操作系统作业 ⑦

姓名：李昱祁
学号：PB18071496

EX1

8+3 命名规范指文件的完整名称为“文件名”+“扩展名”，其中“文件名”长度为 8 Byte，“扩展名”长度为 3 Byte，共可表示 11 个英文字符
FAT32 支持长文件名，即若“文件名”大于 8 Byte，或者“扩展名”大于 3 Byte，文件系统会将其记录在 LFN 项中。每个 LFN 项大小为 32 Byte，其中：

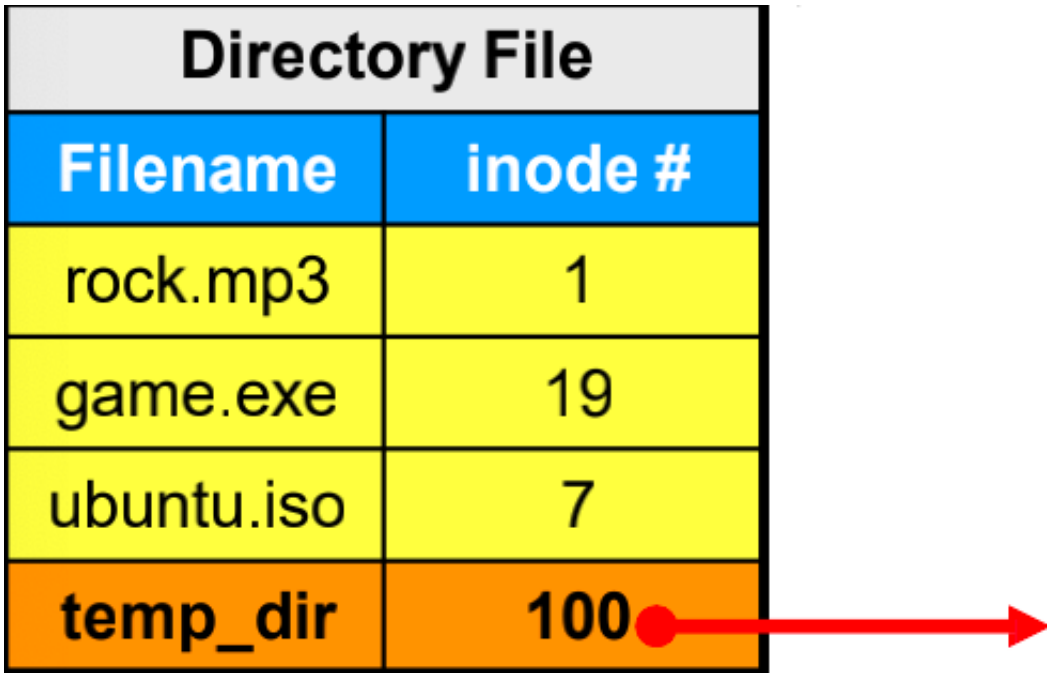
- 26 Byte 用于记录字符（采用 Unicode 编码，共可表示 13 个英文字符，有多余的空间会根据相应规则补充）
- 余下的 Byte 用来：记录该项的内容在完整文件名中的顺序、校验值、文件起始 cluster 号等信息，或被系统保留

EX2

- FAT：
  FAT 文件系统中，目录条目为一个 32 字节的结构体，它包含了完整文件名，以及文件的一些属性：**起始簇地址**、权限、创建日期等等……一些信息，具体如下组织：

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.

- EXT：
  EXT 文件系统中，目录条目包含了两部分：文件名，和一个 inode 结构体的地址。
  大致结构如下：（其中每一行为一个目录条目）



（inode 号指向一个128 字节的 inode 结构体，它记录了关于此文件的**全部信息**：文件类型、文件大小，链接数，以及 12 个直接块的指针、3 个间接块的指针等）

EX3

两者的主要差异在于：

- 硬链接（hard link）不会创建新的 inode，实际上相当于为同一个文件设置了两个路径名，通过这两个路径名都可以索引到同一个 inode
- 符号链接（symbolic link）会创建新的 inode，用于存储目标文件的路径名

EX4

- 一个普通文件（regular file）刚被创建时，其 inode 中 link count 值为 1，表示此时该文件仅通过创建文件时给定的路径名进行链接
- 目录被创建后，link count 初始值为 2，这是因为：此目录不仅与上级目录链接，还要与自身链接（通过“ . ”目录）

EX5

不同之处在于： data journaling 需要将文件数据写两次（即数据也要写入日志区）；而 metadata journaling 只需要一次（只有元数据需要写入日志区）

- **data journaling**：

  1. journal write：在日志区写入 TxB和元数据、数据
  2. journal commit：在日志区写入 TxE
  3. checkpoint：写入文件系统，即把要写的数据写到磁盘相应位置上
  4. free：释放该 transaction 在日志区中所占的空间

- **metadata journaling（ordered mode）**：

  1. Data write + journal metadata write：在日志区写入 TxB、元数据，在磁盘上写入数据
  2. journal commit：在日志区写入 TxE
  3. checkpoint metadata：更新文件系统中相应的元数据
  4. free：释放该 transaction 在日志区中所占的空间

EX6

三种方式如下：

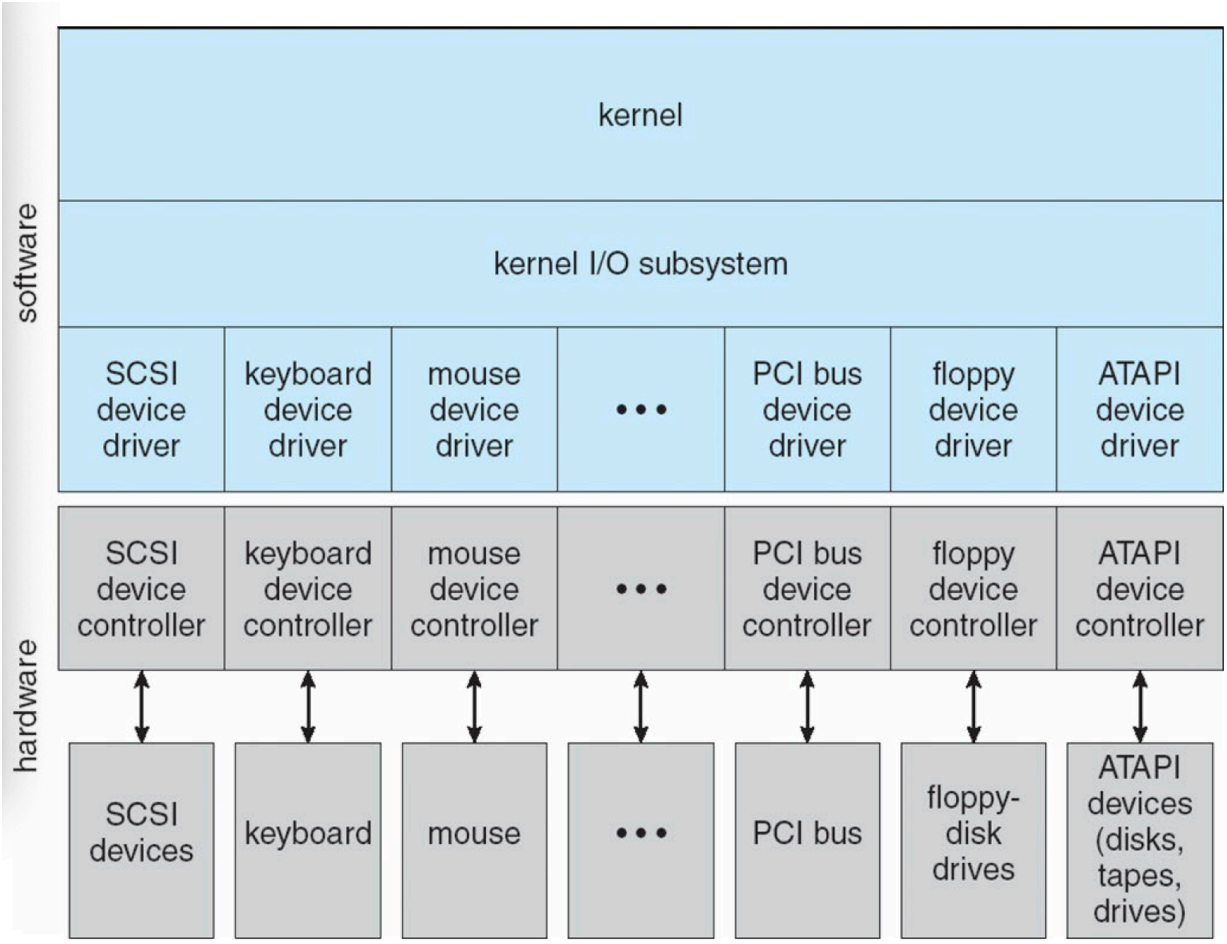
- 轮询：主机与控制器之间重复进行握手协议，不断地检查状态寄存器的**忙位**
- 中断：通过中断机制实现 I/O。进行需要进行 I/O 时产生中断，通过陷入向量找到OS 提供的相应 I/O 中断服务程序，由它来完成 I/O 任务
- 直接内存访问：对要执行大量传输的任务，可通过 DMA 控制器来完成。CPU 只需将要写入内存的命令块地址告诉 DAM 控制器，后续工作便交由 DAM 控制器完成，CPU 可转

而执行其它任务

EX7

I/O 设备有：键盘、鼠标、显示器等
要想提供一个标准和统一的 I/O 接口，需要**封装**、**抽象**、**分层**。具体来说：

- 从各种各样 I/O 设备中抽象出一些通用类型，为每个类型提供一组标准接口
- 通过设备驱动程序将差异封装。设置驱动程序层，为内核 I/O 子系统隐藏控制器设备之间的差异
- 内核中的 I/O 相关部分按软件层来组织（比如上述提到的驱动程序层就为其中一层）



EX8

答：内核的 I/O 子系统可以提供的服务有很多，下面介绍其中几种：

- 调度：对一组 I/O 请求，要按一定的顺序来执行它们，以提高计算机效率
- 缓冲：I/O 子系统设置缓冲区（Buffer），用于保存在两个设备之间，或者设备和应用程序之间传输的数据。作用有三：

1. 处理 数据流的生产者、消费者之间速度不匹配的问题
2. 协调传输大小不一数据的设备，在缓冲区对消息进行分段、重组等
3. 支持应用程序 I/O 的复制语义

- 缓存：一些可能被访问到的磁盘信息会被放入缓存中。内核收到文件 I/O 请求时，首先访问缓冲区的缓存，以便查看文件区域是否已经在内存区域中可用
- 假脱机：保存设备输出的缓冲区，使某些 I/O 设备（如打印机）同时只能接收一个数据流
- 错误处理：用于处理 I/O 传输中的故障，如网络超载等。I/O 子系统可以处理相关 I/O 系统调用返回的错误信息
- I/O 保护：I/O 指令只能通过 I/O 相关系统调用发出；用于 I/O 的映射内存受到内存保护，用户对于该部分内存的访问受到控制

  等等……