работы с файлами (продолжение)

Пример (продолжение)

```
int nFD = open(
  "files.bin",
  O_RDWR | O_CREAT | O_TRUNC,
  S_IRUSR | S_IWUSR);
if (nFD == -1)
{
  perror("open");
  return EXIT_FAILURE;
}
```

Пример работы с файлами (продолжение)

Пример (продолжение)

```
nSize = write(nFD, &d1, sizeof (d1));
if (nSize < sizeof (d1))
{
  if (nSize == -1)
    perror("write");
  fprintf(stderr, "Could not write all bytes\n");
  close(nFD);
  return EXIT_FAILURE;
}
```

Пример работы с файлами (продолжение)

Пример (продолжение)

```
nOffset = lseek(nFD, 0, SEEK_SET);
if (nOffset == -1)
{
  perror("lseek");
  close(nFD);
  return EXIT_FAILURE;
}
```

Пример работы с файлами (окончание)

Пример (окончание)

```
  nSize = read(nFD, &d2, sizeof (d2));
  if (nSize < sizeof (d2))
  {
    if (nSize == -1)
      perror("read");
    fprintf(stderr, "Could not read all bytes\n");
    close(nFD);
    return EXIT_FAILURE;
  }
  printf("Num:  %d\nName: %s\n", d2.m_nNum, d2.m_szName);
  close(nFD);
  return EXIT_SUCCESS;
}    // main()
```

Отображение файла в память

Функции mmap(), munmap() (<sys/mman.h>)

```
void *mmap(
  void *pvStart, size_t uLength, int prot,
  int flags, int nFD, off_t nOffset);    // addr, MAP_FAILED

int munmap(void *pvStart, size_t uLength);    // 0, -1
```

PROT_READ	PROT_WRITE	PROT_EXEC	PROT_NONE

Таблица 7: флаги режима доступа (<sys/mman.h>)

MAP_SHARED	MAP_PRIVATE	MAP_FIXED

Таблица 8: флаги открытия (<sys/mman.h>)

Пример отображения файла в память

Пример

```
#include <sys/mman.h>
#include <fcntl.h>
#include <unistd.h>

#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
```

Пример (продолжение)

```
struct data
{
  int m_nNum;
  char m_szData[20];
};

int main()
{
  const size_t cuSize =
    sizeof (struct data [10]);
  int i;
  struct data *pData;
```

Пример отображения файла в память (продолжение)

Пример (продолжение)

```
int nFD = open(
  "mapped.bin",
  O_RDWR | O_CREAT | O_TRUNC,
  S_IRUSR | S_IWUSR);
if (nFD == -1)
{
  perror("open");
  return EXIT_FAILURE;
}
for (i = 0; i < cuSize; ++ i)
  write(nFD, &i, sizeof (char));
```

Пример (продолжение)

```
pData = (struct data *) mmap(
  NULL,
  cuSize,
  PROT_READ | PROT_WRITE,
  MAP_SHARED,
  nFD,
  0);
if (pData == MAP_FAILED)
{
  perror("mmap");
  close(nFD);
  return EXIT_FAILURE;
}
```

Пример отображения файла в память (окончание)

Пример (окончание)

```
  for (i = 0; i < 10; ++ i)
  {
    pData[i].m_nNum = i;
    sprintf(pData[i].m_szData, "%d", i);
  }
  munmap(pData, cuSize);
  close(nFD);
  return EXIT_SUCCESS;
}    // main()
```