作业讲解

2024年5月16日 9:27

给定一个数组，将数组向左旋转k位.

```
void rotateLeft(int arr[], int n, int k);

输入：[0, 1, 2, 3, 4, 5, 6, 7, 8], k = 0
输出：[0, 1, 2, 3, 4, 5, 6, 7, 8]

输入：[0, 1, 2, 3, 4, 5, 6, 7, 8], k = 3
输出：[3, 4, 5, 6, 7, 8, 0, 1, 2]

输入：[0, 1, 2, 3, 4, 5, 6, 7, 8], k = 10
输出：[1, 2, 3, 4, 5, 6, 7, 8, 0]
```

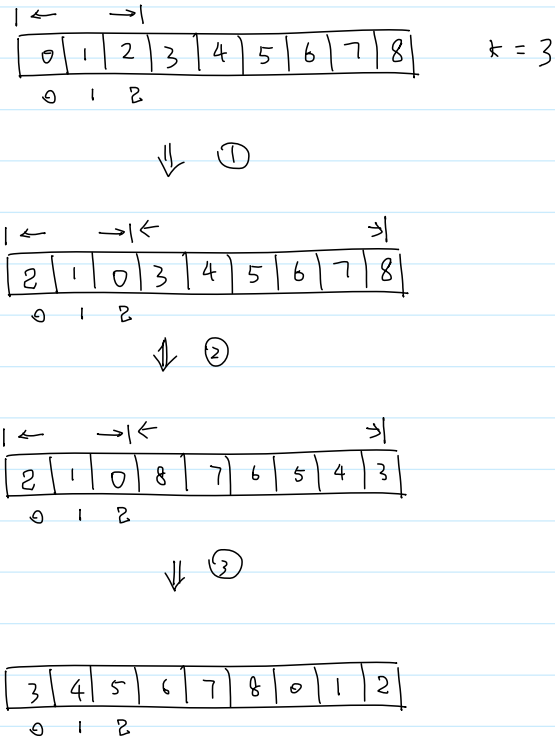
解法一：临时数组

时间复杂度：2n（赋值） O(n)

空间复杂度：O(n)

解法二：反转

```
void reverse(int arr[], int left, int right) {
    // [left, right]
    while (left < right) {
        int tmp = arr[left];
        arr[left++] = arr[right];
        arr[right--] = tmp;
    }
}

void rotateLeft(int arr[], int n, int k) {
    k = k % n;
    if (k == 0) return ;
    reverse(arr, 0, k - 1);
    reverse(arr, k, n - 1);
    reverse(arr, 0, n - 1);
}
```



三次反转 ⟶ 旋转

时间：$\frac{3}{2}n \times 2 = 3n$（赋值） O(n)

空间：O(1)

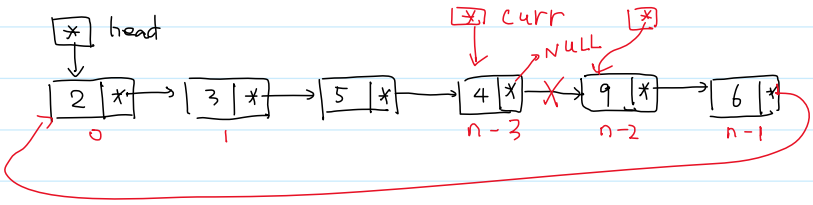
给定一个链表，将链表向右旋转 k 位

```
Node* rotateRight(Node* head, int k);
```

输入：2 —> 3 —> 5 —> 4 —> 9 —> 6，k = 0
输出：2 —> 3 —> 5 —> 4 —> 9 —> 6

输入：2 —> 3 —> 5 —> 4 —> 9 —> 6，k = 2
输出：9 —> 6 —> 2 —> 3 —> 5 —> 4

输入：2 —> 3 —> 5 —> 4 —> 9 —> 6，k = 9
输出：4 —> 9 —> 6 —> 2 —> 3 —> 5

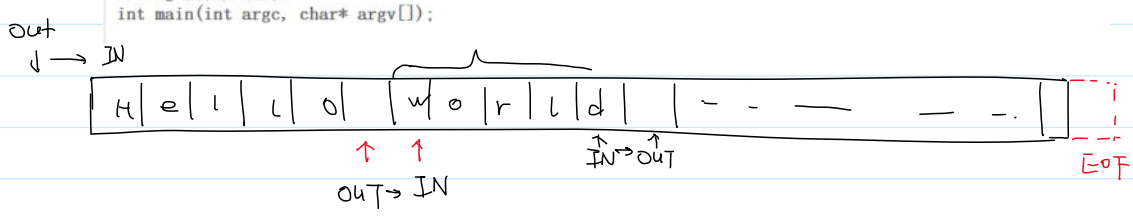


k = 2

```
Node* rotateRight(Node* head, int k) {
    // 边界条件
    if (head == NULL || head->next == NULL) return head;

    // 遍历链表，求链表的长度。
    Node* curr = head;
    int n = 1;
    while (curr->next != NULL) {
        curr = curr->next;
        n++;
    } // curr->next == NULL

    k = k % n;
    if (k == 0) return head;

    curr->next = head; // 构成了循环链表

    // 查找索引为n-k-1的结点
    curr = head;
    for(int i = 0; i < n - k - 1; i++) {
        curr = curr->next;
    } // i == n-k-1

    Node* ret = curr->next;
    curr->next = NULL;
```

'wc' (word count)是Unix下的一个工具，它可以统计一个文本文件中字符的个数(也统计不可打印字符和空白字符)，单词的个数 (单词与单词之间以空白字符分隔) 以及行数。请实现一个 'wc' 程序，当传入的参数个数不对时，请给出提示信息。

```
// 使用方式：./wc file
// argc的值应该为2
int main(int argc, char* argv[]);
```



(1) OUT → IN (✓)

(2) IN → OUT

1. 假定一个数组有 n 个元素，这 n 个元素各不相同，并且循环有序，请找出最小的那个元素。

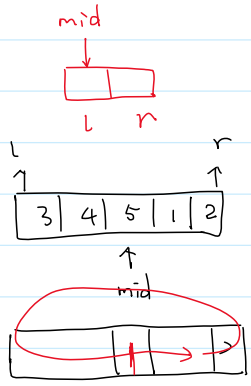
```
int findMin(int arr[], int n);
```

示例1：
输入：[3, 4, 5, 1, 2]
输出：1

示例2：
输入：[4,5,6,7,0,1,2]
输出：0

```
int findMin(int arr[], int n) {
    int left = 0, right = n - 1;

    while (left < right) {
        int mid = left + (right - left >> 1);

        if (arr[mid] < arr[right]) {
            right = mid;
        } else {
            left = mid + 1;
        }
    } // left == right

    return arr[left];
}
```

int mid = r - (r - l >> 1).



a. 查找一个数组中第k小的元素 (思考: 能否将时间复杂度降低到O(n)呢)

```
int find_kth_minimum(int arr[], int n, int k);
```

b. 给定一个无序数组，求它的中位数

```
int find_median(int arr[], int n);
```

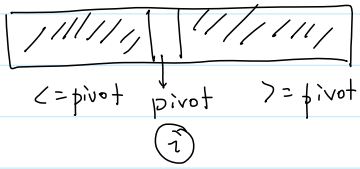
输入：[3, 1, 2]
输出：2

输入 [4, 1, 3, 2]
输出：2

Selection (选择) → O(n)

思路一：先排序 O(nlogn)

思路二：分区 + 二分查找



① i < k-1, (i, n)

② i > k-1, [0, i)

③ i == k-1, return pivot.

$n + \frac{n}{2} + \frac{n}{4} + \cdots + 1 \approx 2n$ O(n)

(a) 判断一个整数是不是丑数 (质因素只包含2, 3, 5的整数，比如：2, 3, 4, 5, 6, 8, 9, 10, ...)

```
bool isUglyNumber(long long n);
```

(b) (拓展题，不要求每个同学都做) 生成前 n 个丑数。

```
long long* generateUglyNumbers(int n);
```

丑数、指数

```cpp
bool isUglyNumber(long long n) {
    if (n <= 1) return false;

    while (n % 2 == 0) {
        n /= 2;
    }
    while (n % 3 == 0) {
        n /= 3;
    }
    while (n % 5 == 0) {
        n /= 5;
    }

    return n == 1;
}
```

思路一：暴力破解 Brute Force 、 n = 1000, 2000

int count = 0

while (count < n) {

}

(X) 指数级别

思路二：生成法：根据前面的丑数，生成下一个丑数。

12 → 只有质因子2

8

3 → 如有质因子3，可以包含质因子2.

6 9 12

5 → 也有质因子5，可以包含质因子2和3.

10 15 20 25

2 3 4 5

⇒ O(n) 时间复杂度

O(n) 空间复杂度

思路三：

① ② ③ ④ ⑤ ⑥ ⑧ ---

i5 ↑ i3 ↑ i2 ↑

1×2 = 2 | 2×2 = 4 | 2×2 = 4 | 3×2 = 6 | 3×2 = 6 | 4×2 = 8

1×3 = 3 | 1×3 = 3 | 2×3 = 6 | 2×3 = 6 | 2×3 = 6 | 3×3 = 9

1×5 = 5 | 1×5 = 5 | 1×5 = 5 | 1×5 = 5 | 2×5 = 10 | 2×5 = 10

时间：O(n)

空间：O(1)

chmod

2024年5月16日 11:29

Change mode

↳ 权限

chmod 目录的用法如下：

1. 文字设定法（较少使用）

```
chmod [ugoa][+=-][rwx] file/dir
    u: user
    g: group
    o: other
    a: all
    +: 添加权限
    -: 删除权限
    =: 将权限设置为
```

2. 数字设定法（常用）

```
chmod mode file/dir
    mode: 三位八进制数字
```

文件创建掩码

2024年5月16日 11:35

umask
↳ unix

```
he@he-vm:~/cpp58/Linux04$ umask → 想去掉的权限
0002
```

文件：664　　666 − 002 = 664
目录：775　　777 − 002 = 775　(x)

```
he@he-vm:~/cpp58/Linux04$ umask 0023
he@he-vm:~/cpp58/Linux04$ umask
0023
```

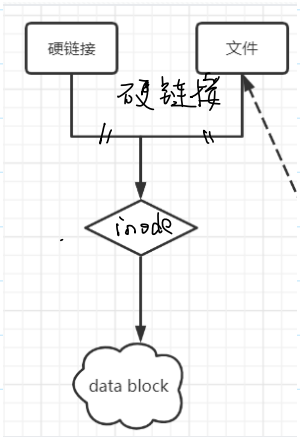
文件：644　　666 − 023 = 643　(x)
目录：754　　777 − 023 = 754

666 & (~umask)
777 & (~umask)　(√)

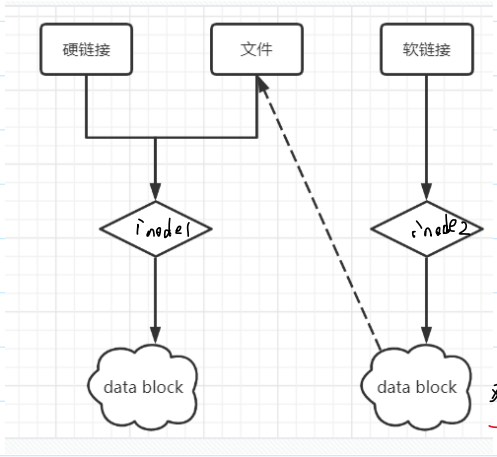
链接

2024年5月16日 14:29

#1. hard link



#2. Symbolic link.



C语言中指针

文件的路径.

text1

```
he@he-vm:~/cpp58/Linux04$ ln -s text1 link
he@he-vm:~/cpp58/Linux04$ ls -li
total 0
1488618 lrwxrwxrwx 1 he he 5  5月 16 14:38 link -> text1
1488617 -rw-r--r-- 1 he he 0  5月 16 14:36 text1


1488619 lrwxrwxrwx 1 he he 18  5月 16 14:41 link2 -> /foo/bar/not_exist
```

→悬空指针(野指针)

dangling link

悬空链接

scp

2024年5月16日 14:46

scp: secure copy (ssh)

远程复制 (上传、下载)

上传: 本地 $\xrightarrow{cp}$ 远程

下载: 远程 $\xrightarrow{cp}$ 本地.

本地路径 { 绝对路径
相对路径.

远程路径 用户名@ip: 绝对路径

```
D:\Class>scp 词法分析器.zip he@192.168.76.128:~
```

上传

```
D:\Class>scp -r he@192.168.76.128:~/cpp58 .
```

下载

打包和压缩

2024年5月16日 14:59

tar (text archieve)

↳ 打包

打包压缩

我们可以用 tar 命令打包和压缩文件。tar 是一个非常有历史的工具，这里我们只介绍它的传统(经典)用法：

格式：

tar [主选项+辅选项] 包名 [文件或目录]...

主选项(有且只能选择其中一个)：

- c：创建
- r：追加
- x：释放
- t：查看

辅选项：

- f：指定包文件的名称
- v：显示详细信息 → verbose
- z：使用gzip算法压缩或解压缩包文件

gcc

2024年5月16日 16:01

-E：预处理后的文件

-S：汇编代码

-c：目标文件 （compile，产生.o的编译）

无：可执行程序

-Wall：warning all

-O0，-O1，-O2，-O3：优化级别 Optimize：优化

-O0 ⇩ 开发 -O1，-O2，-O3 ⇩ 生产

-g：产生调试信息

-Dmacro：在命令行开关 #define macro

```
$ gcc -E main.c -o main.i -DDEBUG
```

main.c

```c
#include <func.h>

int main(int argc, char* argv[])
{
    int i;
#ifdef DEBUG
    printf("i = %d\n", i);
#endif
    return 0;
}
```

-Dmacro=value：在命令行开关 #define macro value

```
$ gcc main.c -o main -DMAXSIZE=10086
```

main.c

```c
#include <func.h>

#ifndef MAXSIZE
#define MAXSIZE 4096
#endif

int main(int argc, char* argv[])
{
    printf("MAXSIZE = %d\n", MAXSIZE);
    return 0;
}
```

-Idir：指定头文件所在的目录

#include <...>：dir → 系统的头文件包含目录

#include "..."：dir → 当前工作目录 → 系统的头文件包含目录

```
$ gcc main.c -o main -Iheader
```

条件编译

2024年5月16日 16:33

① #if ... #else ... #endif → 运算符 defined.

② #ifdef ... #else ... #endif

③ #ifndef ... #else ... #endif

作用.

(1) 编写可移植代码

(2) 给宏提供默认值

(3) 调试式

(4) 避免头文件重复包含.

foo.h

```
1  #ifndef __WD_FOO_H
2  #define __WD_FOO_H
3
4  struct Foo {
5      int a;
6      int b;
7      int c;
8  };
9
10 void foo(void);
11
12 #endif
```

⇒ 和头文件的名字相对应.

#include "foo.h" (✓)

#include "foo.h" (✗)

#include "foo.h" (✗)

gdb

2024年5月16日 17:17

观念：

① 调试程序的难度是写代码的两倍

② 调试程序

走一走

停一停

看一看 （看程序的状态是否和预期的一致）

↳ 不一致 { 预期错了：加深对问题的理解

程序错了：线索 → 指向其正当错的地方

```
$ gcc main.c -o main -g
```

```
$ gdb main
```

↳ 可执行程序的名字

```
$ gdb --args main aaa bbb ccc
```

⇒ 命令行参数

```
(gdb) set args aaa bbb ccc
```

当命令行参数

```
(gdb) quit
```

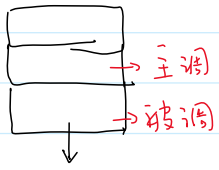
走一走：run/r

next

step / finish

↳ 执行完这次函数调用，回到主调函数

continue



停一停：

Enable

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000000011a8 in main
                                                   at main.c:14
2       breakpoint     keep y   0x0000000000001151 in foo
                                                   at main.c:4
```

```
(gdb) delete 1
(gdb) delete
Delete all breakpoints? (y or n) y
```

next：执行完整个函数调用.

step / finish

continue：执行到逻辑上的下一个断点

continue，我们到这样上的下一个断点

ignore [断点编号] [COUNT]

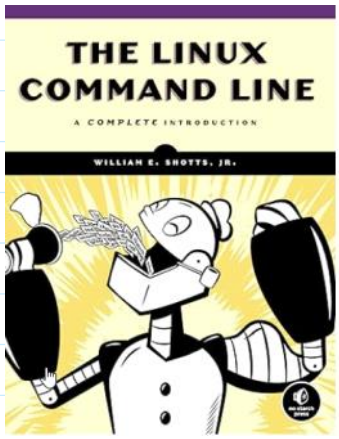
看一看。

推荐书籍

2024年5月16日 21:33

shell 命令：

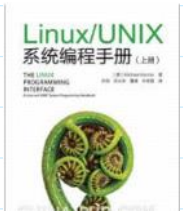
《The Linux Command Line》 ⟷ 《Linux 命令行大全》



Linux 系统编程.

《Linux 系统编程》 Robert Love

《Linux 系统编程接口》 手册



《Unix 环境高级编程》 APUE 手册.

操作系统理论：

必：《操作系统导论》↔《Operating System: Three easy pieces》