Лабораторные работы по курсу
Информационные технологии 1. Операционные системы

Лабораторная работа 6
Файловые системы и RAID массивы

Москва, 2022

Оглавление

Введение

Все данные на любом запоминающем устройстве хранятся в виде последовательности байт. Для непосредственной работы с ними необходимо обращаться к регистрам устройства, искать адреса, по которым лежат данные и производить с ними взаимодействие. Делать все это вручную возможно, но очень сложно и неудобно. Поэтому была разработана абстракция в виде файловой системы, которая абстрагирует пользователя от устройства и предоставляет ему возможность работать с данными на более простом уровне.

1. Теоретическая часть

1.1. Файловые системы

Дадим основные определения:

Файл – поименованная совокупность данных

Файловая система – часть ОС, отвечающая за работу с файлами.

Можно выделить основные функции файловой системы:

1. Создание, удаление, модификация файлов
2. Разделение файлов друг от друга, поддержание целостности
3. Совместная работа нескольких процессов с файлами
4. Изменение структуры файла
5. Восстановление после стирания
6. Обеспечение разных методов доступа и режима секретности
7. Обращение к файлу по символическому имени
8. Дружественный интерфейс [1]

Файловая система позволяет при помощи системы справочников (каталогов, директорий) связать уникальное имя файла с блоками памяти, содержащими данные файла. Помимо собственно файлов и структур данных, используемых для управления файлами (каталоги, дескрипторы файлов, различные таблицы распределения внешней памяти), понятие "файловая система" включает программные средства, реализующие различные операции над файлами. [2]

Над файлами возможно производить стандартные операции – открытие, закрытие, чтение, запись, удаление и др.

1.2. Структура файловой системы

Файловые системы хранятся на дисках. Если посмотреть на схематичное изображение диска (Рисунок 1), то можно увидеть, что его возможно разбить на набор **секторов**, участки которого будут последовательно считываться. Несколько последовательных секторов возможно объединить в один **кластер**.

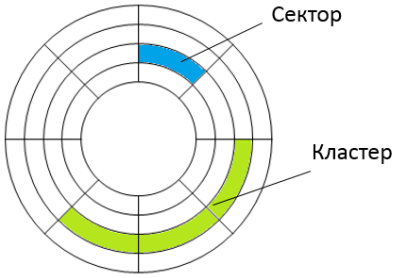

Рисунок 1 Структура диска

Большинство дисков может быть разбито на один или несколько разделов, на каждом из которых будет своя файловая система. Каждая файловая система будет содержать в себе участок, отвечающий за её загрузку и организацию содержимого в ней. Остальные участки памяти будут занимать непосредственно файлы. [3]

1.3. Типы файловых систем

Существует множество различных типов файловых систем. У каждой из них свое устройство и предназначение. Рассмотрим наиболее популярные из них.

1.3.1. FAT

FAT (*File Allocation Table* «таблица размещения файлов») — классическая архитектура файловой системы, которая из-за своей простоты всё ещё широко применяется для флеш-накопителей. Используется в дискетах, картах памяти и некоторых других носителях информации. Ранее находила применение и на жёстких дисках.

В файловой системе FAT смежные секторы диска объединяются в единицы, называемые кластерами. Количество секторов в кластере равно степени двойки (см. далее). Для хранения данных файла отводится целое число кластеров (минимум один). Размер такого кластера стандартизирован и равен числу секторов в нем умноженному на число байт в таком секторе. Обычно, размер сектора равен 512 байтам. Пусть в одном кластере содержится 4 сектора, тогда его объем будет 2 Кб. Если мы захотим сохранить файл, объемом 1 байт, то произойдет выделение целого кластера размером 2 Кб, которой будет практически пуст.

Для определения в каких кластерах содержаться данные файловая система содержит специальную таблицу размещения файлов – FAT. В ней каждому файлу предоставляется цепочка из номеров кластеров, в котором он содержится.

Существует четыре версии FAT — FAT12, FAT16, FAT32 и exFAT (FAT64). Они отличаются разрядностью записей в таблице размещения файлов, то есть количеством бит, отведённых для хранения номера кластера. Например, у FAT16 один кластер будет обозначаться 2 байтами.

1.3.2. NTFS

NTFS (от англ. *new technology file system* — «файловая система новой технологии») — стандартная файловая система для семейства операционных систем Windows NT фирмы Microsoft. Как и любая другая система, NTFS делит все полезное место на кластеры — блоки данных, используемые единовременно. NTFS поддерживает почти любые размеры кластеров — от 512 байт до 64 Кбайт, неким стандартом же считается кластер размером 4 Кбайт.

Основная концепция системы состоит в том, что все данные в ней являются файлами, в том числе и служебные. Они могут располагаться в любом месте тома, вместе с обычными файлами. Ядром NTFS является таблица MFT, содержащая информацию обо всех каталогах

и файлах. Каждый из них представлен как минимум одной записью в этой таблицей, где указаны все их параметры и атрибуты. Дисковое пространство, как и в FAT выделяется кластерами.

1.3.3. ExtX

Во большинстве дистрибутивов Linux используется файловая система ExtX, где X – её номер. Сейчас наиболее популярной является система Ext4. Файловая система начинается с зарезервированной области, далее разбивается на несколько блоков. Блоки в свою очередь возможно объединить в секции. Базовая информация о строении файловой системы хранится в суперблоке, находящимся в самом её начале. Более подробно возможно изучить в [4]

1.4. RAID массивы

RAID расшифровывается как Redundant Array of Inexpensive Disks или избыточный массив недорогих дисков. RAID позволяет превратить несколько физических жестких дисков в один логический жесткий диск. Существует множество уровней RAID, таких как RAID 0, RAID 1, RAID 5, RAID 10 и т. д. Рассмотрим RAID 0. Основная идея заключается в разбиении всех блоков на два диска. Таким образом, в случае повреждения первого диска, второй останется целым и половина информации сохранится (Рисунок 2).

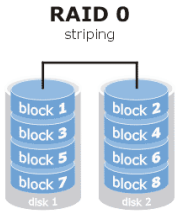

Рисунок 2 RAID 0

RAID 1, который также известен как зеркалирование дисков. RAID 1 создает идентичные копии данных. Если у вас два жестких диска объединенные в RAID 1, данные будут записаны на оба диска, т.е. на обоих дисках будут храниться одинаковые данные. Преимущество RAID 1 заключается в том, что если один из них выйдет из строя, то ваш компьютер или сервер все равно будет работать, потому что у вас есть полная неповрежденная копия данных на другом жестком диске. Вы можете вытащить неисправный жесткий диск во время работы компьютера, вставить новый жесткий диск, и он автоматически восстановит зеркало. Обратной стороной RAID 1 является отсутствие дополнительного дискового пространства. Если ваши два жестких диска имеют размер 1 ТБ, то общий полезный объем составляет 1 ТБ вместо 2 ТБ. (Рисунок 3)

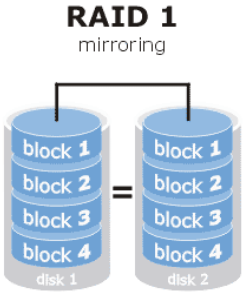

Рисунок 3 RAID 1

RAID 10 (RAID 1+0) — зеркалированный массив, данные в котором записываются последовательно на несколько дисков, как в RAID 0. Эта архитектура представляет собой массив типа RAID 0, сегментами которого вместо отдельных дисков являются массивы RAID

1. Соответственно, массив этого уровня должен содержать как минимум 4 диска (и всегда чётное количество). RAID 10 объединяет в себе высокую отказоустойчивость и производительность.

Существует два типа реализации RAID – **аппаратная** и **программная**. В случае аппаратной реализации используется контроллер жесткого диска для его создания. Контроллер жесткого диска — это карта PCIe, которую вы вставляете в компьютер. Затем вы подключаете свои жесткие диски к этой карте. При загрузке компьютера вы увидите параметр, позволяющий настроить RAID. Вы можете установить операционную систему поверх аппаратного RAID, что может увеличить время безотказной работы. Программный RAID требует, чтобы у вас уже была установлена операционная система. С одной стороны, он ничего не стоит (в отличие от аппаратных RAID-контроллеров). С другой стороны, программный RAID использует некоторое количество ресурсов центрального процессора.

2. Практическая часть

В данной лабораторной работе вам предстоит научиться создавать программный RAID массив и исследовать файловую систему FAT16. Все задания выполняются последовательно, а самостоятельные упражнения выделены *зеленым цветом*. Все результаты работы необходимо заносить в отчет.

Перед тем, как приступить к практическим заданиям, необходимо установить следующие программные компоненты

```
sudo apt-get install mdadm -y
sudo apt-get install hexedit -y
```

2.1. Создание программного RAID10 массива.

Для создания RAID10 массива создадим bash скрипт с кодом, приведенным ниже.

```
#!/bin/bash

if [ $EUID -ne 0 ]
        then
        echo -e "\033[31m Please, use sudo"
        exit
fi
declare -a loopname
for i in {0..3}
do
        loopname[i]=$(losetup -f)
        dd if=/dev/zero of=test.bin.$i bs=1M count=10
        losetup ${loopname[i]} test.bin.$i
        sfdisk --force ${loopname[i]} < parts.txt
        partprobe ${loopname[i]}
done
mdadm --create /dev/md10 --level 10 --raid-devices=4 ${loopname[@]}p1
mdadm -D /dev/md10
mkfs.ext4 /dev/md10
mkdir mnt
mount /dev/md10 mnt
echo "Hello MIET" > 1.txt
cp 1.txt mnt
cat mnt/1.txt
cat /proc/mdstat
echo -e "\033[32m Press Enter to continue"
tput sgr0
read
mdadm /dev/md10 -f ${loopname[3]}p1
cat /proc/mdstat
mdadm /dev/md10 -r ${loopname[3]}p1
cat /proc/mdstat
cat mnt/1.txt
echo -e "\033[32m Press Enter to continue"
tput sgr0
```

```
read
dd if=/dev/zero of=mnt/test bs=1M count=10 ; sync
mdadm /dev/md10 -a ${loopname[3]}p1
echo -e "\033[32m Press Enter to continue"
tput sgr0
read
cat /proc/mdstat
echo -e "\033[32m Press Enter to finish"
tput sgr0
read
umount mnt
rm -rf mnt
mdadm --stop /dev/md10
losetup -d ${loopname[@]}
rm -f test.bin.{0..3}
```

Расположим файл *parts.txt*, в той же директории, где расположен скрипт.

Содержимое файла:

```
unit: sectors
/dev/loop0p1 : start= 2048, size= 1046528, Id=fd
/dev/loop0p2 : start= 0, size= 0, Id= 0
/dev/loop0p3 : start= 0, size= 0, Id= 0
/dev/loop0p4 : start= 0, size= 0, Id= 0
EOF
```

Запустим данный скрипт с помощью следующей команды

```
sudo ./script
```

В данном скрипте происходит создание образа файловой системы, состоящей из 4 виртуальных жестких дисков. Далее образ монтируется в папку *mnt*. После этого создается файл 1.txt и копируется на смонтированный диск. Результат должен появиться на экране. В случае появления сообщения «*Continue creating array?*» нажмите «*Y*»

С помощью команды *cat /proc/mdstat* возможно посмотреть состояние RAID массива на текущий момент.

Результат ее работы будет приблизительно следующим:

```
Personalities : [raid10]
md10 : active raid10 loop0p1[4] loop3p1[3] loop2p1[2] loop1p1[1]
            14336 blocks super 1.2 512K chunks 2 near-copies [4/4] [UUUU]

unused devices: <none>
```

В данном случае видно, что был создан массив RAID10 и все 4 диска находятся в хорошем состоянии. За это отвечают символы [UUUU]

Нажмем на клавишу Enter и продолжим выполнение скрипта. С помощью следующих команд один из дисков был отмечен, как испорченный, а после удален.

```
mdadm: set /dev/loop0p1 faulty in /dev/md10
Personalities : [raid10]
md10 : active raid10 loop0p1[4](F) loop3p1[3] loop2p1[2] loop1p1[1]
          14336 blocks super 1.2 512K chunks 2 near-copies [4/3] [_UUU]

unused devices: <none>
mdadm: hot removed /dev/loop0p1 from /dev/md10
Personalities : [raid10]
md10 : active raid10 loop3p1[3] loop2p1[2] loop1p1[1]
          14336 blocks super 1.2 512K chunks 2 near-copies [4/3] [_UUU]
```

Как видно из вывода на экран, один из дисков исчез [_UUU]. Однако, сообщение текстовое сообщение все равно возможно прочитать, т.е. система продолжает работать.

Далее нажмем Enter и восстановим наш массив. Еще раз нажав клавишу, убедимся в том, что все диски появились обратно.

Изменив скрипт, создайте аналогичным образом RAID5. Заметили ли вы разницу при выполнении скриптов? Если да, то какую?

2.2. Работа с файловой системой FAT16

2.2.1. Создадим файл *img.fat*, который далее разметим файловой системой

```
dd if=/dev/zero of=img.fat bs=1024 count=10000
```

2.2.2. Разметим файл с помощью команды *mkfs*

```
mkfs -t vfat ./img.fat
```

2.2.3. Создадим папку, к которой будем монтировать созданную файловую систему.

```
mkdir 1
```

2.2.4. Монтируем файловую систему

```
sudo mount img.fat 1
```

2.2.5. С помощью следующего скрипта попытаемся скопировать в созданную директорию 600 файлов с текстом HelloWorld.

```
for i in {1..600}; do echo -n HelloWorld >> $i.text; sudo cp $i.text 1; rm $i.text; done
```

Проанализируйте результат выполнения команды. Сколько файлов удалось скопировать? Почему?

2.2.6. Удалите все созданные файлы и создайте директорию dir_1. Смонтируйте в нее файловую систему img.fat (см пункты 2.2.3-2.2.4)

2.2.7. Запустите следующий скрипт, создающий пустые файлы и копирующий их на диск.

```
for i in {1..600}; do touch $i; sudo cp $i 1; rm $i; done
```

Сколько файлов удалось скопировать? Попробуйте объяснить почему. Если не получится – продолжите выполнять задания дальше. Подсказка будет в одном из следующих пунктов.

2.2.8. Удалим созданный файл *img.fat* и директории. Далее создадим из заново, повторив пункты 2.2.1, 2.2.2, 2.2.3, 2.2.4.

2.2.9. Создадим внутри созданной директории несколько файлов различных размеров: 100 байт, 1000 байт, 10000 байт. Для этого возможно приведенным ниже скриптом, где N – число создаваемых байт/10.

```
for i in {1..N}; do echo -n 0123456789 >> 1.txt; done; sudo cp 1.txt 1
```

2.2.10. Перейдем в созданную директорию. Убедимся, что файлы были созданы. Сохраним результат вывода следующей команды в отчет.

```
ls -l --full-time
```

2.2.11. Демонтируем файловую систему

```
sudo umount 1
```

2.2.12. Перейдем в директорию 1 и убедимся, что созданные файлы исчезли.

2.2.13. Откроем файл *img.fat* с помощью 16ого редактора.

```
hexedit img.fat
```

Схематично, файловую систему возможно представить следующим образом (Рисунок 4):

Рисунок 4 Файловая система FAT16

По мере выполнения заданий, заполните пропущенные значения в Таблица 1:

Таблица 1 Адреса файловой системы

Файловая система	Диапазон адресов:
Загрузочный сектор	0x00000000 – 0x????????
FAT Копия 1	0x???????? – 0x????????
FAT Копия 1	0x???????? – 0x????????
Корневой каталог	0x???????? – 0x????????
Область данных	0x???????? – 0x????????
Размер сектора: Размер кластера: Общее число секторов: Адрес первого кластера, содержащего файл 1.txt: Адрес первого кластера, содержащего файл 2.txt: Адрес первого кластера, содержащего файл 3.txt:	

2.2.14. Рассмотрим загрузочный сектор. Его содержимое будет приблизительно таким:

```
00000000 EB 3C 90 6D 6B 66 73 2E 66 61 74 00 02 04 04 00    .<.mkfs.fat.....
00000010 02 00 02 20 4E F8 14 00 20 00 40 00 00 00 00 00    ... N... .@.....
00000020 00 00 00 00 80 00 29 B9 D0 EC B7 4E 4F 20 4E 41    ......)....NO NA
00000030 4D 45 20 20 20 20 46 41 54 31 36 20 20 20 0E 1F    ME FAT16 ..
00000040 BE 5B 7C AC 22 C0 74 0B 56 B4 0E BB 07 00 CD 10    .[|.".t.V.......
00000050 5E EB F0 32 E4 CD 16 CD 19 EB FE 54 68 69 73 20    ^..2.......This
00000060 69 73 20 6E 6F 74 20 61 20 62 6F 6F 74 61 62 6C    is not a bootabl
00000070 65 20 64 69 73 6B 2E 20 20 50 6C 65 61 73 65 20    e disk. Please
00000080 69 6E 73 65 72 74 20 61 20 62 6F 6F 74 61 62 6C    insert a bootabl
00000090 65 20 66 6C 6F 70 70 79 20 61 6E 64 0D 0A 70 72    e floppy and..pr
000000A0 65 73 73 20 61 6E 79 20 6B 65 79 20 74 6F 20 74    ess any key to t
000000B0 72 79 20 61 67 61 69 6E 20 2E 2E 2E 20 0D 0A 00    ry again ... ...
        …
```

Наиболее интересными для нас будут следующие поля (Рисунок 5):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0	Команда перехода к загрузочному коду			Имя OEM								Количество байт в секторе		Количество секторов в кластере	Размер зарезервированной области в секторах	
0x10	Количество копий FAT	Максимальное количество файлов в корневом каталоге		Количество секторов в ФС		Тип носителя	Размер каждой копии FAT в секторах		-	-	-	-	-	-	-	-

Рисунок 5 Поля загрузочного сектора

Обратим внимание, что некоторые поля записаны в обратном порядке. Например, в данном случае, количество байт в секторе (12 и 13 байты) будет равно 0x00 0x02 -> поменяем местами -> 0x02 0x00 ->200_{16}=512_{10}. Т.е. 512 байт. Загрузочный сектор оканчивается последовательностью символов 0x55AA. FAT таблица располагается сразу после зарезервированной областью.

Определите адрес, по которому она расположена. Внесите ответ в таблицу выше. ***Решение*** *поместите в отчет.*

Далее после FAT таблицы и ее копии располагается корневой каталог.

Вычислите его адрес. Внесите ответ в таблицу выше. ***Решение*** *поместите в отчет.*

2.2.15. Перейдите к корневому каталогу.

Одна запись в нем будет выглядеть приблизительно следующим образом:

```
00005800    41 31 00 2E 00 74 00 78 00 74 00 0F 00 89 00 00    A1...t.x.t......
00005810    FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF    ................
00005820    31 20 20 20 20 20 20 20 54 58 54 20 00 9C 17 9A    1 TXT ....
00005830    75 53 75 53 00 00 17 9A 75 53 03 00 36 01 00 00    uSuS....uS..6...
```

На первых двух строчках располагается обозначение имени файла. Следующие две занимает описание его характеристик и параметров.

Запись в нем устроена следующим образом (Рисунок 6):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Имя файла								Тип файла			Атрибуты*	Резерв	10ms создания файла	Время создания файла	
10	Дата создания файла		Дата последнего обращения к файлу		Не используется		Время последнего изменения		Дата последнего изменения		Номер первого кластера		Размер файла			

Атрибуты

Бит	7	6	5	6	3	2	1	0
Значение	0	0	Архив	Директория	Метка тома	Системный файл	Скрытый файл	Только чтение

Рисунок 6 Поля корневого каталога

Измените значения атрибутов записи таким образом, чтобы файл был обозначен, как директория.

Время в дата шифруются следующим образом. Значение из поля разбивается на блоки по несколько бит (0-4 – часы, 5-10 – минуты, 11-15 – секунды), каждый из которых обозначает свой параметр.

Например, время создания данного файла можно вычислить так:

0x179A -> 0x9A17 -> 1001101000010111_2 -> 10011 010000 10111 -> 19 16 23 -> 19:16:46 (секунды считаются парами, поэтому умножаем значение на 2).

Аналогично расшифровывается дата. Значение из поля разбивается следующим образом: 0-6 – год, 7-10 – месяц, 11-15 – день.

Для данного примера:

0x7553 -> 0x5375 -> 101001101110101_2 -> 101001 1011 10101 -> 41 11 21 -> 21-11-2021 (к значению 41 прибавляется значение 1980)

Измените у файла 2.txt дату создания на ваш день рождения и время создания на 12:34:56

Изменим значение первого байта названия (В данном случае 0x31) файла *3.txt* на 0xE5.

Сохраним измененные значения. Далее выйдем из редактора и смонтируем измененную файловую систему в каталог 1. Перейдем в него.

Сравните содержимое каталога со значениями, полученными в пункте 2.2.10. Сделайте выводы.

2.2.16. Демонтируйте файловую систему и снова войдите в Hex редактор.

2.2.17. По вычисленному ранее адресу перейдите к FAT таблице. Она будет представлять из себя примерно следующее:

```
00000800    F8 FF FF FF 00 00 FF FF FF FF 06 00 07 00 08 00    ................
00000810    09 00 FF FF 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

Первый байт – 0xF8 – является дескриптором, определяющим FAT таблицу. Далее следуют 3 байта со значениями FF.

Один элемент таблицы FAT равен числу, стоящую у ее названия. Т.к. мы работаем с FAT16, то одна запись занимает 16 бит – 2 байта. Нумерация начинается с 0.

В данном примере нулевая запись равна 0xF8FF, первая - 0xFFFF, вторая - 0x0000, третья – 0xFFFF и т.д. Нулевые значения обозначают, что кластер с заданным номером пуст. Числа большие нуля обозначают номер следующего кластера, в котором лежит часть нашего файла. 0xFFFF означает, что данный кластер заполнен данными и является последним в цепочке.

Вернемся к корневому каталогу и посмотрим значение номера первого кластера для файла 1.txt. Оно равно 3. Отсчитаем от начала 3 запись. Она равна 0xFFFF. Значит данный файл располагается только в одном кластере с номером 3. Аналогично файл 2.txt располагается в 4 кластере.

У файла 3.txt значение первого кластера равно 5. В данной паре ячеек лежит число 6 – номер следующего кластера. Далее лежат 7, 8, и 9. В девятой ячейке лежит значение 0xFFFF – конец файла. Таким образом, данный файл занимает 5 ячеек, соответственно – 5 кластеров.

*Вычислите адрес, по которому располагаются данные файла 1.txt. Для этого необходимо рассчитать размер корневого каталога и прибавить его к адресу начала каталога. Полученный адрес будет являться адресом **второго** кластера. Относительно него возможно получить адрес следующих кластеров, прибавляя его размер. Полученное значение будет адресом начала файла 1.txt. Внесите ответ в таблицу выше. **Решение** поместите в отчет.*

Аналогичным образом, возможно рассчитать, где располагаются остальные файлы. В случае расположения файлов в нескольких кластерах, данные объединяются в один файл.

*Аналогично получите адреса остальных файлов. Внесите ответ в таблицу выше. **Решение** поместите в отчет.*

2.3. Криминалистический анализ файловой системы.

Вам был передан образ, снятый с диска, оставленного на месте преступления. Файл образа прилагается к лабораторной работе. Известно, что на нем содержался секретный файл, однако диск был испорчен и данные были повреждены. Ваша задача вручную восстановить секретный файл, не прибегая к специализированным программам. В отчете укажите последовательность действий и что было изменено.

Контрольные вопросы

1. Что такое файловая система?
2. Какие существуют типы файловых систем?
3. Чем FAT отличается от NTFS?
4. Для чего предназначаются RAID массивы?
5. Как организован RAID 1?

Список литературы

[1] С. А. Лупин, *Лекция 12 по курсу Операционные сиземы,* Зеленоград, 2022.

[2] В. Е. Карпов и К. А. Коньков, Основы операционных систем, Москва: Физматкнига, 2019, p. 326.

[3] Э. Таненбаум и Х. Бос, Современные операционные системы, Санкт-Петербург: Питер, 2021.

[4] Б. Кэрриэ, Криминалистический анализ файловых систем, Санкт-Петербург: Питер, 2007.