Technology Tarena 达内科技

Unix系统高级编程

File 1

DAY04

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	malloc/free的简化实现
	10:30 ~ 11:20	内存映射的建立与解除
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	系统调用
	15:00 ~ 15:50	文件系统与文件
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑

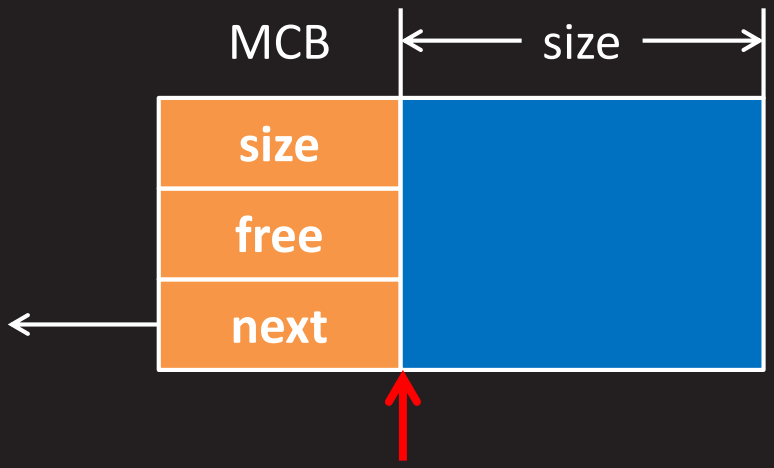
malloc/free的简化实现

Technology Tarena 达内科技

内存控制块

内存控制块

- 内存控制块用于管理每次分配的内存块，记录该内存块的字节大小、忙闲状态，以及相邻内存控制块的首地址
  - typedef struct mem_control_block {
    size_t size; // 本块大小
    bool free; // 空闲标志
    struct mem_control_block* next; // 后块指针
    } MCB;



知识讲解

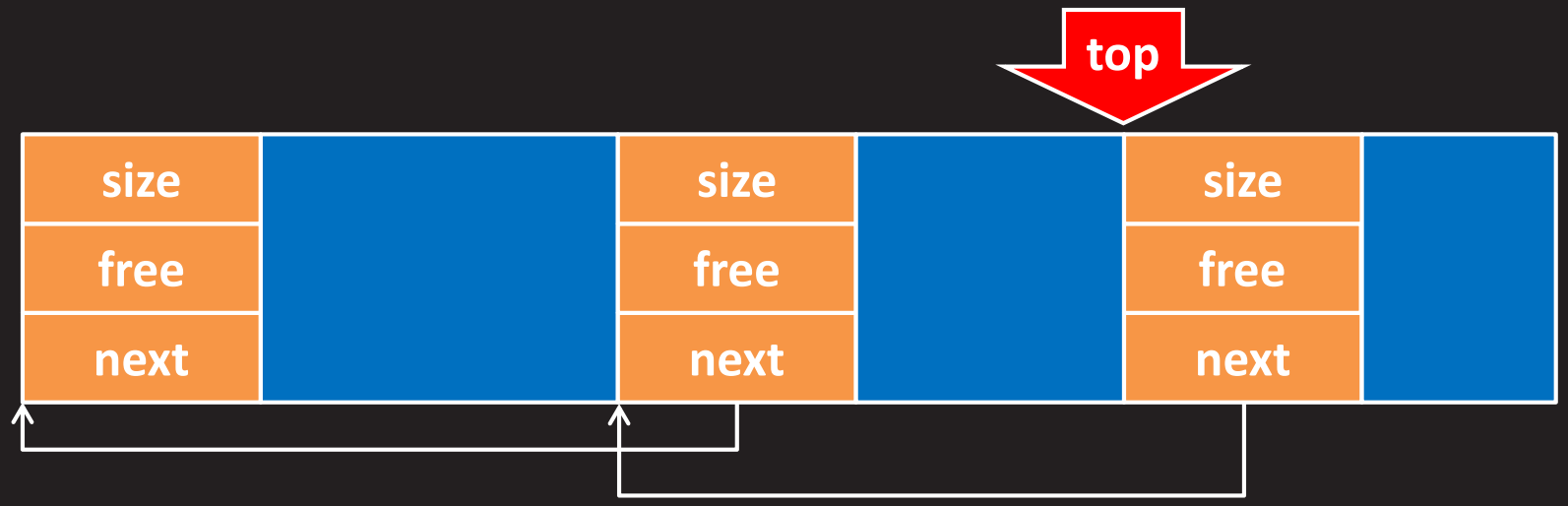
Technology Tarena 达内科技

单向链表栈

单向链表栈

- 多次内存分配会产生多个内存控制块，可将其组织成链表栈的形式以便于集中管理，可由栈顶指针遍历该链表
  - MCB* g_top = NULL; // 栈顶指针
  - MCB* mcb;
    for (mcb = g_top; mcb; mcb = mcb->next)
        …

知识讲解



Technology Tarena 达内科技

分配内存

分配内存

- 遍历内存控制块链表，若有大小足够的空闲块，则重用该块，否则分配新的足量内存并将其控制块压入链表栈
  - void* my_malloc (size_t size) {

```
MCB* mcb;
for (mcb = g_top; mcb; mcb = mcb->next)
    if (mcb->free && mcb->size >= size)
        break;
if (! mcb) {
    mcb = sbrk (sizeof (MCB) + size);
    if (mcb == (void*)-1) return NULL;
    mcb->size = size;
    mcb->next = g_top;
    g_top = mcb; }
mcb->free = false;
return mcb + 1;
}
```

知识讲解

Technology Tarena 达内科技

释放内存

释放内存

- 先将被释放内存块标记为空闲，然后遍历内存控制块链表，将靠近栈顶的连续空闲块及其内存控制块一并释放
  - void my_free (void* ptr) {

```
if (ptr) {
    MCB* mcb = (MCB*)ptr - 1;
    mcb->free = true;
    for (mcb = g_top; mcb->next; mcb = mcb->next)
        if (! mcb->free) break;
    if (mcb->free) {
        g_top = mcb->next;
        brk (mcb); }
    else if (mcb != g_top) {
        g_top = mcb;
        brk ((void*)mcb + sizeof (MCB) + mcb->size); }
}
}
```

知识讲解

malloc/free的简化实现

【参见：malloc.c】

课堂练习

- malloc/free的简化实现

内存映射的建立与解除

Technology Tarena 达内科技

建立内存映射

建立内存映射

- 建立虚拟内存到物理内存或文件的映射

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

void* mmap (void* start, size_t length, int prot,
    int flags, int fd, off_t offset);
```

成功返回映射区内存起始地址，失败返回MAP_FAILED(-1)

  - ***start***：映射区内存起始地址，NULL系统自动选定后返回
  - ***length***：映射区字节长度，自动按页(4K)圆整

知识讲解

建立内存映射（续1）

- 创建虚拟内存到物理内存或文件的映射
  - *prot*：映射区访问权限，可取以下值

    `PROT_READ` - 映射区可读

    `PROT_WRITE` - 映射区可写

    `PROT_EXEC` - 映射区可执行

    `PROT_NONE` - 映射区不可访问

知识讲解

建立内存映射（续2）

Technology Tarena 达内科技

- 创建虚拟内存到物理内存或文件的映射
  - ***flags***：映射标志，可取以下值

    MAP_ANONYMOUS - 匿名映射，将虚拟内存映射到物理内存而非文件，忽略***fd***和***offset***参数

    MAP_PRIVATE - 对映射区的写操作只反映到缓冲区中，并不会真正写入文件

    MAP_SHARED - 对映射区的写操作直接反映到文件中

    MAP_DENYWRITE - 拒绝其它对文件的写操作

    MAP_FIXED - 若在***start***上无法创建映射，则失败(无此标志系统会自动调整)

    MAP_LOCKED - 锁定映射区，保证其不被换出

知识讲解

建立内存映射（续3）

Technology Tarena 达内科技

- 创建虚拟内存到物理内存或文件的映射
  - ***fd***：文件描述符
  - ***offset***：文件偏移量，自动按页(4K)对齐
- 例如
  - char* p = (char*)mmap (NULL, 8192,
    PROT_READ | PROT_WRITE,
    MAP_ANONYMOUS | MAP_PRIVATE, 0, 0);
    if (p == MAP_FAILED) {
    perror ("mmap");
    exit (EXIT_FAILURE); }
  - strcpy (p, "Hello, Memory !");
    printf ("%s\n", p);

知识讲解

Technology Tarena 达内科技

解除内存映射

解除内存映射

Technology Tarena 达内科技

- 解除虚拟内存到物理内存或文件的映射

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

int munmap (void* start, size_t length);

                                成功返回0，失败返回-1
```

  - ***start***：映射区内存起始地址，必须是页的首地址
  - ***length***：映射区字节长度，自动按页(4K)圆整

知识讲解

解除内存映射（续1）

Technology Tarena 达内科技

- 例如
  - if (munmap (p, 4096) == -1) {
    perror ("munmap");
    exit (EXIT_FAILURE);
    }
    strcpy (p += 4096, "Hello, Memory !");
    printf ("%s\n", p);
    if (munmap (p, 4096) == -1) {
    perror ("munmap");
    exit (EXIT_FAILURE);
    }
- munmap允许对映射区的一部分解映射，但必须按页

知识讲解

解除内存映射（续2）

Technology Tarena 达内科技

- mmap/munmap底层不维护任何东西，只是返回一个首地址，所分配内存位于堆中
- brk/sbrk底层维护一个指针，记录所分配的内存结尾，所分配内存位于堆中，底层调用mmap/munmap
- malloc底层维护一个线性链表和必要的控制信息，不可越界访问，所分配内存位于堆中，底层调用brk/sbrk
- 每个进程都有4G的虚拟内存空间，虚拟内存地址只是一个数字，在与实际物理内存建立映射之前是不能访问的
- 所谓内存分配与释放，其本质就是建立或解除从虚拟内存到物理内存的映射，并在底层维护不同形式的数据结构，以把虚拟内存的占用与空闲情况记录下来

知识讲解

Technology Tarena 达内科技

内存映射的建立与解除

【参见：mmap.c】

课堂练习

- 内存映射的建立与解除

系统调用

Technology
Tarena
达内科技

Unix应用的层次结构

Unix应用的层次结构

Technology Tarena 达内科技

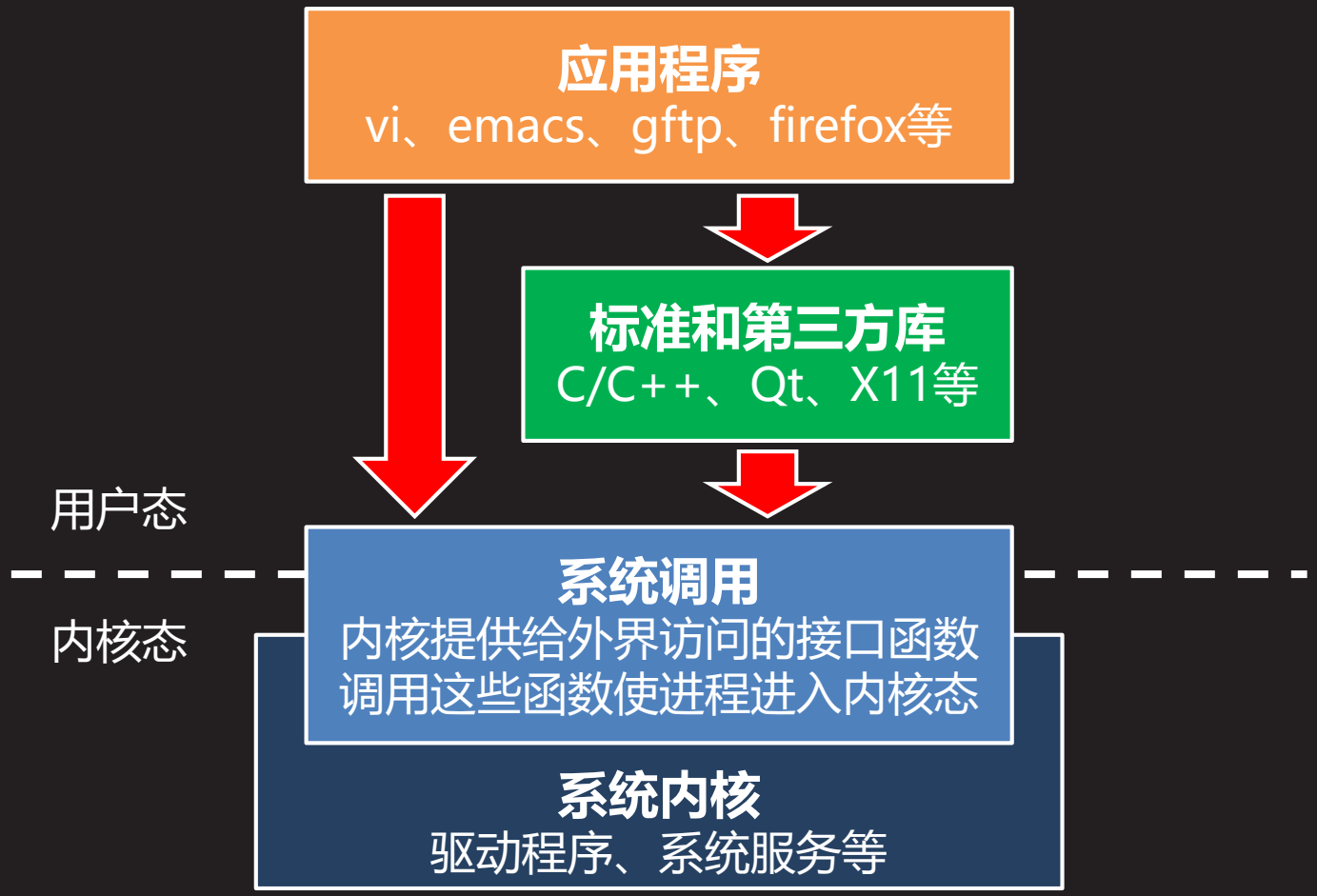
- Unix/Linux系统的大部分功能都是通过系统调用实现的，如open、close等
- Unix/Linux的系统调用已被封装成C函数的形式，但它们并不是C语言标准库的一部分
- 标准库函数大部分时间运行在用户态，但部分函数偶尔也会调用系统调用，进入内核态，如malloc、free等
- 程序员自己编写的代码也可以跳过标准库，直接使用系统调用，如brk、sbrk、mmap和munmap等，与操作系统内核交互，进入内核态
- 系统调用在内核中实现，其外部接口定义在C库中，该接口的实现借助软中断进入内核

知识讲解

Unix应用的层次结构（续1）

Technology Tarena 达内科技

- 从应用程序到操作系统内核需要经历如下调用链



Technology Tarena 达内科技

测试运行时间

测试运行时间

Technology Tarena 达内科技

- 测试应用程序的运行时间分配

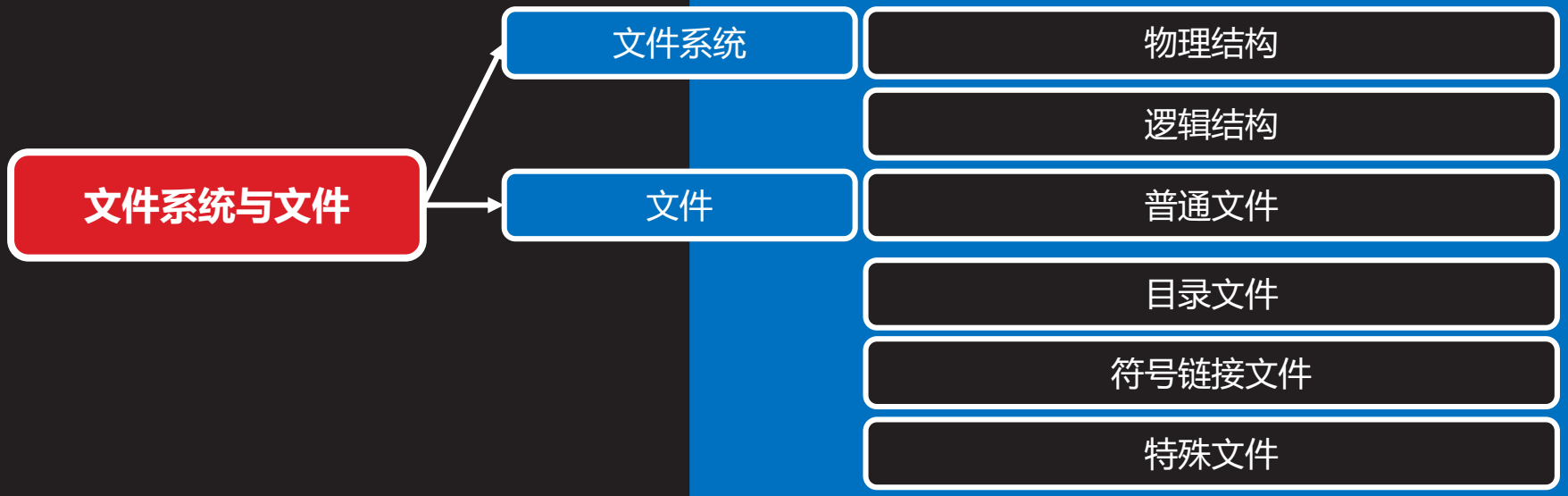
```
$ time a.out
```

```
real    0m18.705s
user    0m0.452s
sys     0m18.173s
```

  - real：总耗时 = 用户空间耗时 + 内核空间耗时 + 等待耗时
  - user：用户空间耗时
  - sys ：内核空间耗时

知识讲解

文件系统与文件



Technology Tarena 达内科技

文件系统

物理结构

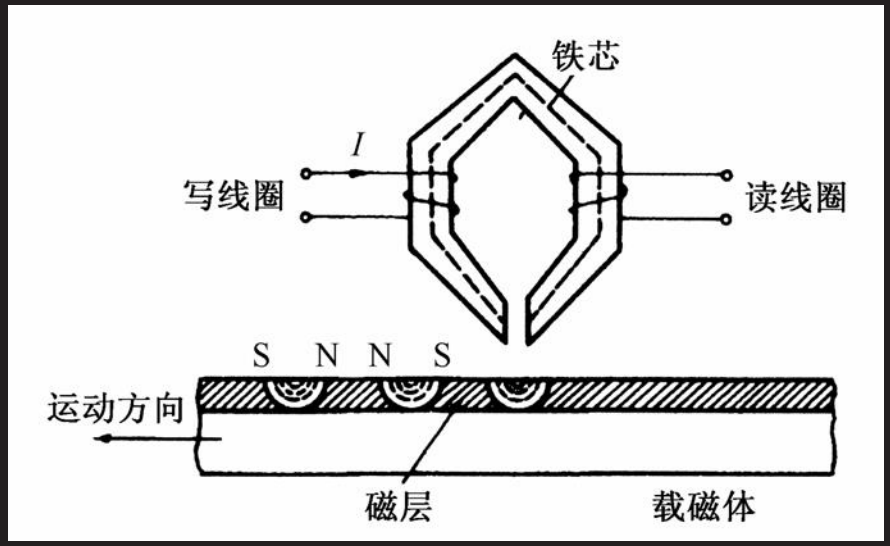
- 硬盘的物理结构
  - 驱动臂
  - 盘片
  - 主轴
  - 磁头

知识讲解

物理结构（续1）

Technology Tarena 达内科技

- 磁表面存储器的读写原理
  - 硬盘片的表面覆盖着薄薄的磁性涂层，涂层中含有无数微小的磁性颗粒，谓之磁畴
  - 相邻的若干磁畴组成一个磁性存储元，以其剩磁的极性表示二进制数字0和1
  - 为磁头的写线圈施加脉冲电流，可把一位二进制数字转换为磁性存储元的剩磁极性
  - 利用磁电变换，通过磁头的读线圈，可将磁性存储元的剩磁极性转换为相应的电信号，表示二进制数

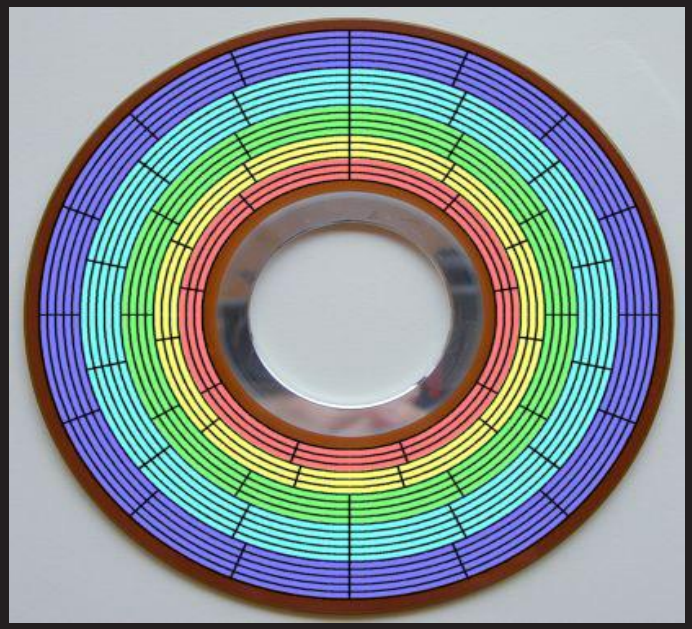


物理结构（续2）

Technology Tarena 达内科技

知识讲解

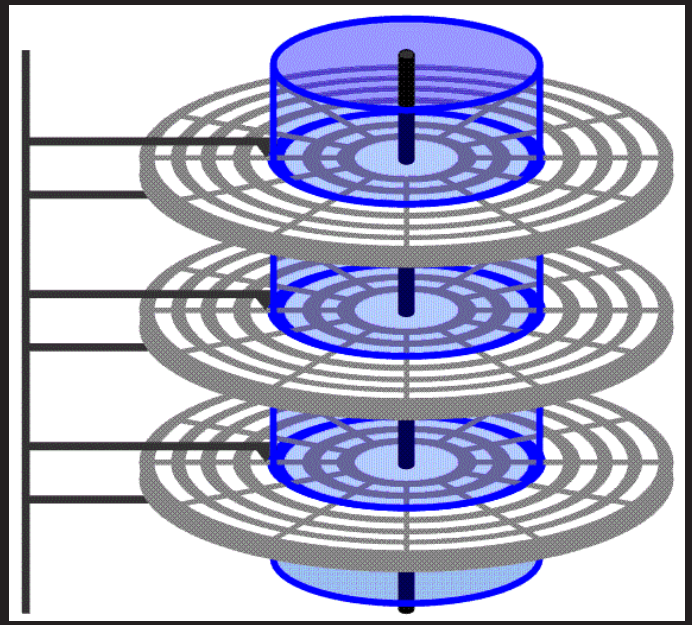
- 磁道和扇区
  - 磁盘旋转，磁头固定，每个磁头都会在盘片表面划出一个圆形轨迹。改变磁头的位置，可以形成若干大小不等的同心圆，这些同心圆就叫做磁道(Track)
  - 每张盘片的每个面上都有成千上万个磁道
  - 一个磁道，按照512字节等分，其中每个等分叫做扇区(Sector)
  - 扇区是最基本的文件存储单位
  - 每个磁道所包含的扇区数并不相等，越靠近外圈的磁道所包含的扇区越多



物理结构（续3）

Technology Tarena 达内科技

- 柱面、柱面组、分区和磁盘驱动器
  - 硬盘中，不同盘片相同半径的磁道所组成的圆柱称为柱面(Cylinder)。整个硬盘的柱面数与每张盘片的磁道数相等
  - 硬盘中的每个扇区可由以下三个坐标唯一确定
    磁头号：确定哪张盘面(一张盘片有两个盘面各对应一个磁头)
    柱面号：确定哪条磁道
    扇区号：确定哪个区域
  - 若干连续的柱面构成一个柱面组
  - 若干连续的柱面组构成一个分区
  - 每个分区都建有独立的文件系统
  - 若干个分区构成一个磁盘驱动器



知识讲解

逻辑结构

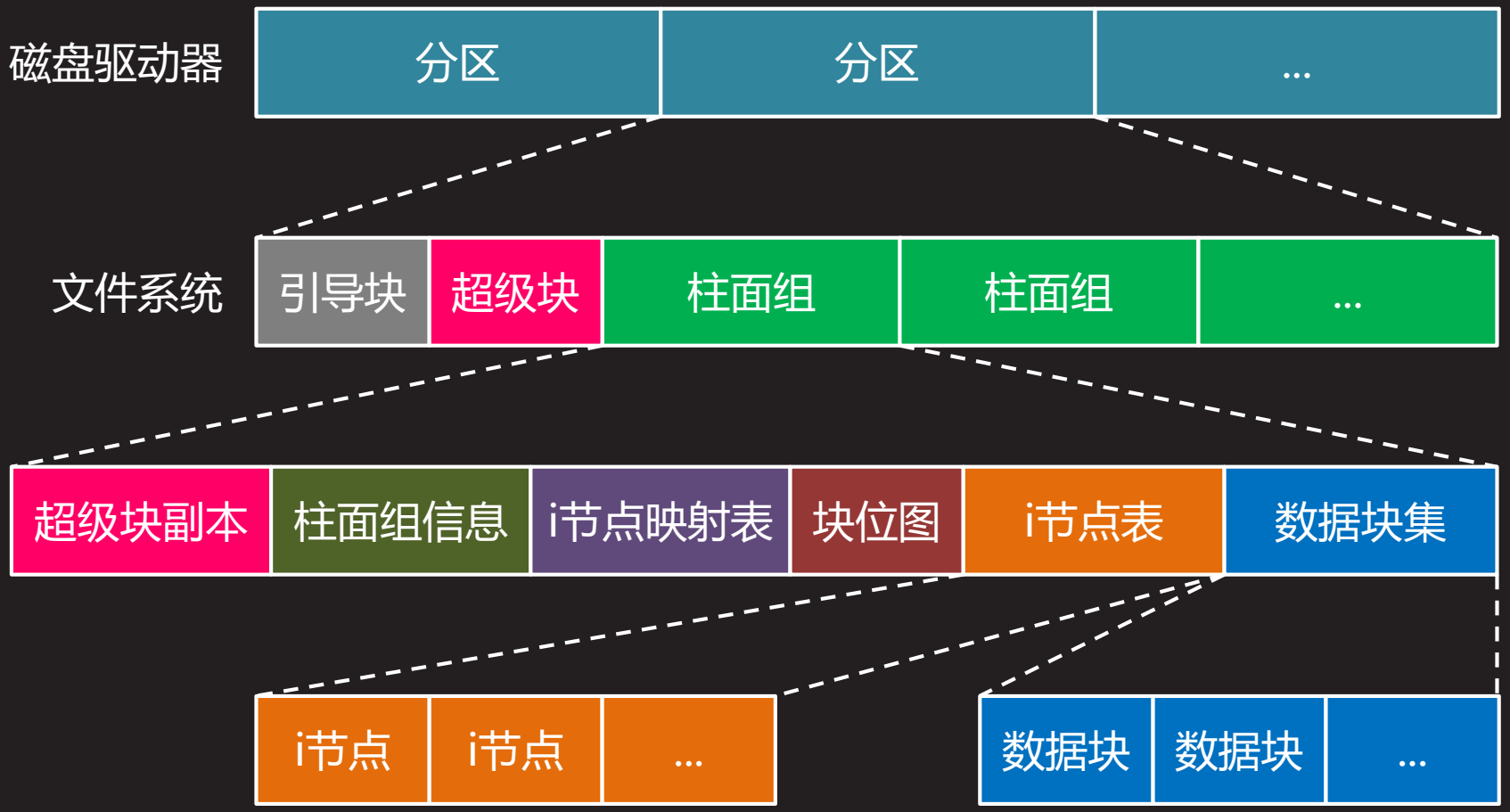
Technology Tarena 达内科技

- 一个磁盘驱动器被划分成一到多个分区，其中每个分区上都建有独立的文件系统，每个文件系统包括
  - 引导块：计算机加电启动时，ROM BIOS从这里读取可执行代码和数据，以完成操作系统自举
  - 超级块：记录文件系统的整体信息，如文件系统的格式和大小，i节点和数据块的总量、使用量和剩余量等等
  - 若干柱面组，其中每个柱面组包括
    - 超级块副本：同上
    - 柱面组信息：柱面组的整体描述
    - i节点映射表：i节点号与i节点磁盘位置的对应表
    - 块位图：位图中的每个二进制位对应一个数据块，用1和0表示该块处于占用或是空闲状态
    - i节点表：包含若干i节点，记录文件的元数据和数据块索引表
    - 数据块集：包含若干数据块，存储文件的内容数据

知识讲解

逻辑结构（续1）

- UFS (Unix File System)文件系统结构



知识讲解

逻辑结构（续2）

Technology Tarena 达内科技

- i节点，即索引节点(index node, inode)
  - 磁盘中的每个文件或目录都有唯一的一个的i节点与之对应
  - 每个i节点都有唯一的编号即i节点号，通过i节点映射表可以查到与每个i节点号相对应的i节点在磁盘上的存储位置
  - 文件名和i节点号的对应关系记录在该文件所在目录的目录文件中，目录文件中的一条这样的记录就是一个硬链接

知识讲解

```
$ ls -i /etc
```

```
262169 console-setup                    262342 os-release
262170 cron.d                           262334 pam.conf
262171 cron.daily                       262225 pam.d
262172 cron.hourly                      264643 papersize
262173 cron.monthly                     283386 passwd
```

逻辑结构（续3）

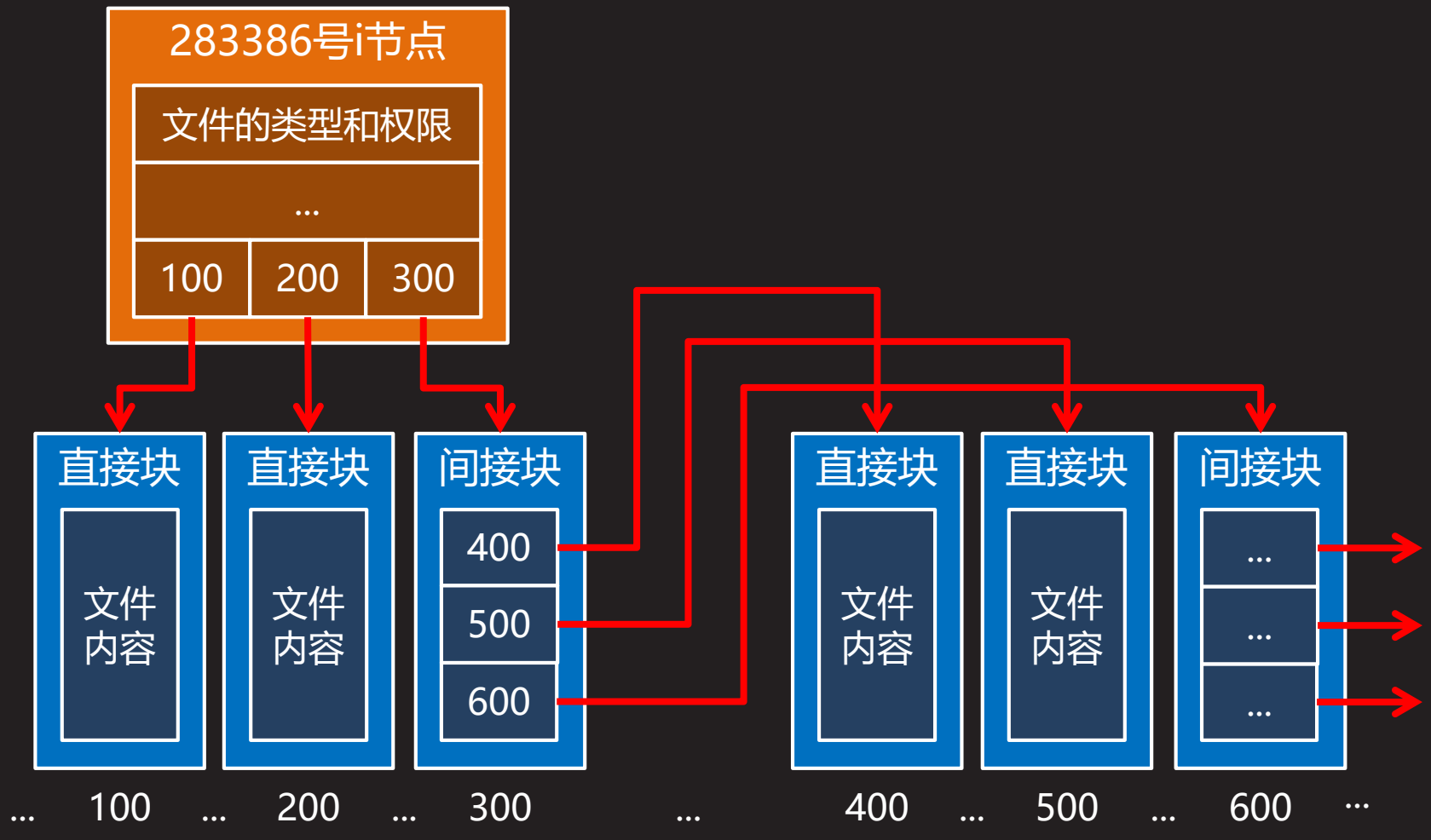
Technology Tarena 达内科技

- i节点，即索引节点(index node, inode)
  - i节点的具体内容包括
    - 文件类型和权限
    - 文件的硬链接数
    - 文件的用户和组
    - 文件的字节大小
    - 文件的最后访问时间、最后修改时间和最后状态改变时间
    - 文件数据块索引表
- 数据块(data block)，512/1024/4096字节
  - 直接块：存储文件的实际内容数据
    - 文件块：存储普通文件的内容数据
    - 目录块：存储目录文件的内容数据
  - 间接块：存储下级文件数据块索引表

知识讲解

逻辑结构（续4）

- 通过i节点索引数据块

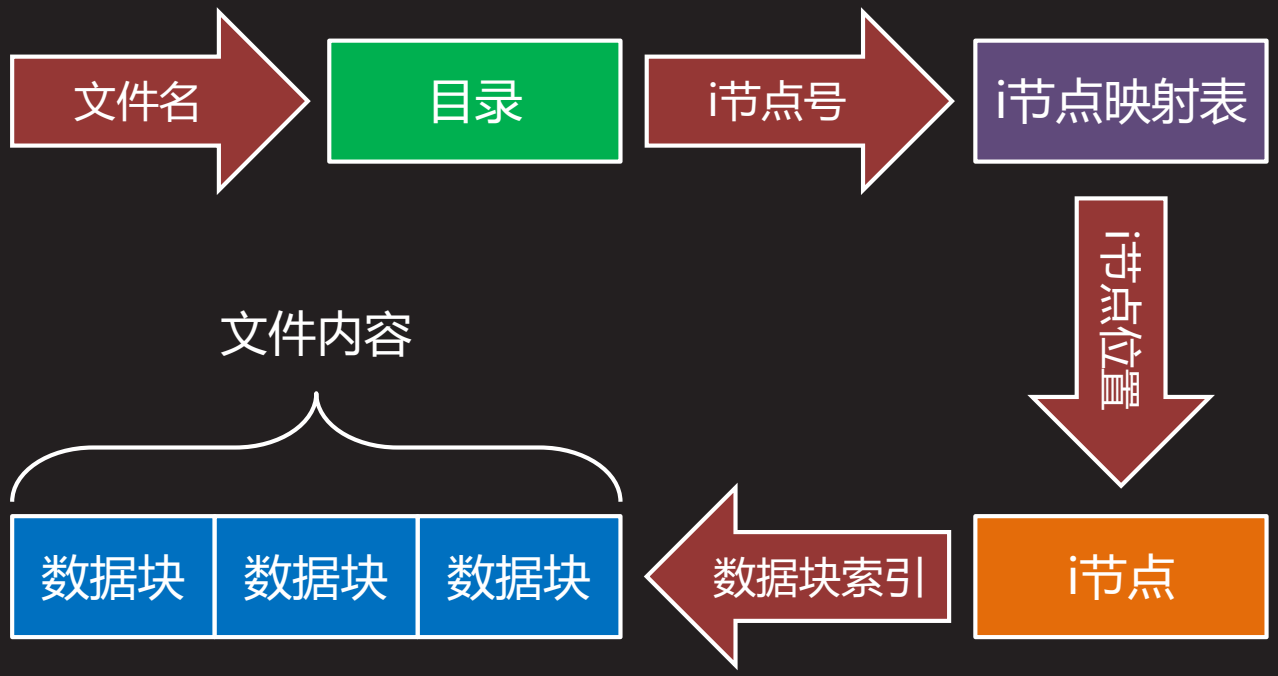


知识讲解

逻辑结构（续5）

Technology Tarena 达内科技

- 文件访问流程
  - 针对给定的文件名，从其所在目录中可以得到与之对应的i节点号，再通过i节点映射表可以查到该i节点在磁盘上的具体位置，读取i节点信息并从中找到数据块索引，进而找到相应的数据块，最终获得文件的完整内容



Technology Tarena 达内科技

文件

普通文件

Technology Tarena 达内科技

知识讲解

- 在Unix/Linux系统中通常所见到的文件，如用C/C++语言编写的源代码文件，编译器、汇编器和链接器产生的汇编文件、目标文件和可执行文件，各种系统配置文件，Shell脚本文件，包括音频、视频等数字内容的多媒体文件，乃至包括数据库在内的各种应用程序所维护、管理和使用的数据文件等，都是普通文件
- 一个普通文件包含以线性字节数组方式组织的数据，通常也称为字节流，Unix/Linux文件没有更进一步的组织结构或格式，因此也不存在类似VMS系统中记录的概念
- 文件中的任何字节都可以被读或者写，这些操作皆始于某个处于特定位置的字节，该位置即所谓当前文件偏移

普通文件（续1）

Technology Tarena 达内科技

- 当前文件偏移的最大值仅受存储该值的C语言变量的数据类型的限制，最新版本的Linux系统已经可以支持到64位
- 组成文件的线性数组里字节的数目，即文件长度或称文件大小，其最大值受限于Linux内核中用于管理文件的C语言代码数据类型的大小，某些文件系统还可能强加自己的限制，将其限定在更小的值
- 操作系统内核并没有对并发文件访问强加任何限制，不同的进程能够同时读写同一个文件，并发访问的结果取决于独立操作的顺序，且通常是不可预测的
- 一个文件包括两部分数据，一部分是元数据，如文件的类型、权限、大小、用户、组、各种时间戳等，存储在i节点中，另一部分是内容数据，存储在数据块中

知识讲解

目录文件

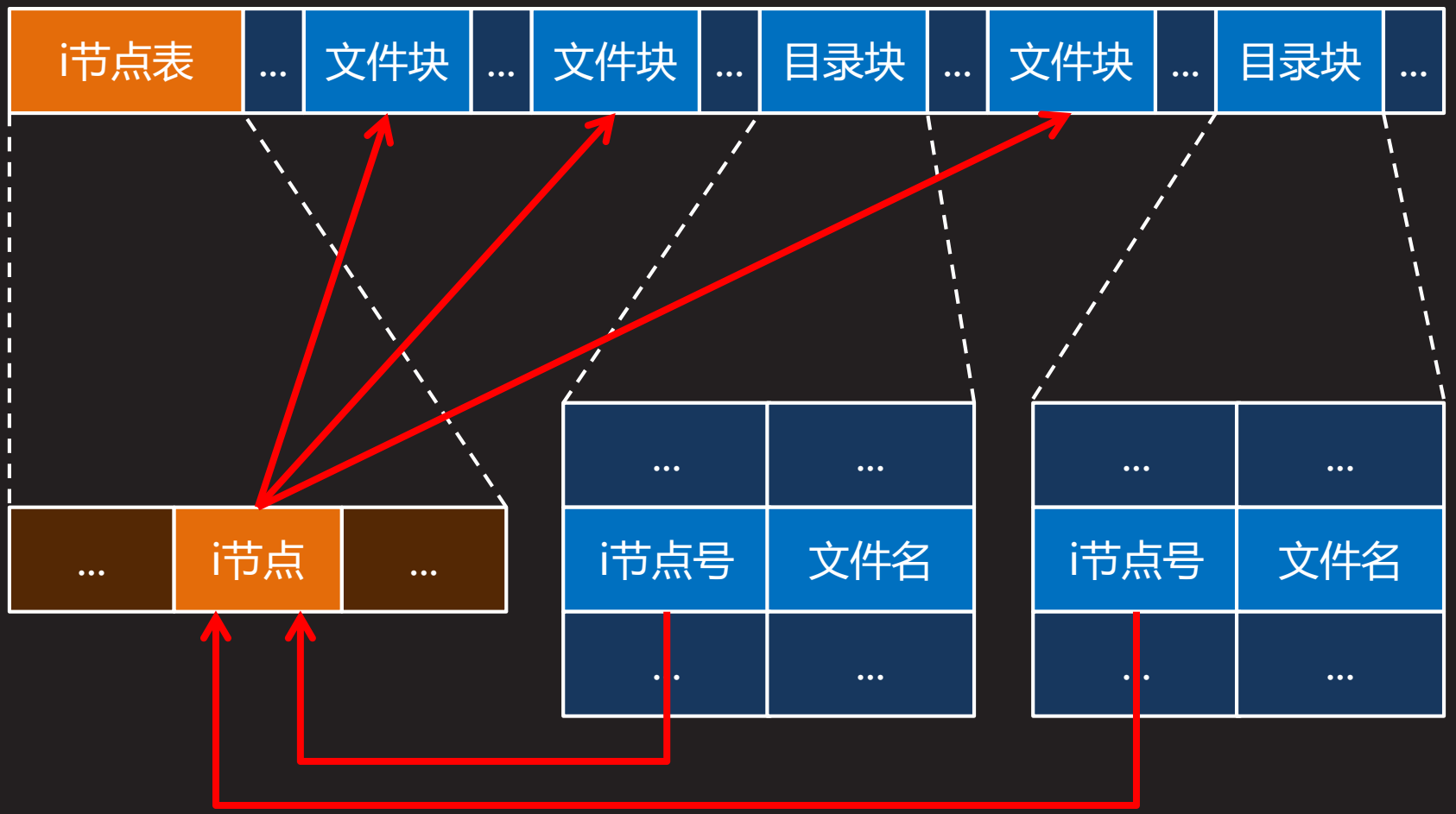
Technology Tarena 达内科技

- 系统通过i节点号唯一地标识一个文件的存在，但人们更愿意使用有意义的文件名来访问文件，目录就是用来建立文件名和i节点号之间的映射的
- 目录的本质就是一个普通文件，与其它普通文件唯一的区别就是它仅仅存储文件名和i节点号的映射，每一个这样的映射，用目录中的一个条目表示，谓之硬链接
- 既然目录也是文件，那么它同样也有自己的i节点，每个目录的i节点号和它的文件名(目录名)之间的映射，记录在它的父目录中，以此类推，形成了一棵目录树
- 根目录的i节点在i节点表中的存储位置是固定的，因此即使没有父目录，根目录也能被正确检索

知识讲解

目录文件（续1）

Technology Tarena 达内科技



目录文件（续2）

Technology Tarena 达内科技

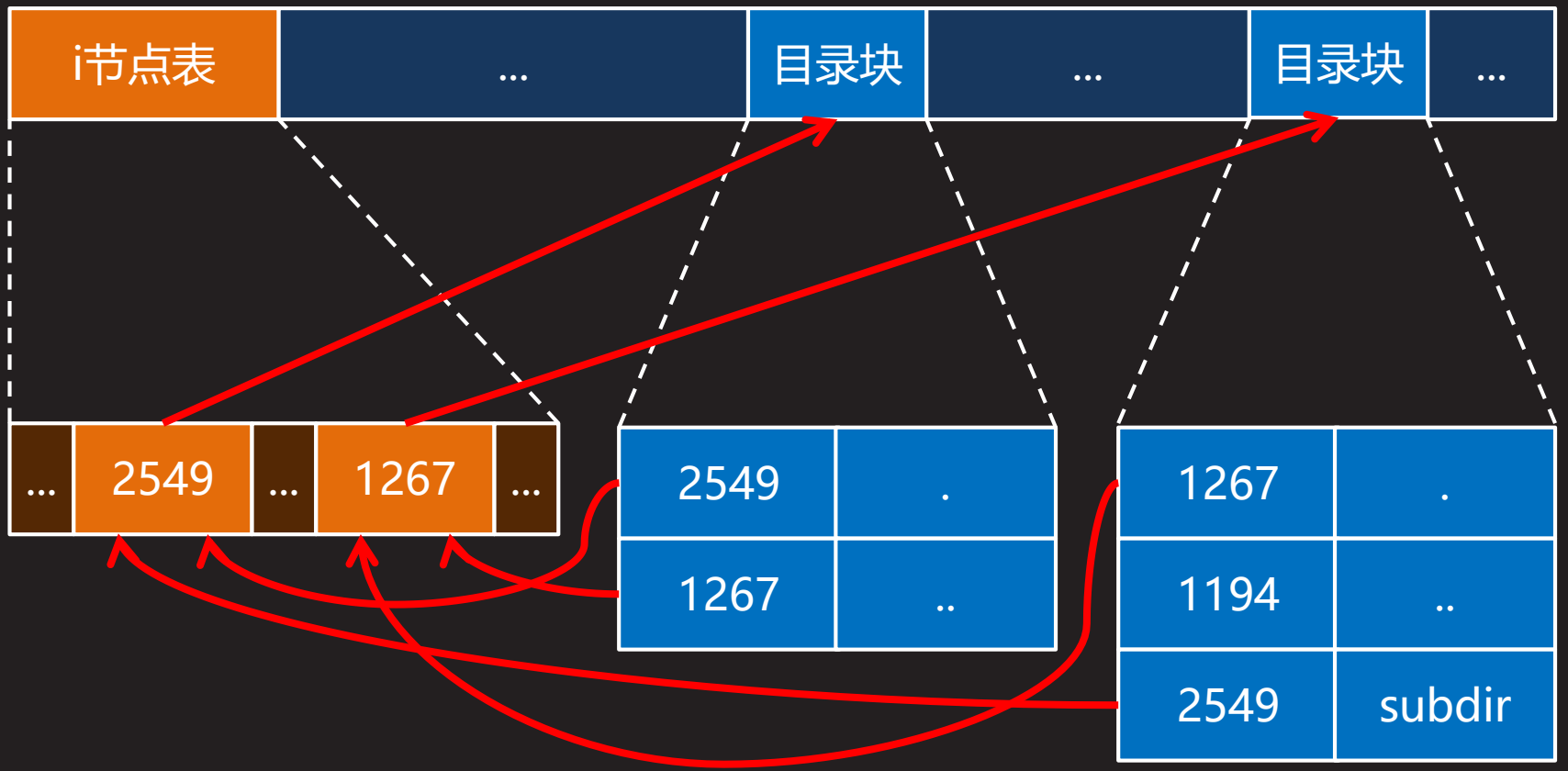
知识讲解

- 当系统内核打开类似“/home/tarena/unixc/day01.txt”这样的路径时，它会从根目录开始遍历路径中的每一个目录项来查找下一项的i节点号。根目录的i节点可以直接拿到，这样就可以在根目录中找到home目录的i节点号，然后在home目录中找到tarena目录的i节点号，再在tarena目录中找到unixc目录的i节点号，最终在unixc目录中找到day01.txt文件的i节点号并打开该文件
- 如果路径字符串的第一个字符不是“/”，则表示相对路径，基于相对路径的路径解析从当前工作目录开始
- 每个目录中都有两个特殊的条目“.”和“..”分别映射该目录本身和其父目录的i节点号，根目录没有父目录，故其“..”和“.”一样都映射到根目录本身

目录文件（续3）

Technology Tarena 达内科技

知识讲解



符号链接文件

Technology Tarena 达内科技

- 符号链接文件看上去象普通文件，每个符号链接文件都有自己的i节点和包含被链接文件完整路径名的数据块
- 符号链接可以指向任何地方，包括不同文件系统上的文件和目录，甚至不存在的文件和目录(坏链接也是链接)
- 相比于符号链接，硬链接不能跨越文件系统(因为i节点号仅在自己的文件系统内有意义)，且其链接目标必须存在
- 相比于硬链接，符号链接的解析需要更大的系统开销，因为有效地解析符号链接至少需要解析两个文件，符号链接文件本身和它所链接的文件
- 文件系统会为硬链接维护链接计数，但不会为符号链接维护任何东西，因此相较于硬链接符号链接缺乏透明性

知识讲解

特殊文件

Technology Tarena 达内科技

- 特殊文件是以文件形式表示的内核对象，具体包括
  - 本地套接字
    - 套接字是网络编程的基础，本地套接字是其面向本机通信的一个变种，它需要依赖文件系统中的一种特殊文件——本地套接字文件
  - 字符设备
    - 设备驱动将字节按顺序写入队列，用户程序从队列中按其被写入的顺序将字节依次读出，如键盘
  - 块设备
    - 设备驱动将字节数组映射到可寻址的设备上，用户程序可以任意顺序访问数组中的任意字节，如硬盘
  - 有名管道
    - 有名管道是一种以文件描述符为信道的进程间通信(IPC)机制，即使是不相关的进程也能通过有名管道文件交换数据

特殊文件（续1）

Technology Tarena 达内科技

- 用ls -l命令查看文件的类型

```
$ ls -l /etc
```

```
-rw-r--r-- 1 root root 1916  9月 14  2013 /etc/passwd
drwxr-xr-x 2 root root 4096  4月 23  2012 opt
```

  - -：普通文件
  - d：目录
  - s：本地套接字
  - c：字符设备
  - b：块设备
  - l：符号链接
  - p：有名管道

知识讲解

Technology Tarena 达内科技

总结和答疑