操 作 系 统 **第 五 次 作 业**

2021/6/11

1. Consider a RAID organization comprising five disks in total, how many blocks are accessed in order to perform the following operations for RAID-5 and RAID-6?

a. An update of one block of data

b. An update of seven continuous blocks of data. Assume that the seven contiguous blocks begin at a boundary of a stripe.

RAID-5: a.2 个块，一个块包含需要更新的数据，一个块包含校验数据（RMW）

b.10 个块，读 B4 写 A1-A4, Ap, B1-B3, Bp（RRW）

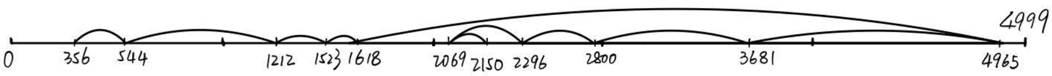
RAID-6: a. 3 个块，一个块包含需要更新的数据，另外两个块包含校验数据；(RMW)

b.13 个块，写 A1-A3, Ap, Aq, B1-B3, Bp, Bq（RRW）

读 Cp, Cq, C1 写 C1, Cp, Cq（RMW）

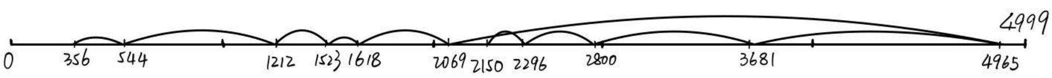
2. Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is: 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681 Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms? a. FCFS b. SSTF c. SCAN d. LOOK e. C-SCAN f. C-LOOK

FCFS: 81+857+1084+504+2256+1074+1262+1167+3442+1284 = 13011 cylinders

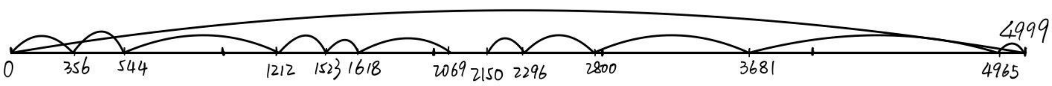


SSTF: 81+227+504+881+1284+3347+95+311+668+188 = 7586 cylinders

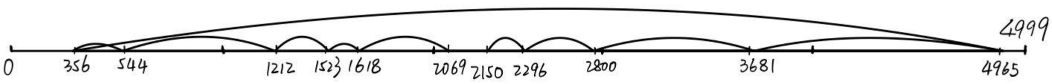
SCAN: 146+504+881+1284+34+2930+478+95+311+668+188 = 7492 cylinders



LOOK: 146+504+881+1284+2896+478+95+311+668+188 = 7424 cylinders



C-SCAN: 146+504+881+1284+34+4999+356+188+856+311+95+451 = 9917 cylinders



C-LOOK: 146+504+881+1284+4609+188+856+311+95+451 = 9137 cylinders

3. Explain what open-file table is and why we need it.

打开文件表是指一个存储了一个打开文件相关全部信息的表格，这个表格在内核中，其中的信息包括：

1. 当前文件偏移量（调用 read()和 write()时更新，或使用 lseek()直接修改）
2. 打开文件时所使用的状态标识（即，open()的 flags 参数）
3. 文件访问模式（如调用 open()时所设置的只读模式、只写模式或读写模式）
4. 与信号驱动相关的设置
5. 对该文件 i-node 对象的引用
6. 文件类型（例如：常规文件、套接字或 FIFO）和访问权限
7. 一个指针，指向该文件所持有的锁列表
8. 文件的各种属性，包括文件大小以及与不同类型操作相关的时间戳

表格中的每一项根据文件描述符进行索引。

需要该表的原因：在文件的读写操作中，操作系统根据文件描述符在打开文件表中找到对应文件的属性，并根据这些属性进行相关操作。

4. Explain the concept of file and directory, and what does “755” mean for file permission?

文件：A file is a uniform logical view of stored information provided by OS.

操作系统的角度：操作系统对存储设备的物理属性加以抽象，从而定义的逻辑存储单位

用户的角度：逻辑外存的最小分配单位

目录：目录也是一种文件，其内容是目录条目（文件名+文件属性）的序列

文件权限：

<table>
<tr><td>权限项</td><td>文件类型</td><td>读</td><td>写</td><td>执行</td><td>读</td><td>写</td><td>执行</td><td>读</td><td>写</td><td>执行</td></tr>
<tr><td>字符表示</td><td>(d|l|c|s|p)</td><td>(r)</td><td>(w)</td><td>(x)</td><td>(r)</td><td>(w)</td><td>(x)</td><td>(r)</td><td>(w)</td><td>(x)</td></tr>
<tr><td>数字表示</td><td></td><td>4</td><td>2</td><td>1</td><td>4</td><td>2</td><td>1</td><td>4</td><td>2</td><td>1</td></tr>
<tr><td>权限分配</td><td></td><td colspan="3">文件所有者</td><td colspan="3">文件所属组用户</td><td colspan="3">其他用户</td></tr>
</table>

“775”即为“111 111 101”表示文件所有者可读可写可执行该文件，文件所属组用户可读可写可执行该文件，其他用户可读不可写可执行该文件。

5. Explain the problems of using continuous allocation for file system layout and how to solve them

Problem: 1.External fragmentation 2.Growth problem. There is no space for file to grow

Solution: 1.Chop the storage device into equal-sized blocks and fill the new file into the empty space in a block-by-block manner.

2.Use the idea of linked list. Borrow 4bytes from each block, to write the bloch# of the next block.

6. What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file? What is the major problem of FAT?

Advantages: The random access problem can be eased by keeping a cache vision of FAT inside the kernel

Major problem: The entire FAT has to be stored in memory.

7. Consider a file system similar to the one used by UNIX with indexed allocation, and assume that every file uses only one block. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c in the following two cases? Should provide the detailed

workflow.

a. Assume that none of the disk blocks and inodes is currently being cached.

b. Assume that none of the disk blocks is currently being cached but all inodes are in memory.

a. 1. Search the Root Directory for the inode# of “a/”

2. Use the number of inode to locate the inode in the Index Node Table

3. Read the inode and locate the direct block#0 in the disk. Search the block for the inode# of “b/”

4. Use the number of inode to locate the inode in the Index Node Table

5. Read the inode and locate the direct block#0 in the disk. Search the block for the inode# of “c”

6. Use the number of inode to locate the inode in the Index Node Table

7. Read the inode and locate the direct block#0 in the disk. Read the block which contains the file.

7 times of I/O in total

b. 1. Search the Root Directory for the inode# of “a/”

2. Use the number of inode to locate the inode in the Index Node Table

3. Read the inode and locate the direct block#0 in the disk. Search the block for the inode# of “b/”

4. Use the number of inode to locate the inode in the Index Node Table

5. Read the inode and locate the direct block#0 in the disk. Search the block for the inode# of “c”

6. Use the number of inode to locate the inode in the Index Node Table

7. Read the inode and locate the direct block#0 in the disk. Read the block which contains the file.

4 times of I/O in total

8. Consider a file system that uses inodes to represent files. Disk blocks are 8-KB in size and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, plus single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

12 * 8KB + 2048 * 8KB + 2048 * 2048 * 8KB + 2048 * 2048 * 2048 * 8KB = 64TB

9. What is the 8+3 naming convention in FAT32 file system, and how to manage long filenames?

8+3: 8 characters for file name and 3 characters for file extension.

long filenames: Add more entries(LEN entry) to represent the filename. Each LEN entry represents 13 characters in Unicode, i.e. 2 bytes per character.

Normal entry

Bytes	Description
0-0	1st character of the filename (0x00 or 0xe5 means unallocated)
1-10	7+3 characters of filename + extension.
11-11	File attributes (e.g., read only, hidden)
12-12	Reserved.
13-19	Creation and access time information.
20-21	High 2 bytes of the first cluster address (0 for FAT16 and FAT12).
22-25	Written time information.
26-27	Low 2 bytes of first cluster address.
28-31	File size.

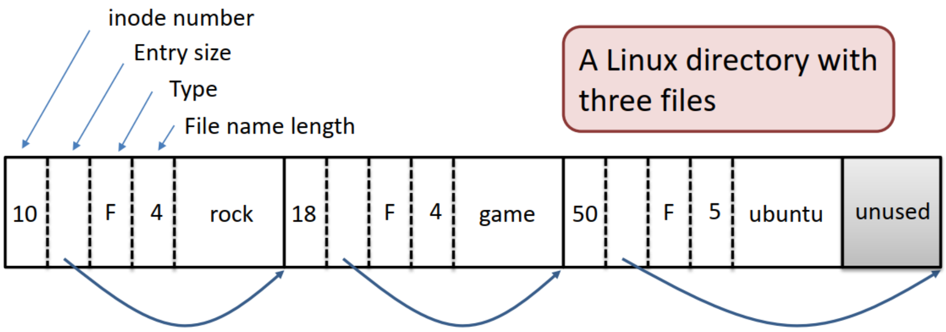
LFN entry

Bytes	Description
0-0	Sequence Number
1-10	File name characters (5 characters in Unicode)
11-11	File attributes - always 0x0F
12-12	Reserved.
13-13	Checksum
14-25	File name characters (6 characters in Unicode)
26-27	Reserved
28-31	File name characters (2 characters in Unicode)

10. How are directory entries managed in FAT and Ext file systems?

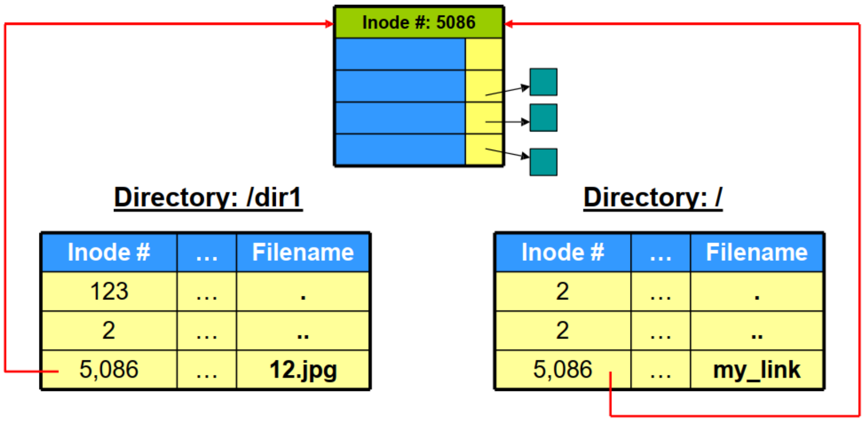
FAT : A FAT table manages all director entries, a directory entry stores filename, extension name, etc. In every block visited stores the address of the next block, or EOF

Ext : A directory file's content is the inode-number-table of files in this directory, while the inode table, as a layered linked list, provides the address of every block in this file

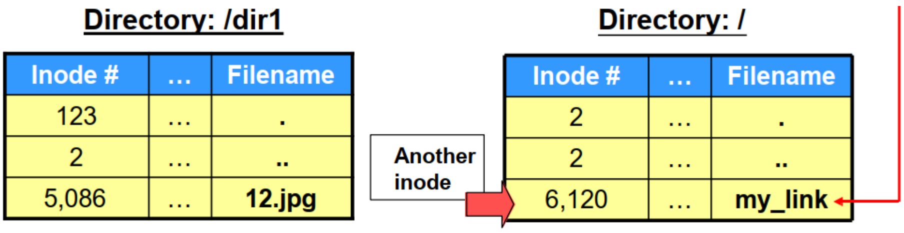


11. What is the difference between hard link and symbolic link?

Hard Link: No file content is created.



Symbolic Link: A new inode is created for each symbolic link, it stores the pathname

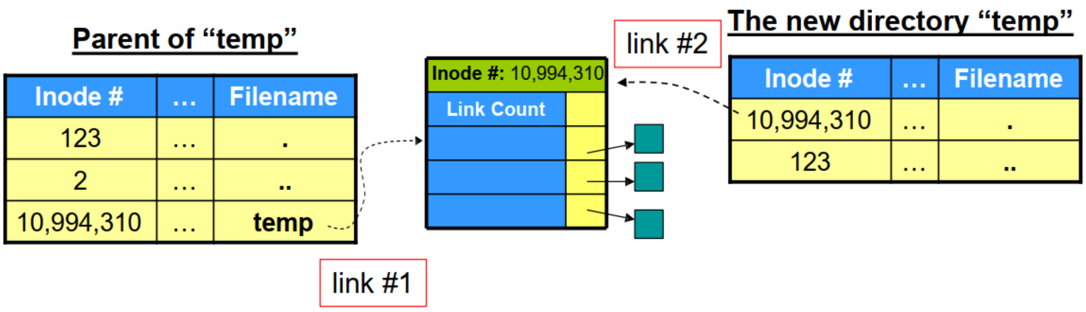


12. What are the initial link counts when a regular file or a directory is created? Why?

When a regular file is created, the initial link count is 1;

When a directory is created, the initial link count is 2;

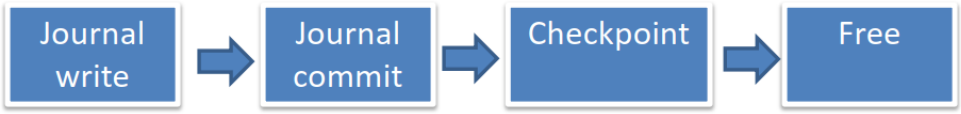
Reason: For a regular file, only the directory entry in the parent directory points to the inode. For a directory, the directory entry in the parent directory and the hand link to itself in the newly created directory will both point to the inode.



13. What is the difference between data journaling and metadata journaling? Explain the operation sequence for each of the two journaling methods.

Difference: Metadata Journaling only logs metadata, while data journaling logs both data and metadata

Data Journaling:



Journal write: Write the contents of the transaction (including TxB, metadata and data)
Journal commit: Metadata and data (including TxE)
Checkpoint: Write the contents of the update to their on-disk locations

TxB	Journal Contents (metadata)	Journal Contents (data)	TxE	File System Metadata	File System Data
issue	issue	issue			
complete					
	complete				
		complete			
			issue		
			complete		
				issue	issue
					complete
				complete	

Metadata Jurnaling: the first two steps can be issued in parallel.

TxB	Journal Contents (metadata)	TxE	File System Metadata	File System Data
issue	issue			issue
				complete
complete				
	complete			
		issue		
		complete		
			issue	
			complete	

14. What are the three I/O control methods?

1.Polling（轮询）：由 CPU 定时发出询问，依序询问每一个周边设备是否需要其服务，有即给予服务，服务结束后再问下一个周边，接着不断周而复始。

2.Interrupt（中断）：中断过程包括：

①中断源发出中断请求;

②判断当前处理机是否允许中断和该中断源是否被屏蔽;

③优先权排队;

④处理机执行完当前指令或当前指令无法执行完，则立即停止当前程序，保护断点地址和处理机当前状态，转入相应的中断服务程序;

⑤执行中断服务程序;

⑥恢复被保护的状态，执行“中断返回”指令回到被中断的程序或转入其他程序。

上述过程中前四项操作是由硬件完成的，后两项是由软件完成的。

3.DMA（直接存储器访问）

①请求：CPU 对 DMA 控制器初始化，并向 I/O 接口发出操作命令，I/O 接口提出 DMA 请求。

②响应：DMA 控制器对 DMA 请求判别优先级及屏蔽，向总线裁决逻辑提出总线请求。当 CPU 执行完当前总线周期即可释放总线控制权。此时，总线裁决逻辑输出总线应答，表示 DMA 已经响应，通过 DMA 控制器通知 I/O 接口开始 DMA 传输。

③传输：DMA 控制器获得总线控制权后，CPU 即刻挂起或只执行内部操作，由 DMA 控制器输出读写命令，直接控制 RAM 与 I/O 接口进行 DMA 传输。

在 DMA 控制器的控制下，在存储器和外部设备之间直接进行数据传送，在传送过程中不需要 CPU 的参与。开始时需提供要传送的数据的起始位置和数据长度。

④结束：当完成规定的成批数据传送后，DMA 控制器即释放总线控制权，并向 I/O 接口发出结束信号。当 I/O 接口收到结束信号后，一方面停 止 I/O 设备的工作，另一方面向 CPU 提出中断请求，使 CPU 从不介入的状态解脱，并执行一段检查本次 DMA 传输操作正确性的代码。最后，带着本次操作结果及状态继续执行原来的程序。

15. List at least three kinds of I/O devices and explain how to provide a standard and uniform application I/O interface?

I/O devices: Block device(disk), character device(keyboard, mice), clock & timer;

How?: 从各种各样的 I/O 设备中，抽象一些通用类型，每种通用类型可以通过一组标准函数来访问，这些差异被封装到内核模块中（称为设备驱动程序），这些设备驱动程序一方面可以定制以适应各种设备，另一方面也提供一组标准接口。

16. What services are provided by the kernel I/O subsystem?

1.I/O scheduling: Maintain a per-device queue, Re-ordering the request, Average waiting time, fairness;

2.Buffering: Store data in memory while transferring between devices;

3.Caching: Faster device holding copy of data

4.SPOOLing

5.Error handling and I/O protection

6.Power management