一、基础命令(15分)

1.创建新用户 newuser，将用户的登陆 shell 设置为 bash，设置用户密码为 linux24。

代码：

创建新用户：

```shell
sudo useradd -m -s /bin/bash newuser
```

设置用户密码：

```shell
echo "newuser:linux24" | sudo chpasswd
```

结果：

```
root@iZ2ze0olgnf08nfb73yrokZ:~# sudo useradd -m -s /bin/bash newuser
root@iZ2ze0olgnf08nfb73yrokZ:~# echo "newuser:linux24" | sudo chpasswd
root@iZ2ze0olgnf08nfb73yrokZ:~# id newuser
uid=1004(newuser) gid=1007(newuser) groups=1007(newuser)
root@iZ2ze0olgnf08nfb73yrokZ:~#
```

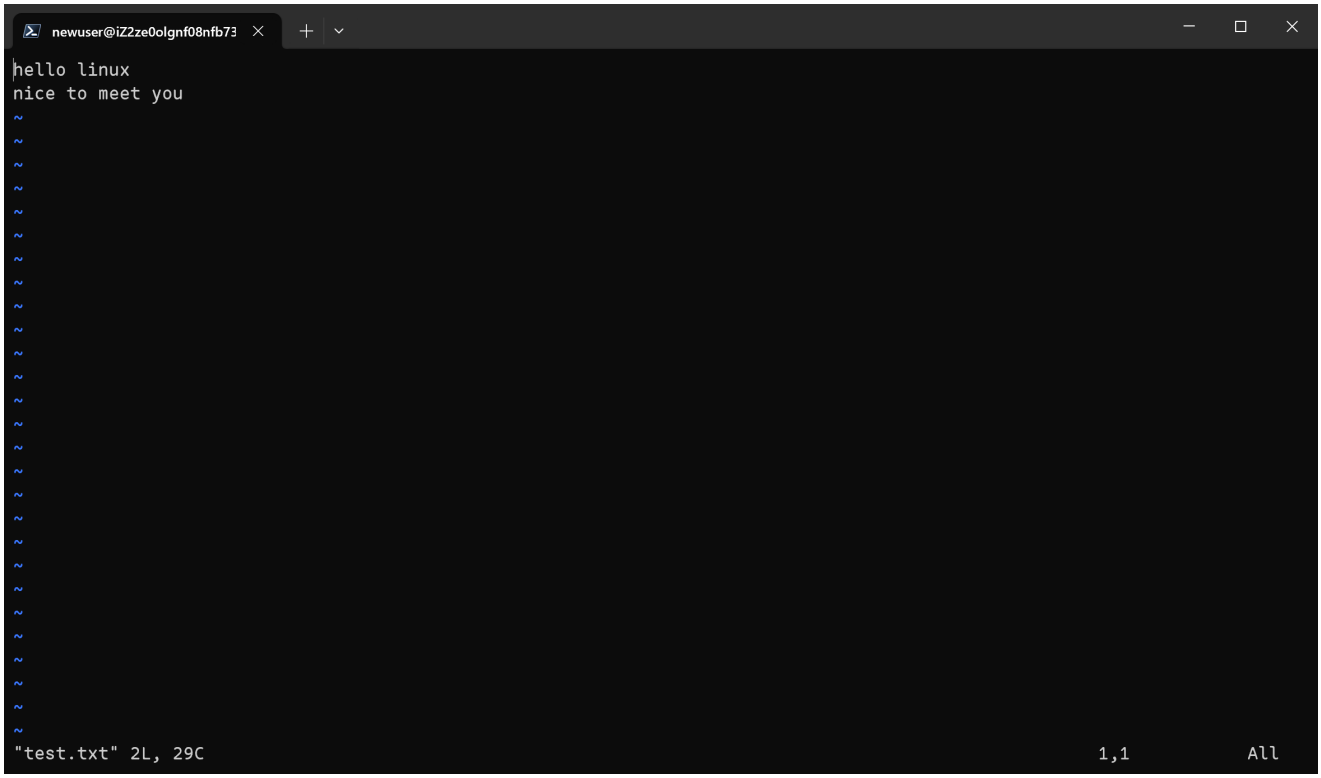
2.用一行命令创建文件 test.txt 并写入‘hello linux’，然后用输出重定向追加写入‘nice to meet you’

代码：

```shell
echo 'hello linux' > test.txt && echo 'nice to meet you' >> test.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ echo 'hello linux' > test.txt && echo 'nice to meet you' >> test.txt
newuser@iZ2ze0olgnf08nfb73yrokZ:~$
```



3.查看test.txt文件的大小，定位到用户主目录，查看主目录大小，查看主目录下所有次一级文件夹的大小并按降序排序（以人类可读的方式）

查看 test.txt 文件的大小：

代码：

```shell
ls -lh test.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ls -lh test.txt
-rw-rw-r-- 1 newuser newuser 29 Jun  2 11:44 test.txt
```

得到 test.txt 的文件大小为 29B

定位到用户主目录

代码：

```shell
cd ~
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~/temp$ cd ~
newuser@iZ2ze0olgnf08nfb73yrokZ:~$
```

查看主目录大小：

代码：

```shell
du -sh ~
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ du -sh ~
1.6G    /home/newuser
```

查看主目录下所有次一级文件夹的大小并按降序排序：

代码：

```shell
du -sh ~/*/ | sort -hr
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ du -sh ~/*/ | sort -hr
1.1G    /home/newuser/temp/
513M    /home/newuser/temp2/
28K     /home/newuser/c_project/
```

4.查看所有磁盘的使用情况

代码：

```shell
df -h
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            915M     0  915M   0% /dev
tmpfs           190M  712K  189M   1% /run
/dev/vda3        40G  6.4G   31G  18% /
tmpfs           946M     0  946M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           946M     0  946M   0% /sys/fs/cgroup
/dev/vda2       189M  6.1M  183M   4% /boot/efi
tmpfs           190M     0  190M   0% /run/user/0
```

5.查看系统信息

代码：

```shell
uname -a
```

结果：

6.查看系统的 cpu 信息，运行内存信息

查看 cpu 信息：

代码：

```shell
lscpu
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ lscpu
Architecture:                       x86_64
CPU op-mode(s):                     32-bit, 64-bit
Byte Order:                         Little Endian
Address sizes:                      46 bits physical, 48 bits virtual
CPU(s):                             2
On-line CPU(s) list:                0,1
Thread(s) per core:                 2
Core(s) per socket:                 1
Socket(s):                          1
NUMA node(s):                       1
Vendor ID:                          GenuineIntel
CPU family:                         6
Model:                              85
Model name:                         Intel(R) Xeon(R) Platinum
Stepping:                           4
CPU MHz:                            2500.002
BogoMIPS:                           5000.00
Hypervisor vendor:                  KVM
Virtualization type:                full
L1d cache:                          32 KiB
L1i cache:                          32 KiB
L2 cache:                           1 MiB
L3 cache:                           33 MiB
NUMA node0 CPU(s):                  0,1
Vulnerability Gather data sampling: Unknown: Dependent on hypervisor status
Vulnerability Itlb multihit:        KVM: Vulnerable
Vulnerability L1tf:                 Mitigation; PTE Inversion
Vulnerability Mds:                  Vulnerable: Clear CPU buffers attempted, no microcode; SMT Host state unknown
Vulnerability Meltdown:             Mitigation; PTI
Vulnerability Mmio stale data:      Vulnerable: Clear CPU buffers attempted, no microcode; SMT Host state unknown
Vulnerability Retbleed:             Vulnerable
Vulnerability Spec store bypass:    Vulnerable
Vulnerability Spectre v1:           Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:           Mitigation; Retpolines, STIBP disabled, RSB filling, PBRSB-eIBRS Not affected
Vulnerability Srbds:                Not affected
Vulnerability Tsx async abort:      Vulnerable: Clear CPU buffers attempted, no microcode; SMT Host state unknown
Flags:                              fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx
                                    fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopolog
                                    y nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x
                                    2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm a
                                    bm 3dnowprefetch invpcid_single pti fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms
                                     invpcid rtm mpx avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw
                                    avx512vl xsaveopt xsavec xgetbv1 xsaves arat
```

查看运行内存信息：

代码：

```shell
free -h
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           1.8Gi       136Mi       238Mi       2.0Mi       1.5Gi       1.5Gi
Swap:             0B          0B          0B
```

二、字符串操作（30分）

1.创建 content.txt 文件，其内容为：

Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date;

Sometime too hot the eye of heaven shines,
And often is his gold complexion dimm'd;
And every fair from fair sometime declines,
By chance or nature's changing course untrimm'd;
But thy eternal summer shall not fade,
Nor lose possession of that fair thou ow'st;
Nor shall death brag thou wander'st in his shade,
When in eternal lines to time thou grow'st:
So long as men can breathe or eyes can see,
So long lives this and this gives life to thee

(1) 打印含有单词 summer 的行，修改 content.txt，将所有的 summer 替换为 winter

打印含有单词 `summer` 的行：

代码：

```shell
grep 'summer' content.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ grep 'summer' content.txt
Shall I compare thee to a summer's day?
And summer's lease hath all too short a date;
But thy eternal summer shall not fade,
```

将所有的 `summer` 替换为 `winter`：

代码：

```shell
sed -i 's/summer/winter/g' content.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ sed -i 's/summer/winter/g' content.txt
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ grep 'winter' content.txt
Shall I compare thee to a winter's day?
And winter's lease hath all too short a date;
But thy eternal winter shall not fade,
```

(2) 将每个单词的首字母改为大写并写入 upper.txt 中

代码：

```shell
awk '{ for (i=1;i≤NF;i++) $i=toupper(substr($i,1,1))
tolower(substr($i,2)); print }' content.txt > upper.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ awk '{ for (i=1;i<=NF;i++) $i=toupper(substr($i,1,1)) tolower(substr($i,2)); print }'
 content.txt > upper.txt
```

```
Shall I Compare Thee To A Winter's Day?
Thou Art More Lovely And More Temperate:
Rough Winds Do Shake The Darling Buds Of May,
And Winter's Lease Hath All Too Short A Date;
Sometime Too Hot The Eye Of Heaven Shines,
And Often Is His Gold Complexion Dimm'd;
And Every Fair From Fair Sometime Declines,
By Chance Or Nature's Changing Course Untrimm'd;
But Thy Eternal Winter Shall Not Fade,
Nor Lose Possession Of That Fair Thou Ow'st;
Nor Shall Death Brag Thou Wander'st In His Shade,
When In Eternal Lines To Time Thou Grow'st:
So Long As Men Can Breathe Or Eyes Can See,
So Long Lives This And This Gives Life To Thee

~
~
~
~
~
~
~
~
~
~
~
~
~
"upper.txt" 15L, 636C                                          1,1           All
```

(3) 去除 upper.txt 中每行末的标点符号，然后统计每个单词（带单引号的视作一个单词）出现的频率并按降序排序

去除 upper.txt 中每行末的标点符号：

代码：

```shell
sed 's/[[:punct:]]*$//' upper.txt
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ newuser@iZ2ze0olgnf08nfb73yrokZ:~$ sed 's/[[:punct:]]*$//' upper.txt
Shall I Compare Thee To A Winter's Day
Thou Art More Lovely And More Temperate
Rough Winds Do Shake The Darling Buds Of May
And Winter's Lease Hath All Too Short A Date
Sometime Too Hot The Eye Of Heaven Shines
And Often Is His Gold Complexion Dimm'd
And Every Fair From Fair Sometime Declines
By Chance Or Nature's Changing Course Untrimm'd
But Thy Eternal Winter Shall Not Fade
Nor Lose Possession Of That Fair Thou Ow'st
Nor Shall Death Brag Thou Wander'st In His Shade
When In Eternal Lines To Time Thou Grow'st
So Long As Men Can Breathe Or Eyes Can See
So Long Lives This And This Gives Life To Thee
```

统计每个单词（带单引号的视作一个单词）出现的频率并按降序排序：

代码：

```shell
tr ' ' '\n' < upper.txt | grep -v '^$' | sort | uniq -c | sort -nr
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ tr ' ' '\n' < upper.txt | grep -v '^$' | sort | uniq -c | sort -nr
      5 And
      4 Thou
      3 To
      3 Shall
      3 Of
      3 Fair
      2 Winter’s
      2 Too
      2 This
      2 Thee
      2 The
      2 Sometime
      2 So
      2 Or
      2 Nor
      2 More
      2 Long
      2 In
      2 His
      2 Eternal
      2 Can
      2 A
      1 Winter
      1 Winds
      1 When
      1 Wander’st
      1 Untrimm’d;
      1 Time
      1 Thy
      1 That
      1 Temperate:
      1 Short
      1 Shines,
      1 Shake
      1 Shade,
      1 See,
      1 Rough
      1 Possession
      1 Ow’st;
      1 Often
      1 Not
      1 Nature’s
      1 Men
      1 May,
      1 Lovely
      1 Lose
      1 Lives
      1 Lines
      1 Life
      1 Lease
      1 Is
      1 I
      1 Hot
      1 Heaven
      1 Hath
      1 Grow’st:
      1 Gold
      1 Gives
      1 From
      1 Fade,
      1 Eyes
      1 Eye
      1 Every
      1 Do
      1 Dimm’d;
      1 Declines,
      1 Death
      1 Day?
      1 Date;
      1 Darling
      1 Course
      1 Complexion
      1 Compare
```

```
      1 Changing
      1 Chance
      1 By
      1 But
      1 Buds
      1 Breathe
      1 Brag
      1 As
      1 Art
      1 All
newuser@iZ2ze0olgnf08nfb73yrokZ:~$
```

2.假设有一个字符串 source_str="Hello,World,How,Are,You"，请按照以下要求完成相应的操作：

(1) 输出字符串的长度，并输出第[5，10)个字符

代码：

```shell
echo ${#source_str} && echo ${source_str:4:5}
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ echo ${#source_str} && echo ${source_str:4:5}
23
o,Wor
```

(2) 将逗号（,）替换为空格

代码：

```shell
source_str=${source_str//,/ }
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ source_str=${source_str//,/ }
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ echo $source_str
Hello World How Are You
```

(3) 提取第三个字段（单词）

代码：

```shell
echo $source_str | cut -d' ' -f3
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ echo $source_str | cut -d' ' -f3
How
```

(4) 提取第二个字段（单词）并转换为大写

代码：

```shell
echo $source_str | cut -d' ' -f2 | tr '[:lower:]' '[:upper:]'
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ echo $source_str | cut -d' ' -f2 | tr '[:lower:]' '[:upper:]'
WORLD
```

3.现有train.log文件，内容如下：

```
-----Epoch 951/999-----
Epoch 951 Train, Loss: 5.76,  MSE: 9.18 MAE: 5.55, Cost 19.4 sec
-----Epoch 952/999-----
Epoch 952 Train, Loss: 6.98, MSE: 11.68 MAE: 6.80, Cost 19.1 sec
-----Epoch 953/999-----
Epoch 953 Train, Loss: 4.81, MSE: 8.38 MAE: 4.62, Cost 19.0 sec
-----Epoch 954/999-----
Epoch 954 Train, Loss: 5.61,  MSE: 9.68 MAE: 5.42, Cost 19.0 sec
-----Epoch 955/999-----
Epoch 955 Train, Loss: 5.75, MSE: 9.16 MAE: 5.57, Cost 18.6 sec
Epoch 955 Val, MSE: 106.85 MAE: 65.82, Cost 7.5 sec
-----Epoch 956/999-----
Epoch 956 Train, Loss: 5.43,  MSE: 8.61 MAE: 5.23, Cost 19.5 sec
-----Epoch 957/999-----
Epoch 957 Train, Loss: 6.04,  MSE: 9.70 MAE: 5.85, Cost 18.0 sec
-----Epoch 958/999-----
Epoch 958 Train, Loss: 4.62, MSE: 7.11 MAE: 4.44, Cost 19.3 sec
-----Epoch 959/999-----
Epoch 959 Train, Loss: 5.73, MSE: 9.33 MAE: 5.55, Cost 17.9 sec
-----Epoch 960/999-----
Epoch 960 Train, Loss: 5.88,  MSE: 10.15 MAE: 5.67, Cost 19.2 sec
Epoch 960 Val, MSE: 108.59 MAE: 67.02, Cost 7.3 sec
-----Epoch 961/999-----
Epoch 961 Train, Loss: 5.30,  MSE: 8.41 MAE: 5.12, Cost 19.3 sec
-----Epoch 962/999-----
Epoch 962 Train, Loss: 5.90,  MSE: 9.97 MAE: 5.71, Cost 18.9 sec
-----Epoch 963/999-----
Epoch 963 Train, Loss: 6.07,  MSE: 10.30 MAE: 5.88, Cost 19.5 sec
```

```
-----Epoch 964/999-----
Epoch 964 Train, Loss: 7.08,  MSE: 13.13 MAE: 6.88, Cost 18.4 sec
-----Epoch 965/999-----
Epoch 965 Train, Loss: 5.59,  MSE: 8.58 MAE: 5.40, Cost 20.0 sec
```

请完成以下操作:

(1)统计文件的行数

代码:

```shell
wc -l train.log
```

结果:

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ wc -l train.log
33 train.log
```

(2)提取出所有带字符串 Train 的行

代码:

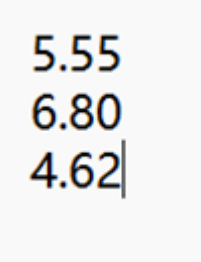
```shell
grep 'Train' train.log
```

结果:

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ grep 'Train' train.log
Epoch 951 Train, Loss: 5.76,  MSE: 9.18 MAE: 5.55, Cost 19.4 sec
Epoch 952 Train, Loss: 6.98, MSE: 11.68 MAE: 6.80, Cost 19.1 sec
Epoch 953 Train, Loss: 4.81, MSE: 8.38 MAE: 4.62, Cost 19.0 sec
Epoch 954 Train, Loss: 5.61,  MSE: 9.68 MAE: 5.42, Cost 19.0 sec
Epoch 955 Train, Loss: 5.75, MSE: 9.16 MAE: 5.57, Cost 18.6 sec
Epoch 956 Train, Loss: 5.43,  MSE: 8.61 MAE: 5.23, Cost 19.5 sec
Epoch 957 Train, Loss: 6.04,  MSE: 9.70 MAE: 5.85, Cost 18.0 sec
Epoch 958 Train, Loss: 4.62, MSE: 7.11 MAE: 4.44, Cost 19.3 sec
Epoch 959 Train, Loss: 5.73, MSE: 9.33 MAE: 5.55, Cost 17.9 sec
Epoch 960 Train, Loss: 5.88,  MSE: 10.15 MAE: 5.67, Cost 19.2 sec
Epoch 961 Train, Loss: 5.30,  MSE: 8.41 MAE: 5.12, Cost 19.3 sec
Epoch 962 Train, Loss: 5.90,  MSE: 9.97 MAE: 5.71, Cost 18.9 sec
Epoch 963 Train, Loss: 6.07,  MSE: 10.30 MAE: 5.88, Cost 19.5 sec
Epoch 964 Train, Loss: 7.08,  MSE: 13.13 MAE: 6.88, Cost 18.4 sec
Epoch 965 Train, Loss: 5.59,  MSE: 8.58 MAE: 5.40, Cost 20.0 sec
```

(3)提取出其中所有Train Epoch中的`MAE`值。

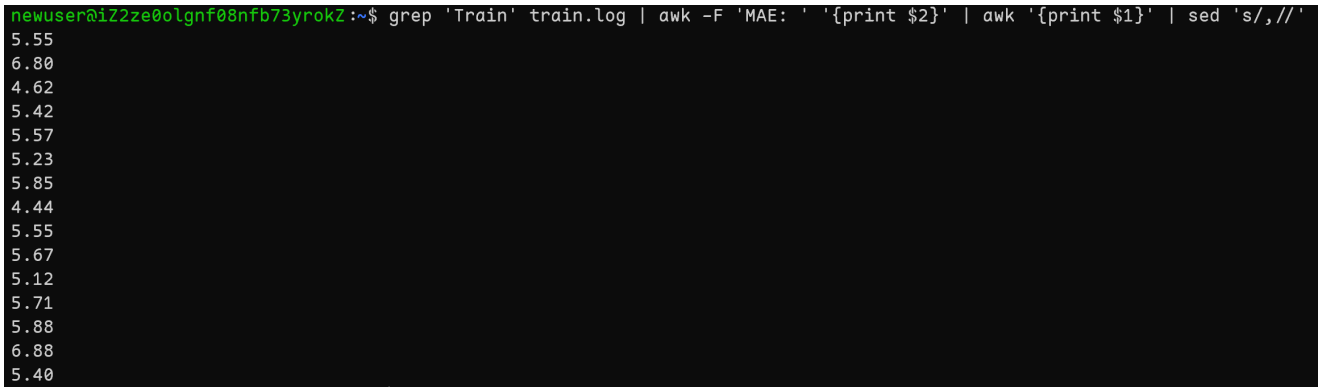
输出例如：



代码：

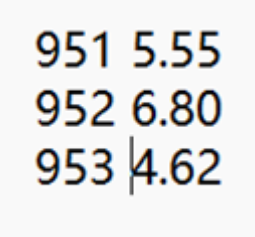
```shell
grep 'Train' train.log | awk -F 'MAE: ' '{print $2}' | awk '{print $1}' | sed 's/,//'
```

结果：



(4)提取出其中所有Train Epoch中的`Epoch`值以及对应的`MAE`值并输出到文件`mae.txt`，每行的内容为`Epoch MAE`。

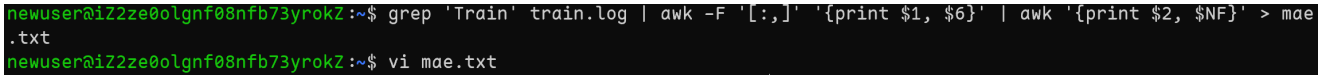
输出例如：

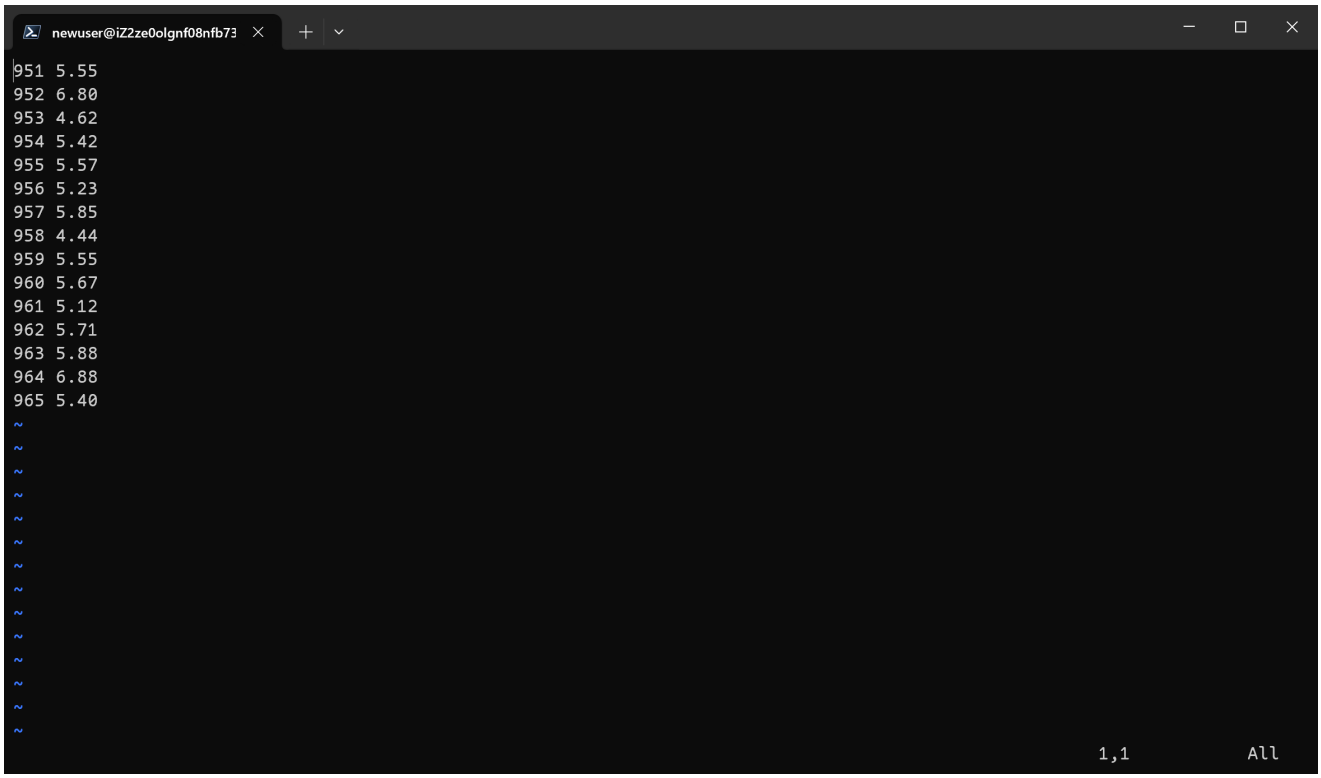


代码：

```shell
grep 'Train' train.log | awk -F '[:,]' '{print $1, $6}' | awk '{print $2, $NF}' > mae.txt
```

结果：





三、Shell 脚本（30分）

1.生成6位随机密码

给定字符串

str='ABCDRFGHIJKLMNOPQRSTUVWXTZabcdrfghijklmnopqrstuvwxtz0123456789',

请编写 shell 脚本 rand_passwd.sh，在 str 中随机选择6位创建一个密码（可重复选取），并将密码返回给用户

脚本代码如下：

Shell

```shell
#!/bin/bash
str='ABCDRFGHIJKLMNOPQRSTUVWXTZabcdrfghijklmnopqrstuvwxtz0123456789'
# 获取字符串长度
len=${#str}
# 初始化密码变量
password=""
# 生成6位随机密码
for i in {1..6}
do
  # 生成一个随机索引
  rand_index=$((RANDOM % len))
  # 从字符串中选择一个字符并添加到密码中
  password="${password}${str:$rand_index:1}"
```

```
done
# 返回生成的密码
echo "Generated password: $password"
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ vi rand_passwd.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ chmod +x rand_passwd.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./rand_passwd.sh
Generated password: rotR7j
newuser@iZ2ze0olgnf08nfb73yrokZ:~$
```

2.温度单位转换

编写脚本 temp_convert.sh，脚本接受一个整数参数，如10C 或10F，C 表示摄氏温度，F 表示华氏温度，实现摄氏度与华氏度的相互转化，并将结果保留到整数，以与输入相同的形式返回给用户。

脚本代码如下：

Shell

```
#!/bin/bash

# 检查是否提供了一个参数
if [ -z "$1" ]; then
  echo "Usage: $0 <temperature>"
  echo "Example: $0 10C or $0 50F"
  exit 1
fi

# 提取温度值和单位
temp=${1%[CF]}
unit=${1: -1}

# 检查温度值是否为整数
if ! [[ $temp =~ ^-?[0-9]+$ ]]; then
  echo "Error: Temperature must be an integer."
  exit 1
fi

# 转换温度
if [ "$unit" == "C" ]; then
  # 摄氏度转换为华氏度

```

```
    result=$(( (temp * 9 / 5) + 32 ))
    echo "${result}F"
  elif [ "$unit" == "F" ]; then
    # 华氏度转换为摄氏度
    result=$(( (temp - 32) * 5 / 9 ))
    echo "${result}C"
  else
    echo "Error: Unknown temperature unit. Use 'C' for Celsius
or 'F' for Fahrenheit."
    exit 1
  fi
```

结果：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ vi temp_convert.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ chmod +x temp_convert.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./temp_convert.sh 10C
50F
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./temp_convert.sh 90F
32C
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./temp_convert.sh 9.0F
Error: Temperature must be an integer.
newuser@iZ2ze0olgnf08nfb73yrokZ:~$
```

3.编写shell脚本print_diamond.sh，脚本接受一个用户输入n（n>=2），用`*`打印边长为n、竖直放置的菱形。

脚本输出如：

```
yq@215F4:~/exam$ bash print_diamond.sh 5
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

脚本代码如下：

Shell

```
#!/bin/bash

# 检查输入参数是否存在并且为一个整数
if [ -z "$1" ] || ! [[ "$1" =~ ^[0-9]+$ ]]; then
  echo "Usage: $0 <integer>"
```

```
  echo "Please enter an integer greater than or equal to 2."
  exit 1
fi

# 检查输入参数是否大于等于2
n=$1
if [ "$n" -lt 2 ]; then
  echo "Please enter an integer greater than or equal to 2."
  exit 1
fi

# 打印上半部分
for i in $(seq 1 $n); do
  # 打印前面的空格
  for j in $(seq $i $n); do
    echo -n " "
  done
  # 打印*
  for j in $(seq 1 $((2*i-1))); do
    echo -n "*"
  done
  echo
done

# 打印下半部分
for i in $(seq $((n-1)) -1 1); do
  # 打印前面的空格
  for j in $(seq $n $i); do
    echo -n " "
  done
  # 打印*
  for j in $(seq 1 $((2*i-1))); do
    echo -n "*"
  done
  echo
done

```

结果:

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ vi print_diamond.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ chmod +x print_diamond.sh
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./print_diamond.sh 5
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ ./print_diamond.sh 7
      *
     ***
    *****
   *******
  *********
 ***********
*************
 ***********
  *********
   *******
    *****
     ***
      *
```

四、Makefile（25分）

完成一个 C 语言项目，项目功能是实现二进制数与十进制数的相互转化，项目包括三个 C 文件 `main.c`，`binary2decimal.c` 和 `decimal2binary.c`。其中 `main.c` 调用 `binary2decimal.c` 中的 `binary2decimal` 函数和 `decimal2binary.c` 中的 `decimal2binary` 函数。

代码要求：

1. main.c包含main函数，并引用头文件binary2decimal.h和binary2decimal.h
2. main.c接受两个命令行输入，第一个输入为字符b或d，表示输入数据是二进制(b)或十进制(d)，第二个输入为需要转换进制的32位无符号整数
3. binary2decimal函数实现二进制数转十进制数，decimal2binary实现十进制数转二进制数
4. 编写一个Makefile文件，来编译和链接所有的文件，生成最终的可执行文件app
5. Makefile 应该包括一个 clean 命令，用于删除所有编译生成的文件

题目要求

1. 给出main.c和两个子函数文件的内容
2. 给出Makefile文件内容
3. 实现如下功能：
   $./app d 57
   $Decimal 57 converted to binary: 111001
   $./app b 10110
   $Binary 10110 converted to decimal: 22

代码：

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "binary2decimal.h"
#include "decimal2binary.h"

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <b|d> <number>\n", argv[0]);
        return 1;
    }

    char mode = argv[1][0];
    char *input = argv[2];

    if (mode == 'b') {
        // Binary to Decimal
        unsigned int decimal = binary2decimal(input);
        printf("Binary %s converted to decimal: %u\n", input,
decimal);
    } else if (mode == 'd') {
        // Decimal to Binary
        unsigned int decimal = atoi(input);
        char binary[33];
        decimal2binary(decimal, binary);
        printf("Decimal %u converted to binary: %s\n",
decimal, binary);
    } else {
        printf("Invalid mode. Use 'b' for binary or 'd' for
decimal.\n");

```

```
32          return 1;
        }

        return 0;
    }
```

binary2decimal.c

```c
#include "binary2decimal.h"

unsigned int binary2decimal(const char *binary) {
    unsigned int decimal = 0;
    while (*binary) {
        decimal = (decimal << 1) + (*binary - '0');
        binary++;
    }
    return decimal;
}
```

binary2decimal.h

```c
#ifndef BINARY2DECIMAL_H
#define BINARY2DECIMAL_H

unsigned int binary2decimal(const char *binary);

#endif
```

decimal2binary.c

```c
#include "decimal2binary.h"

void decimal2binary(unsigned int decimal, char *binary) {
    int index = 31;
    binary[32] = '\0';

    while (index ≥ 0) {
        binary[index] = (decimal & 1) ? '1' : '0';
        decimal >>= 1;
        index--;
    }
```

```c

    // Remove leading zeros
    int i = 0;
    while (binary[i] == '0' && i < 31) {
        i++;
    }
    memmove(binary, binary + i, 33 - i);
}
```

decimal2binary.h

```c
#ifndef DECIMAL2BINARY_H
#define DECIMAL2BINARY_H

void decimal2binary(unsigned int decimal, char *binary);

#endif
```

Makefile

```Makefile
CC = gcc
CFLAGS = -Wall -Wextra -std=c99
OBJ = main.o binary2decimal.o decimal2binary.o
DEPS = binary2decimal.h decimal2binary.h

app: $(OBJ)
	$(CC) -o $@ $^

%.o: %.c $(DEPS)
	$(CC) $(CFLAGS) -c -o $@ $<

clean:
	rm -f $(OBJ) app
```

结果:

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ mkdir c_project
newuser@iZ2ze0olgnf08nfb73yrokZ:~$ cd c_project
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi main.c
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi binary2decimal.c
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi decimal2binary.c
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi binary2decimal.h
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi decimal2binary.h
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ vi Makefile
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ make
gcc -Wall -Wextra -std=c99 -c -o main.o main.c
gcc -Wall -Wextra -std=c99 -c -o binary2decimal.o binary2decimal.c
gcc -Wall -Wextra -std=c99 -c -o decimal2binary.o decimal2binary.c
decimal2binary.c: In function ‘decimal2binary’:
decimal2binary.c:18:5: warning: implicit declaration of function ‘memmove’ [-Wimplicit-function-declaration]
   18 |     memmove(binary, binary + i, 33 - i);
      |     ^~~~~~~
decimal2binary.c:18:5: warning: incompatible implicit declaration of built-in function ‘memmove’
decimal2binary.c:2:1: note: include ‘<string.h>’ or provide a declaration of ‘memmove’
    1 | #include "decimal2binary.h"
  +++ |+#include <string.h>
    2 |
gcc -o app main.o binary2decimal.o decimal2binary.o
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ ./app d 57
Decimal 57 converted to binary: 111001
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ ./app b 10110
Binary 10110 converted to decimal: 22
```

删除所有编译生成的文件：

```
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ ls
app  binary2decimal.c  binary2decimal.h  binary2decimal.o  decimal2binary.c  decimal2binary.h  decimal2binary.o  main.c  main.o  Makefile
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$ make clean
rm -f main.o binary2decimal.o decimal2binary.o app
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$  ls
binary2decimal.c  binary2decimal.h  decimal2binary.c  decimal2binary.h  main.c  Makefile
newuser@iZ2ze0olgnf08nfb73yrokZ:~/c_project$
```