2024秋季学期 Linux大作业（研究生）

**作业要求**

请大家认真阅读，按照要求完成：

1. 上机操作，作业以本文档为基础，写下源码，并在文档里提供源码运行后的拷屏结果。可以为doc或者pdf格式。

2. 禁止相互抄袭，被抄袭的题目，无论抄袭者还是被抄袭者，一经发现，都被扣分。

3. 请于2025 年 01 月 01 日之前交作业。[作业提交给 yuandong211@qq.com](mailto:作业提交给yuandong211@qq.com)

4. 为便于整理，email 提交方式 ：

邮件标题必须为 linux大作业-姓名-班级-学号，以文件附件形式提交, 附件的文件名也要按照 linux大作业-姓名-班级-学号 起名字 。 [收件邮箱yuandong211@qq.com设立了自动回复功能，如果收不到自动回复，请联系老师](mailto:邮件设立了自动回复功能，如果收不到自动回复，请联系老师（yuandong@bupt.edu.cn) 。

1. 作业只允许提交一次，不要提交多次。

**前置条件**

准备一个搭载Ubuntu系统or CentOS的Bash的Linux运行环境，如云服务器、VMware虚拟机。

1. 基础命令
   1. 查看系统的发行版本和系统架构。

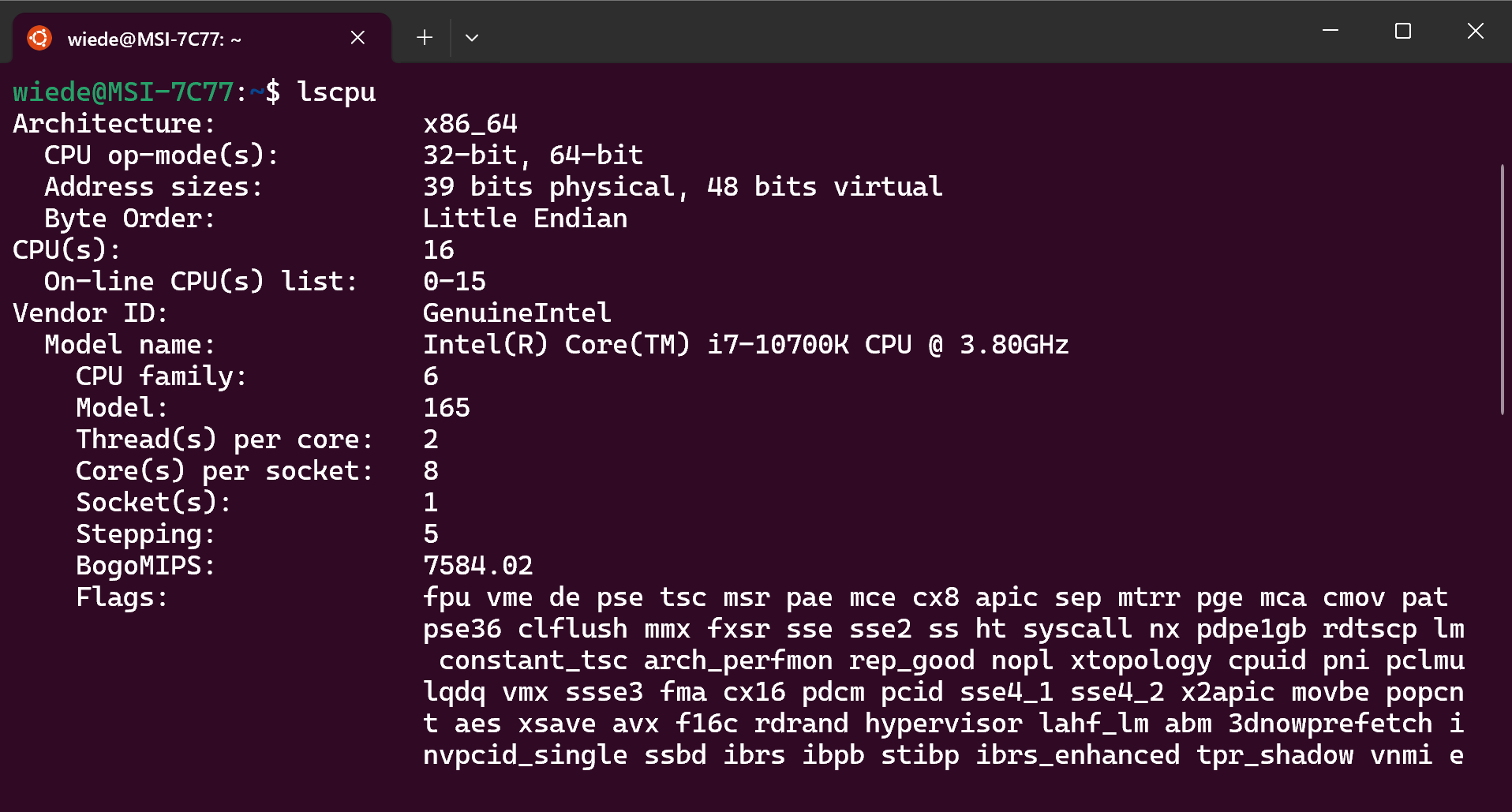
lsb\_release -a

文本

描述已自动生成

* 1. 查看CPU信息，指出CPU个数、单CPU核心数、总CPU逻辑核心数，并说明它们之间的计算关系。

lscpu



* **CPU(s):** 16 --- 表示总的逻辑CPU核心数 (包括超线程)
* **Thread(s) per core:** 2 --- 表示每个物理核心的线程数
* **Core(s) per socket:** 8 --- 表示每个CPU 插槽（Socket）的核心数
* **Socket(s):** 1 --- 表示CPU插槽数，也即物理CPU个数

关系：单CPU时，总逻辑核心数 = 核心数 \* 单核心线程数

* 1. 查看当前用户相关的所有进程

ps aux | grep $USER

文本

描述已自动生成

* 1. 每秒查看一次的GPU使用情况

watch -n 1 nvidia-smi

文本

描述已自动生成

说明：WSL下nvidia-smi无法查看进程

* 1. 查看磁盘使用情况，查看网络配置并说明IP地址，查看SSH端口号
* 查看磁盘：df -h  
  文本

  描述已自动生成
* 查看网络配置：ip addr show

文本

描述已自动生成

IP: 172.28.254.20@eth0

* 查看SSH端口：sudo ss -tlnp | grep sshd

文本

描述已自动生成

说明：主机MSI-7C77没有配置OpenSSH Server，故临时更换主机演示

* 1. 使用curl或wget命令连接北邮校园网，校园网地址为“http://10.3.8.211/login”（可以将密码信息打码）

curl 'http://10.3.8.211/login' --data 'user=2024141187&pass=xxxxxx'

文本

描述已自动生成

* 1. 以易读的方式查看系统运行内存使用和存储空间的使用量

htop

电脑萤幕画面

描述已自动生成

* 1. 在当前目录下不区分大小写查找文件名中带utils的文件。

find . -iname "\*utils\*"

文本

描述已自动生成

* 1. 在当前目录下创建文件夹/work/run，在该文件夹中创建一个文件test.sh，并修改权限：将读写执行权限分配给用户所有者，将读取执行权限分配给文件所属组和其他用户。执行test.sh并将输出内容重定向到log.txt文件中。

mkdir -p ./work/run # 创建/work/run目录，-p选项可以递归创建目录

touch ./work/run/test.sh # 创建test.sh文件

nano nano ./work/run/test.sh

test.sh:

#!/bin/bash

echo "Hello, world!"

date  # 输出当前日期和时间

chmod 755 ./work/run/test.sh # 设置权限：用户rwx (7)，组和其他用户rx (5)

./work/run/test.sh > ./work/run/log.txt # 执行test.sh并将输出重定向到log.txt

文本

描述已自动生成

* 1. 创建并进入目录linux\_project，在linux\_project下创建若干后缀为txt、png的文件。完成以下操作：统计所有文件数量、统计后缀为png的文件数量、将所有png文件打包为images.tar文件、删除所有后缀为png的文件、解压images.tar包。

mkdir linux\_project

cd linux\_project

touch file1.txt file2.txt image1.png image2.png image3.png

ls | wc -l

文本

描述已自动生成

ls \*.png | wc -l

文本

描述已自动生成

tar -cvf images.tar \*.png

文本

描述已自动生成

rm \*.png

tar -xvf images.tar

文本

描述已自动生成

1. 字符串操作
   1. 创建一个名为poem.txt的文本文件，并在其中输入以下内容：

In the heart of the night, when the world is asleep

In the heart of the day, where the shadows creep

In the silence, a whisper, a secret to keep

In the silence, a promise, a truth to believe

In the heart of the journey, we find what we seek

But sometimes the path leads us back where we began

nano poem.txt

文本

描述已自动生成

(1) 使用适当的命令将下面的两行添加到poem.txt文件中，同时确保任何产生的错误信息都被重定向到/dev/null，以避免干扰：

In the heart of the dream, where the lost can be found

In the heart of the silence, where whispers are loud

{

echo "In the heart of the dream, where the lost can be found"

echo "In the heart of the silence, where whispers are loud"

} >> poem.txt 2>/dev/null

文本

描述已自动生成

（2）使用sed或awk命令将poem.txt中的所有单词“heart”替换为“core”，并将结果保存到新的文件modified\_poem.txt中。

sed 's/heart/core/g' poem.txt > modified\_poem.txt

文本

描述已自动生成

（3）利用grep、wc以及sort命令组合，统计modified\_poem.txt文件中每个单词的出现次数（忽略大小写），并按出现次数降序排序。将排序结果保存到word\_count.txt文件中，输出格式应为word: count的形式。

tr '[:space:]' '[\n\*]' < modified\_poem.txt |

tr '[:upper:]' '[:lower:]' |

grep -v '^$' |

sort |

uniq -c |

sort -nr |

awk '{print $2 ": " $1}' > word\_count.txt

文本

描述已自动生成

* 1. 假设有一个名为 1.txt 的文件，内容如下所示：

BUPT!THU!BUAA!PKU!BNU 10.168.192.125 213.140.16.1 2023141061

BUPT!THU!BUAA!PKU!BNU 10.168.192.123 213.140.16.1 2023141063

BUPT!THU!BUAA!PKU!BNU 10.168.192.122 213.140.16.1 2023141062

BUPT!THU!BUAA!PKU!BNU 10.168.192.124 213.140.16.1 2023141062

BUPT!THU!BUAA!PKU!BNU 10.168.192.124 213.140.16.1 2023141062

BUPT!THU!BUAA!PKU!BNU 10.168.192.121 213.140.16.1 2023141065

BUPT!THU!BUAA!PKU!BNU 10.168.192.126 213.140.16.1 2023141064

请使用Linux命令完成以下任务，每个任务均需使用一行命令实现：

（1）将 1.txt 文件中的所有感叹号 ! 替换为井号 #，并将结果输出到屏幕上。

sed 's/!/\#/g' 1.txt

文本

中度可信度描述已自动生成

（2）对 1.txt 文件中的第2至第5行按IP地址字段（即第三列）进行升序排序，并将排序后的结果输出到屏幕上。

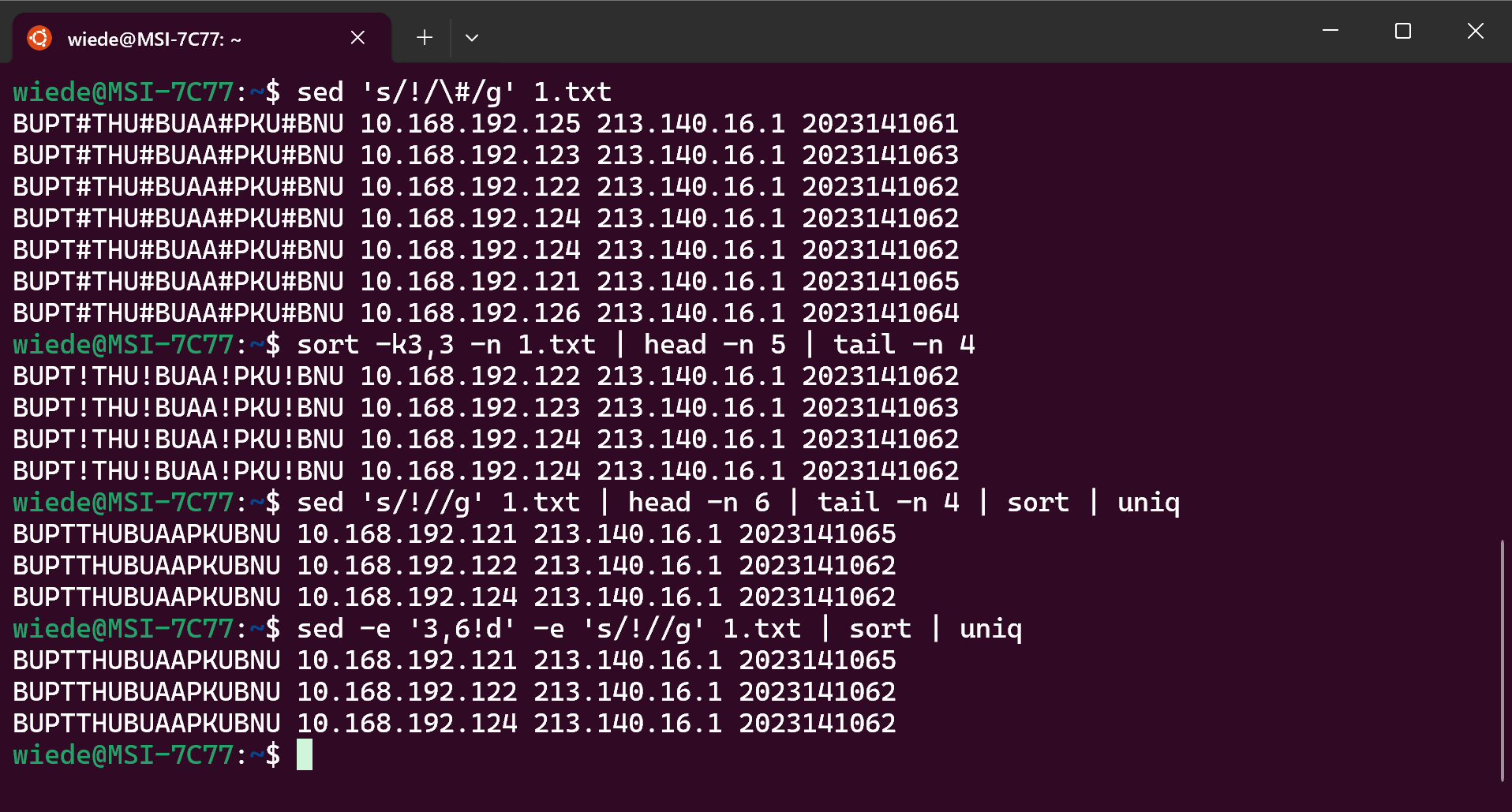
sort -k3,3 -n 1.txt | head -n 5 | tail -n 4

图形用户界面

中度可信度描述已自动生成

（3）统计并删除 1.txt 文件中第3至第6行的重复行，同时删除每行中的所有感叹号 !。最后，将处理后的内容输出到屏幕上。

sed -e '3,6!d' -e 's/!//g' 1.txt | sort | uniq



1. Shell脚本
   1. 文件命名批量修改

任务要求：

编写一个Shell脚本 add\_numbering.sh，实现以下功能：

给指定目录中的所有文件（不包括子目录）添加编号，编号从1开始；

格式为：编号\_{原始文件名}，编号固定为3位，不足补零，例如001\_file.txt；保持文件的扩展名不变。

脚本接受以下参数：

目录路径：待处理的文件所在目录；

脚本运行方式为：

./add\_numbering.sh <directory>

重命名规则：

按照文件名字母顺序（不区分大小写）处理文件；

输出每次操作的原始文件名和新文件名。

示例：

假设目录 /home/user/docs 中有以下文件：

summary.pdf

notes.txt

report.docx

运行命令：

./add\_numbering.sh /home/user/docs

重命名后，目录中的文件为：

001\_notes.txt

002\_report.docx

003\_summary.pdf

脚本输出：

Renaming 'notes.txt' -> '001\_notes.txt'

Renaming 'reports.txt' -> '002\_reports.txt'

Renaming 'summary.txt' -> '003\_summary.txt'

Renaming completed!

add\_numbering.sh:

#!/bin/bash

# 检查参数

if [ $# -ne 1 ]; then

    echo "Usage: $0 <directory>"

    exit 1

fi

directory="$1"

# 检查目录是否存在且可访问

if [ ! -d "$directory" ] || [ ! -r "$directory" ]; then

    echo "Error: Directory '$directory' does not exist or is not accessible"

    exit 1

fi

# 进入目标目录

cd "$directory" || exit 1

# 获取所有文件并排序（不包括子目录）

files=($(find . -maxdepth 1 -type f -not -name ".\*" -printf "%f\n" | sort -f))

# 计数器

count=1

# 处理每个文件

for file in "${files[@]}"; do

    # 生成新文件名（3位数字编号）

    new\_name=$(printf "%03d\_%s" "$count" "$file")

    # 执行重命名并输出

    if [ "$file" != "$new\_name" ]; then

        mv "$file" "$new\_name"

        echo "Renaming '$file' -> '$new\_name'"

    fi

    ((count++))

done

echo "Renaming completed!"

文本

描述已自动生成

* 1. 给定一个英文短语，将其转化为首字母的缩写词。

示例：

1. As Soon As Possible 缩写为 ASAP

2. Also Known as 缩写为 AKA

3. Frequently Asked Questions 缩写为 FAQ

题目要求：

1. 编写脚本acronym.sh，实现将短语转化为首字母缩写词的功能；

2. 脚本接受一个字符串输入，输出对应的小写缩写词，连字符'-'视作单词分隔符，其他符号可以忽略；

3. 使用脚本将 "Laugh Out Loud"，"I Don’t Know" 和 "In My Opinion" 缩写。

$ bash acronym.sh "Laugh Out Loud"

$ LOL

$ bash acronym.sh "I Don't Know"

$ IDK

$ bash acronym.sh "In My Opinion "

$ IMO

acronym.sh:

#!/bin/bash

# 检查是否有输入参数

if [ $# -eq 0 ]; then

    echo "请提供一个短语作为参数"

    exit 1

fi

# 获取输入短语

phrase="$1"

# 使用 tr 将连字符替换为空格，然后用 awk 处理每个单词

# 最后用 tr 转换为大写

echo "$phrase" | tr '-' ' ' | \

awk '{

    result = ""

    for(i=1; i<=NF; i++) {

        # 获取每个单词的第一个字符

        first\_char = substr($i,1,1)

        if (first\_char ~ /[A-Za-z]/) {

            result = result first\_char

        }

    }

    print result

}' | tr '[:lower:]' '[:upper:]'

文本

描述已自动生成

* 1. 给定一串数字，验证它是否符合Luhn公式。

Luhn公式是也称模数10，是一种简单的数学运算，用来验证信用卡编号是否有效。以16位的VISA卡号为例，前6位数字为发卡行识别号，7到15位数字为账户号码，最后一位数字是Luhn校验位，用来进行校验，防止输入错误卡号。

示例：



4539 3195 0343 6467是一串有效的信用卡号，Luhn算法的步骤为：

1. 从右边开始，将每第二个数字加倍（上图箭头所指数字），如果加倍后大于9，则减去9，4539 3195 0343 6467加倍后的结果为8569 6195 0383 3437；

2. 将所有数字相加，如果总和能被10整除，则是有效号码，8+5+6+9+6+1+9+5+0+3+8+3+3+4+3+7=80，所以编号是有效的。

题目要求：

1. 编写脚本validate.sh，验证 8273 1034 7352 0869和6252 4958 6360 3058是否是有效编码，脚本接受输入编号，并输出校验结果；

2. 输入中允许有空格，脚本校验数字前需要将空格删除；

3. 若编号符合Luhn公式，则输出Validation Passed，反之输出Validation Failed。

$ bash validate.sh "8273 1034 7352 0869"  
$ Validation Passed

$ bash validate.sh "6252 4958 6360 3058"

$ Validation Failed

validate.sh:

#!/bin/bash

# 删除输入中的空格

input=$(echo "$1" | tr -d ' ')

# 检查输入是否全为数字

if ! [[ "$input" =~ ^[0-9]+$ ]]; then

    echo "Validation Failed"

    exit 1

fi

# Luhn 算法实现

sum=0

length=${#input}

# 从右到左遍历数字

for (( i=length-1; i>=0; i-- )); do

    digit=${input:i:1}

    # 如果是偶数位（从右边算起），则加倍

    if (( (length - i) % 2 == 0 )); then

        digit=$((digit \* 2))

        # 如果加倍后的数字大于9，则减去9

        if (( digit > 9 )); then

            digit=$((digit - 9))

        fi

    fi

    # 累加

    sum=$((sum + digit))

done

# 检查总和是否能被10整除

if (( sum % 10 == 0 )); then

    echo "Validation Passed"

else

    echo "Validation Failed"

fi

文本

描述已自动生成

* 1. 实现Atbash加密和解密。

Atbash是一种简单的单字母替换密码，它通过翻转字母表，将所有字母进行替换，拉丁字母的替换规则为：

明文：abcdefghijklmnopqrstuvwxyz

密文：zyxwvutsrqponmlkjihgfedcba

其中明文和密文中的字母上下一一对应，数字保持不变。此外，密文用固定的长度的组表示，每组5个字母，组之间由空格分割，所有明文都编码为小写字母。

示例：

1. 明文 'test' 编码得到 'gvhg'；

2. 明文 'x123 yes' 编码得到 'c123b vh'；

3. 密文 'c123b vh' 解码得到 'x123yes'。

题目要求：

1. 编写atbash.sh脚本，实现明文加密和密文解密功能；

2. atbash.sh运行方式为 "./atbash.sh <mode> <text>"，<mode>为 'encode' 时，对<text>进行加密；<mode>为 'decode' 时，对<text>进行解密；

3. 将明文"attack start at 1pm"加密，将密文"fitvm gnvvg rmtlu u"解密。

$ bash atbash.sh encode "attack starts at 1pm"

$ zggzx phgzi gzg1k n

$ bash atbash.sh decode "fitvm gnvvg rmtlu u"

$ urgentmeetingoff

atbash.sh:

#!/bin/bash

# 检查参数个数

if [ $# -ne 2 ]; then

    echo "Usage: $0 <mode> <text>"

    echo "mode: encode or decode"

    exit 1

fi

mode=$1

text=$2

# 定义字母替换函数

atbash\_char() {

    char=$1

    if [[ $char =~ [a-z] ]]; then

        # 获取字母在字母表中的位置(0-25)

        pos=$(printf "%d" "'$char")

        pos=$((pos - 97))

        # 计算替换后的位置

        new\_pos=$((25 - pos))

        # 转换回字符

        printf \\$(printf "%o" $((new\_pos + 97)))

    else

        echo -n "$char"

    fi

}

# 编码函数

encode() {

    # 转为小写并移除空格

    text=$(echo "$1" | tr '[:upper:]' '[:lower:]' | tr -cd '[a-z0-9]')

    result=""

    # 处理每个字符

    for (( i=0; i<${#text}; i++ )); do

        char="${text:$i:1}"

        if [[ $char =~ [0-9] ]]; then

            result+="$char"

        else

            result+=$(atbash\_char "$char")

        fi

        # 每5个字符添加空格

        if [ $(( (i+1) % 5 )) -eq 0 ] && [ $i -lt $((${#text}-1)) ]; then

            result+=" "

        fi

    done

    echo "$result"

}

# 解码函数

decode() {

    # 移除空格

    text=$(echo "$1" | tr -d ' ')

    result=""

    # 处理每个字符

    for (( i=0; i<${#text}; i++ )); do

        char="${text:$i:1}"

        if [[ $char =~ [0-9] ]]; then

            result+="$char"

        else

            result+=$(atbash\_char "$char")

        fi

    done

    echo "$result"

}

# 根据模式执行相应功能

case "$mode" in

    "encode")

        encode "$text"

        ;;

    "decode")

        decode "$text"

        ;;

    \*)

        echo "Invalid mode. Use 'encode' or 'decode'"

        exit 1

        ;;

esac

文本

描述已自动生成

1. Makefile工程
   1. 完成C语言项目，实现数组的排序、反转和求最大值。

现在有一个文件 1.txt ， 内容如下 ：

2

5

1

8

5

16

9

32

22

77

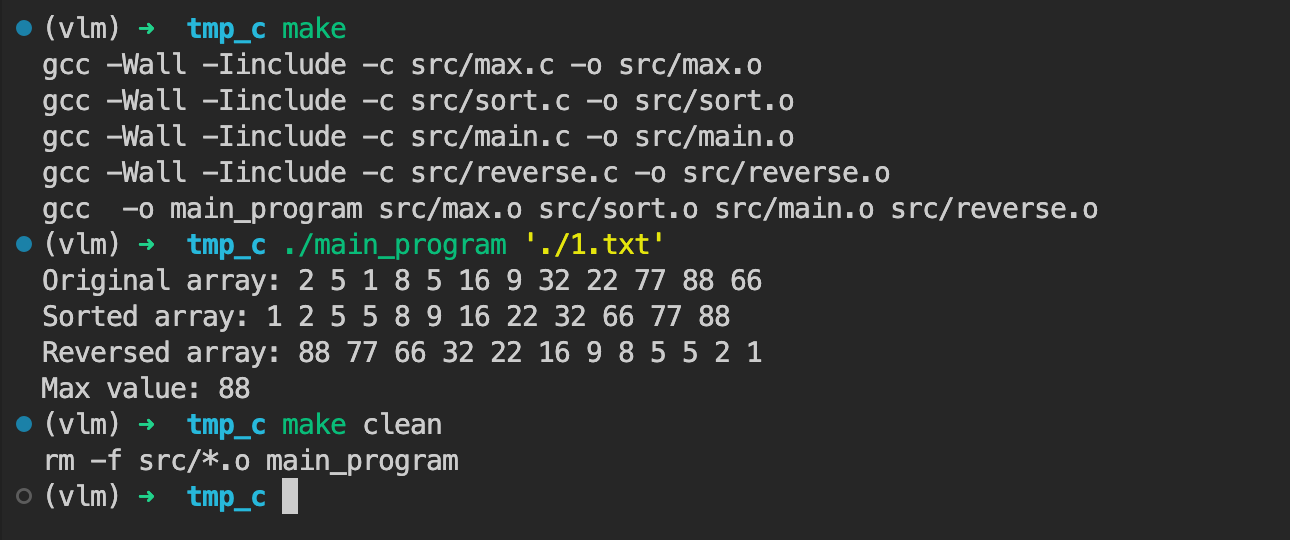
88

66

设计一个含makefile文件的工程，功能为从某个文件中读取数字并存放在一个数组中(使用int即可)，实现对数组排序的函数sort.c；对数组反转的函数reverse.c；以及求取最大值的max.c

1. 写出 main.c, makefile和各个子函数
2. 将.h文件放在include目录中，.c文件放在src目录中
3. 存储有数据的文本文件应该通过命令行参数传入
4. 编译并执行，贴出拷屏的结果

示例运行结果如下(函数的操作不一定要inplace)：



1. 目录结构：

.

├── Makefile

├── include

│ └── array\_ops.h

└── src

├── main.c

├── max.c

├── reverse.c

└── sort.c

2. 头文件include/array\_ops.h

#ifndef ARRAY\_OPS\_H

#define ARRAY\_OPS\_H

void sort\_array(int arr[], int size);

void reverse\_array(int arr[], int size);

int find\_max(int arr[], int size);

#endif

3. src/main.c:

#include <stdio.h>

#include <stdlib.h>

#include "../include/array\_ops.h"

#define MAX\_SIZE 100

int main(int argc, char \*argv[]) {

    if (argc != 2) {

        printf("Usage: %s <filename>\n", argv[0]);

        return 1;

    }

    FILE \*fp = fopen(argv[1], "r");

    if (!fp) {

        printf("Cannot open file %s\n", argv[1]);

        return 1;

    }

    int arr[MAX\_SIZE];

    int size = 0;

    int num;

    while (fscanf(fp, "%d", &num) == 1 && size < MAX\_SIZE) {

        arr[size++] = num;

    }

    fclose(fp);

    // Print original array

    printf("Original array:");

    for (int i = 0; i < size; i++) {

        printf(" %d", arr[i]);

    }

    printf("\n");

    // Sort array

    int sorted[MAX\_SIZE];

    for (int i = 0; i < size; i++) sorted[i] = arr[i];

    sort\_array(sorted, size);

    printf("Sorted array:");

    for (int i = 0; i < size; i++) {

        printf(" %d", sorted[i]);

    }

    printf("\n");

    // Reverse array

    int reversed[MAX\_SIZE];

    for (int i = 0; i < size; i++) reversed[i] = arr[i];

    reverse\_array(reversed, size);

    printf("Reversed array:");

    for (int i = 0; i < size; i++) {

        printf(" %d", reversed[i]);

    }

    printf("\n");

    // Find max

    int max = find\_max(arr, size);

    printf("Max value: %d\n", max);

    return 0;

}

4. src/sort.c:

void sort\_array(int arr[], int size) {

    for (int i = 0; i < size - 1; i++) {

        for (int j = 0; j < size - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

}

5. src/reverse.c:

void reverse\_array(int arr[], int size) {

    for (int i = 0; i < size / 2; i++) {

        int temp = arr[i];

        arr[i] = arr[size - 1 - i];

        arr[size - 1 - i] = temp;

    }

}

6. max.c:

int find\_max(int arr[], int size) {

    int max = arr[0];

    for (int i = 1; i < size; i++) {

        if (arr[i] > max) {

            max = arr[i];

        }

    }

    return max;

}

7. Makefile:

CC = gcc

CFLAGS = -Wall -Iinclude

SRCS = src/main.c src/sort.c src/reverse.c src/max.c

OBJS = $(SRCS:.c=.o)

TARGET = main\_program

$(TARGET): $(OBJS)

    $(CC) -o $(TARGET) $(OBJS)

%.o: %.c

    $(CC) $(CFLAGS) -c $< -o $@

clean:

rm -f src/\*.o $(TARGET)

文本

描述已自动生成

* 1. 完成C语言项目，实现两个矩阵之间的逐元素乘法和矩阵乘法。

项目结构：

1. c文件：main.c、 point\_wise\_product.c、matrix\_product.c；

h文件：point\_wise\_product.h、matrix\_product.h；

Makefile文件；

matrix.app可执行程序；

2. main.c调用point\_wise\_product.c和matrix\_product.c中定义的函数，分别实现矩阵的逐元素乘法和矩阵乘法，Makefile将所有的源文件编译、链接并生成可执行程序matrix.app。

题目要求：

1. 给出3个c文件的内容；

2. 给出Makefile文件的内容；

3. matrix.app程序实现如示例所示功能（不要求输入输出的格式或内容一致，只需要能够处理不同形状的矩阵即可），给出运行结果的截屏：

4. 给出程序识别不合法输入（例如两个矩阵形状不匹配）并提示报错信息的一个例子。

示例：

$ ./matrix.app

Enter '0' for Point-wise Product, '1' for Matrix Product, or 'q' to quit: 0

Enter number of rows and columns: 2 2

Enter elements of the first matrix:

3 -1

0 2

Enter elements of the second matrix:

3 2

1 1

Point-wise product result is:

9 -2

0 2

Enter '0' for Point-wise Product, '1' for Matrix Product, or 'q' to quit: 1

Enter number of rows and columns for the first matrix: 2 3

Enter elements of the first matrix:

2 1 3

4 0 2

Enter number of rows and columns for the second matrix: 3 2

Enter elements of the second matrix:

1 2

0 1

3 3

Matrix product result is:

11 14

10 14

Enter '0' for Point-wise Product, '1' for Matrix Product, or 'q' to quit: q

program ended.

$

1. point\_wise\_product.h:

#ifndef POINT\_WISE\_PRODUCT\_H

#define POINT\_WISE\_PRODUCT\_H

int point\_wise\_product(int\* A, int\* B, int\* C, int rows, int cols);

#endif

2. point\_wise\_product.c:

#include "point\_wise\_product.h"

int point\_wise\_product(int\* A, int\* B, int\* C, int rows, int cols) {

    for(int i = 0; i < rows; i++) {

        for(int j = 0; j < cols; j++) {

            C[i \* cols + j] = A[i \* cols + j] \* B[i \* cols + j];

        }

    }

    return 0;

}

3. matrix\_product.h:

#ifndef MATRIX\_PRODUCT\_H

#define MATRIX\_PRODUCT\_H

int matrix\_product(int\* A, int\* B, int\* C, int rows1, int cols1, int cols2);

#endif

4. matrix\_product.c:

#include "matrix\_product.h"

int matrix\_product(int\* A, int\* B, int\* C, int rows1, int cols1, int cols2) {

    for(int i = 0; i < rows1; i++) {

        for(int j = 0; j < cols2; j++) {

            C[i \* cols2 + j] = 0;

            for(int k = 0; k < cols1; k++) {

                C[i \* cols2 + j] += A[i \* cols1 + k] \* B[k \* cols2 + j];

            }

        }

    }

    return 0;

}

5. main.c

#include <stdio.h>

#include <stdlib.h>

#include "point\_wise\_product.h"

#include "matrix\_product.h"

void print\_matrix(int\* matrix, int rows, int cols) {

    for(int i = 0; i < rows; i++) {

        for(int j = 0; j < cols; j++) {

            printf("%d ", matrix[i \* cols + j]);

        }

        printf("\n");

    }

}

int main() {

    char choice;

    while(1) {

        printf("Enter '0' for Point-wise Product, '1' for Matrix Product, or 'q' to quit: ");

        scanf(" %c", &choice);

        if(choice == 'q') {

            printf("program ended.\n");

            break;

        }

        if(choice == '0') {

            int rows, cols;

            printf("Enter number of rows and columns: ");

            if(scanf("%d %d", &rows, &cols) != 2) {

                printf("Invalid input!\n");

                continue;

            }

            int \*A = (int\*)malloc(rows \* cols \* sizeof(int));

            int \*B = (int\*)malloc(rows \* cols \* sizeof(int));

            int \*C = (int\*)malloc(rows \* cols \* sizeof(int));

            printf("Enter elements of the first matrix:\n");

            for(int i = 0; i < rows \* cols; i++)

                scanf("%d", &A[i]);

            printf("Enter elements of the second matrix:\n");

            for(int i = 0; i < rows \* cols; i++)

                scanf("%d", &B[i]);

            point\_wise\_product(A, B, C, rows, cols);

            printf("Point-wise product result is:\n");

            print\_matrix(C, rows, cols);

            free(A);

            free(B);

            free(C);

        }

        else if(choice == '1') {

            int rows1, cols1, rows2, cols2;

            printf("Enter number of rows and columns for the first matrix: ");

            scanf("%d %d", &rows1, &cols1);

            printf("Enter number of rows and columns for the second matrix: ");

            scanf("%d %d", &rows2, &cols2);

            if(cols1 != rows2) {

                printf("Error: Matrix dimensions don't match for matrix multiplication!\n");

                continue;

            }

            int \*A = (int\*)malloc(rows1 \* cols1 \* sizeof(int));

            int \*B = (int\*)malloc(rows2 \* cols2 \* sizeof(int));

            int \*C = (int\*)malloc(rows1 \* cols2 \* sizeof(int));

            printf("Enter elements of the first matrix:\n");

            for(int i = 0; i < rows1 \* cols1; i++)

                scanf("%d", &A[i]);

            printf("Enter elements of the second matrix:\n");

            for(int i = 0; i < rows2 \* cols2; i++)

                scanf("%d", &B[i]);

            matrix\_product(A, B, C, rows1, cols1, cols2);

            printf("Matrix product result is:\n");

            print\_matrix(C, rows1, cols2);

            free(A);

            free(B);

            free(C);

        }

    }

    return 0;

}

6. Makefile:

CC = gcc

CFLAGS = -Wall -I.

SRCS = main.c point\_wise\_product.c matrix\_product.c

OBJS = $(SRCS:.c=.o)

TARGET = matrix.app

$(TARGET): $(OBJS)

    $(CC) $(OBJS) -o $(TARGET)

%.o: %.c

    $(CC) $(CFLAGS) -c $< -o $@

clean:

    rm -f $(OBJS) $(TARGET)

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成