

**Linux开发环境及应用**



**Shell脚本程序设计**

**姓 名 鄭毓恒**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2020211302**

**学 号 2020211262**

**任课教师 张雪松**

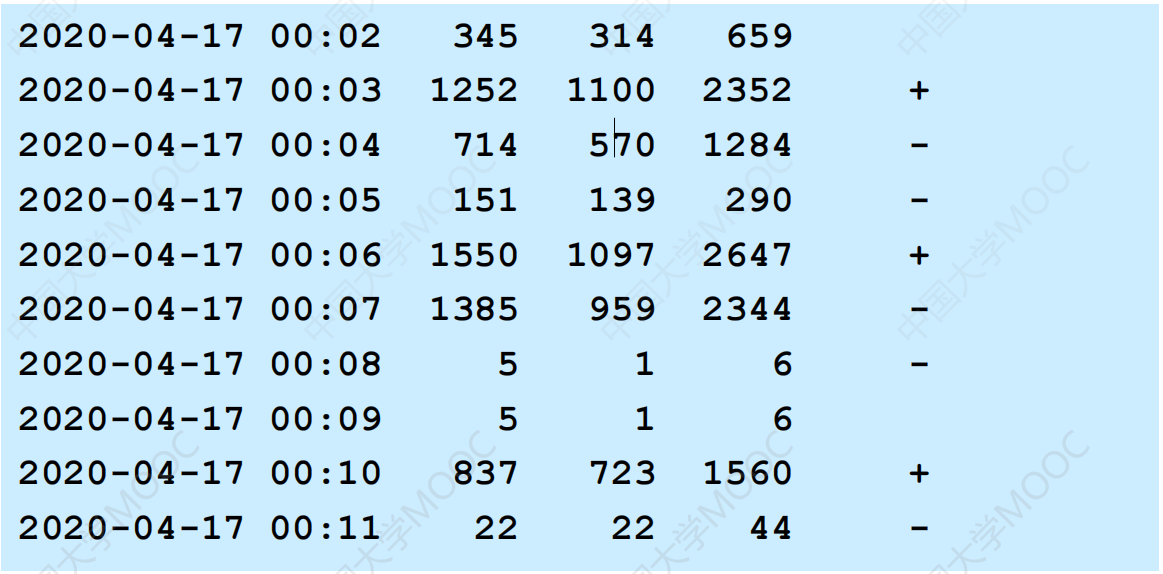
**2023年 5 月**

**第一题 生成TCP活动状况报告**

**作业要求**

netstat --statistics命令可以列出tcp等协议的统计信息。编写shell脚本程序，每隔1分钟生成1行信息：当前时间；这一分钟内TCP发送了多少报文；接收了多少报文；收发报文总数；行尾给出符号+或-或空格（+表示这分钟收发报文数比上分钟多10包以上，差别在10包或以内用空格，否则用符号-）。

运行示例：



**程序设计思路**

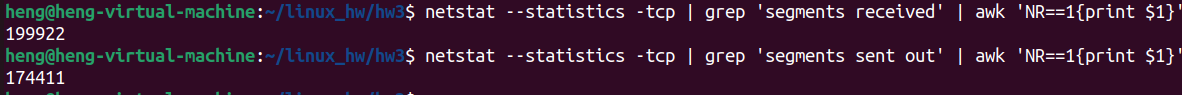
本次实验需要按规定格式输出年、月、日、时和分，可以使用date命令，按所需格式获取时间。



输入指令netstat –statistics -tcp，可得到以下数据。其中，segments received和segment sent out为系统启动至今收发的TCP报文总数。



使用以下两条指令，可以获取这两项数据。只要每隔一分钟获取一次这两项数据，计算与上一分钟的数据的差，可以得到这一分钟的收发TCP报文总数，也可以决定在行尾输出+或-或空格。



程序代码中所使用的变量信息如下。

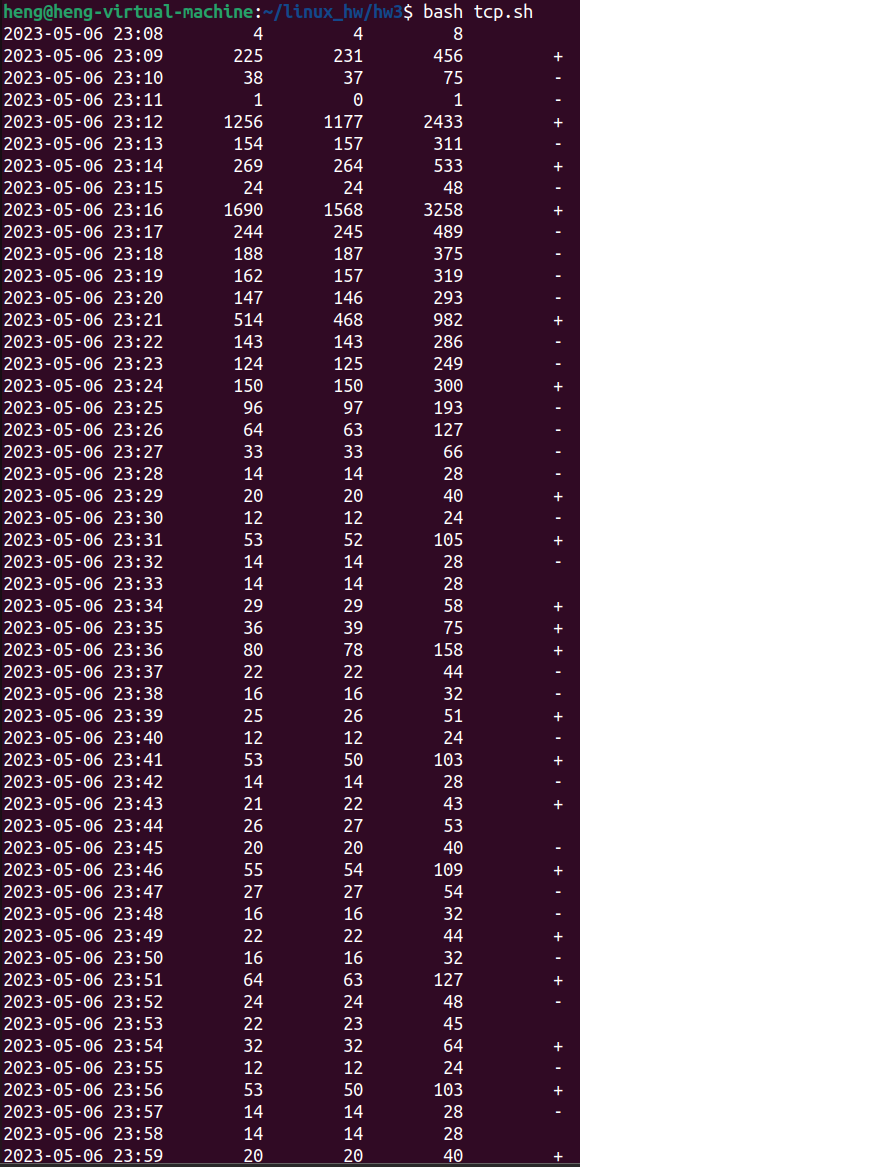
|  |  |
| --- | --- |
| 变量名 | 作用 |
| nowRecv | 当前获取的segment received数据 |
| nowSend | 当前获取的segment sent out 数据 |
| nowTotal | 当前这一分钟的TCP收发报文总数 |
| count | 循环次数 |
| lastRecv | 上一分钟获取的segment received数据 |
| lastSend | 上一分钟获取的segment sent out数据 |
| lastTotal | 上一分钟的TCP收发报文总数 |
| recv | 这一分钟收到的TCP报文 |
| send | 这一分钟发出的TCP报文 |
| totalDiff | 上一分钟和现在的TCP收发报文总数的差 |
| sign | 应该输出的符号，+或-或空格 |

程序代码主要由一个循环组成。循环在程序开始后一分钟才开始，在此之前，先将变量进行初始化。给lastRecv和lastSend程序开始时的TCP数据，以便循环中使用。

每次循环开始，再次获取TCP收发数据，计算这一分钟的TCP收发报文总数，计算与上一分钟的收发总数的差。然后，使用if，判断应该输出的符号。由变量count可得知是第几次循环，第一次循环不需判断符号，输出空格。最后，按格式输出信息，再将本次TCP数据保存。

**运行结果**

程序运行结果如下，符合题目要求，输出正确。



**实验心得**

在本次实验中，初次体验了shell脚本的设计与编写，令我更加掌握shell脚本语言。本次实验中遇到的问题主要因为对shell脚本语言的不熟悉，例如在C语言中的代码规范化写法，如sign = “+”，在shell中会因为多余的空格而无法执行。

**源代码**

|  |
| --- |
| nowRecv=`netstat --statistics -tcp | grep 'segments received' | awk 'NR==1{print $1}'`  nowSend=`netstat --statistics -tcp | grep 'segments sent out' | awk 'NR==1{print $1}'`  nowTotal=0  count=0  lastRecv=$nowRecv  lastSend=$nowSend  lastTotal=0  while true  do  sleep 60  time=`date +"%Y-%m-%d %H:%M"`  nowRecv=`netstat --statistics -tcp | grep 'segments received' | awk 'NR==1{print $1}'`  nowSend=`netstat --statistics -tcp | grep 'segments sent out' | awk 'NR==1{print $1}'`  count=`expr $count + 1`  recv=`expr $nowRecv - $lastRecv`  send=`expr $nowSend - $lastSend`  nowTotal=`expr $recv + $send`    totalDiff=`expr $nowTotal - $lastTotal`  if [ $count -eq 1 ]  then sign=" "  elif [ $totalDiff -gt 10 ]  then sign="+"  elif [ $totalDiff -lt -10 ]  then sign="-"  else  sign=" "  fi    printf "%-s %-s%10s%10s%10s%10s\n" $time $recv $send $nowTotal $sign    lastRecv=$nowRecv  lastSend=$nowSend  lastTotal=$nowTotal  done |

**第二题 下载bing图库中图片**

**作业要求**

访问https://bing.ioliu.cn/?p=36可以看到bing图库第36页的内容，这个Web页有多个图片小样，将鼠标放到某个小样上，如右上角，可见中文说明信息“野花草甸上的一只欧亚雕鸮，德国莱茵兰-普法尔茨”和日期信息2019-08-03，点击一下，此图片就可以下载。

编写脚本程序bing.sh，将图库中照片全部下载下来存放到本地bing目录，上面URL中p=36可以换成p=126可访问126号页面，每页有12个图，每个图的日期，中文说明信息和下载地址及文件名html文件中可提取。要求下载后的文件命名为“日期 说明.jpg”例如：

2019-08-03 野花草甸上的一只欧亚雕鸮，德国莱茵兰-普法尔茨.jpg

* 不重复下载已下载的图片

检查图片是否已下载，如果已下载，则不再下载

支持批量任务在被中断后再次启动程序可以从中断处继续

* 考虑并发

启动多个进程（nproc=5）同时下载，可以加快任务完成的速度

* 考虑下载文件出现故障的情况

如果一个图片有5MB，接收1.5MB后网络断开，则残存一个不完整的图片文件。

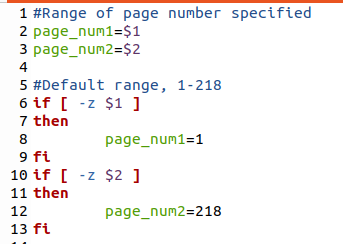
避免这种现象发生的一种方法是wget下载时使用一个临时文件名。

判断wget是否成功，若成功则将文件改名为正式名称；若失败，删除临时文件。

临时文件名的选取要考虑的并发问题，至少不可以程序中写死一个文件名导致两进程因使用相同的临时文件名而失败。

**程序设计思路**

程序支持用户输入两个参数，表示下载的起始页数。如果不输入，则默认下载全部页的图片。



本次实验需要图片的日期、名称和下载链接。查看网页HTML文件，三个元素可利用grep和sed指令得到。同时，可通过更改URL的数字达到翻页，这样就可以下载所有页面的所有图片。

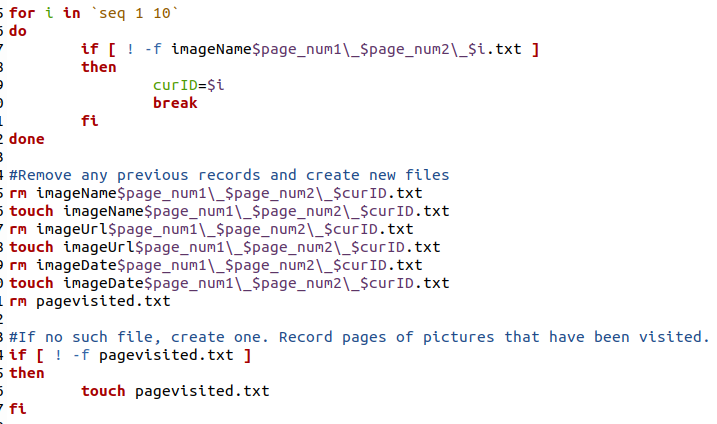
|  |  |
| --- | --- |
| 日期 | grep -o '20[0-9][0-9]-[0-9][0-9]-[0-9][0-9]' |
| 名称 | grep -Eo '<h3>(.|\n)\*?</h3>' | sed 's/<[^>]\*>/ /g' | sed 's/([^)]\*)/ /g' | sed 's/分享/\n/g' | sed 's/[0-9a-zA-Z]\*/ /g' | sed 's/- -/ /g' | sed 's/\ //g' |
| 链接 | grep -Eo '<img[^>]\*src="https://[^"]\*[^"]\*"[^>]\*>' | grep -Eo 'src="[^"]\*"' | sed 's/src="//g' | sed 's/"//g' |

程序首先将要下载的页数范围的每个图片的日期、名称和链接都分别存储在三个不同的文件里。在确定所有需要下载的图片后，利用这些数据得到目的文件名，再进行图片的下载。

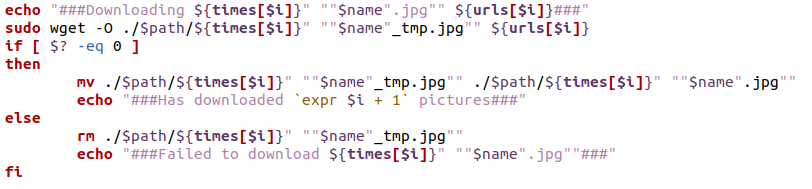
由于题目要求的图片文件命名格式是固定的，同时全都存储在一个文件夹下。因此，只需要在下载图片前检查目的文件名是否已经存在，则可以避免重复下载图片。

为了实现进程并发下载图片，首先使用一个文件pagevisited.txt，为所有进程共有，表示当前已经访问过的页面，避免不同进程对同一页面的图片进行重复下载。每个进程都有自己独特的命名方式，用于命名各自要下载的图片的数据（日期、名称和链接）的存储文件，避免访问同一文件，造成重复下载。

这种命名方式下，首先文件名将包含进程要下载的起始页数，将不同起始页数的进程分开。然后，对于相同起始页数的进程，为其分配一个独有的ID，为1-10之间的一个数字。因此，最多支持十个进程同时下载相同的起始页数。

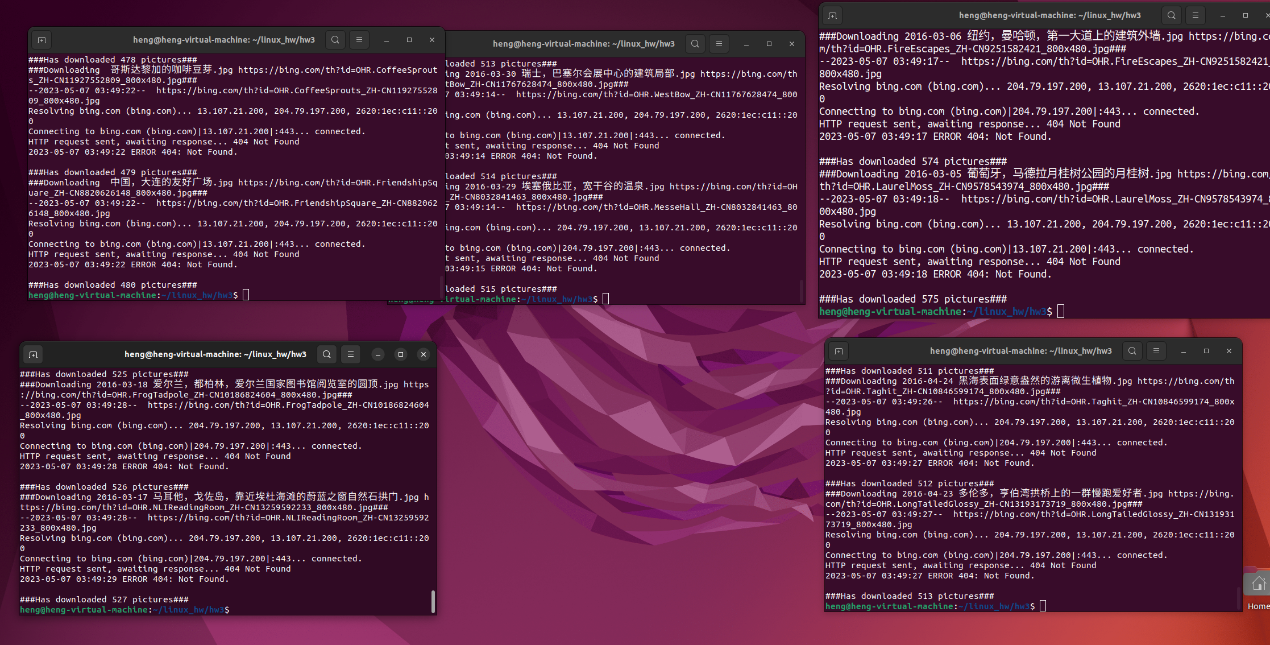


图片下载后，会先存储在一个临时文件中。使用$?判断wget指令的返回值后，如果下载成功，才会更名为正式文件，否则删除。



**运行结果**

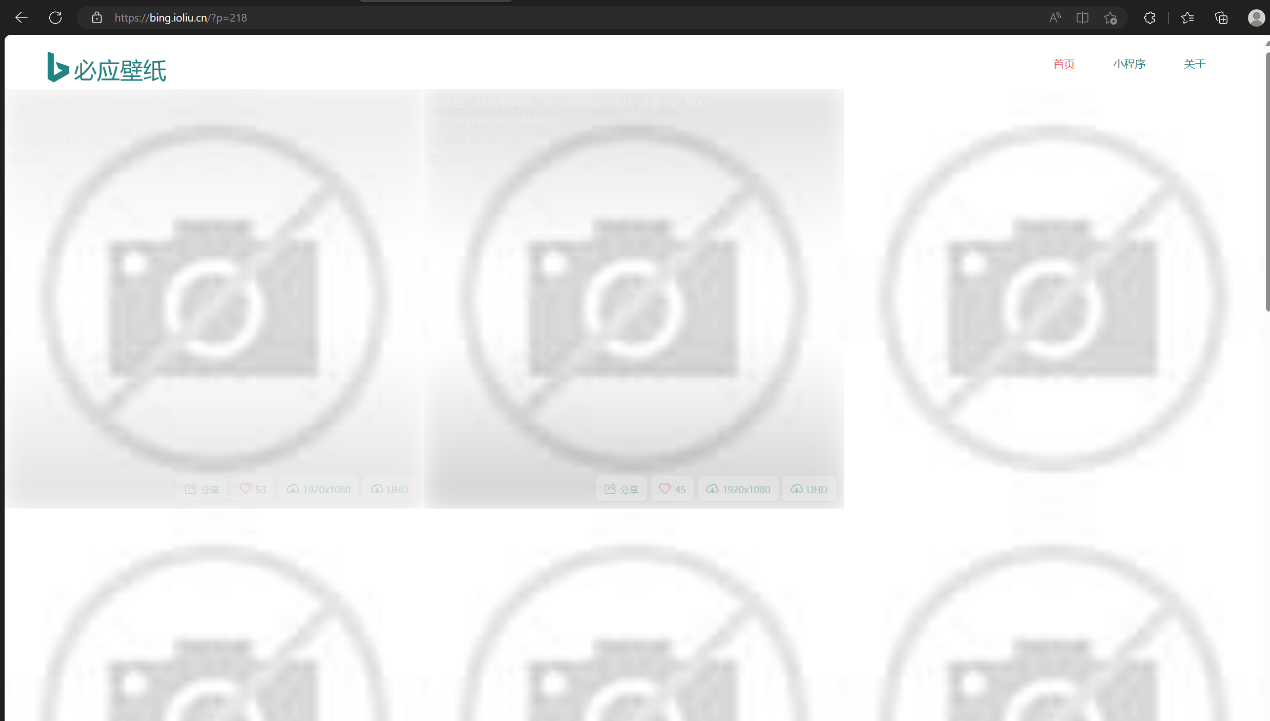
启动5个进程，下载全部图片，结果如下。根据输出的下载图片数量，与网站图片数量一致，没有重复下载，结果正确。



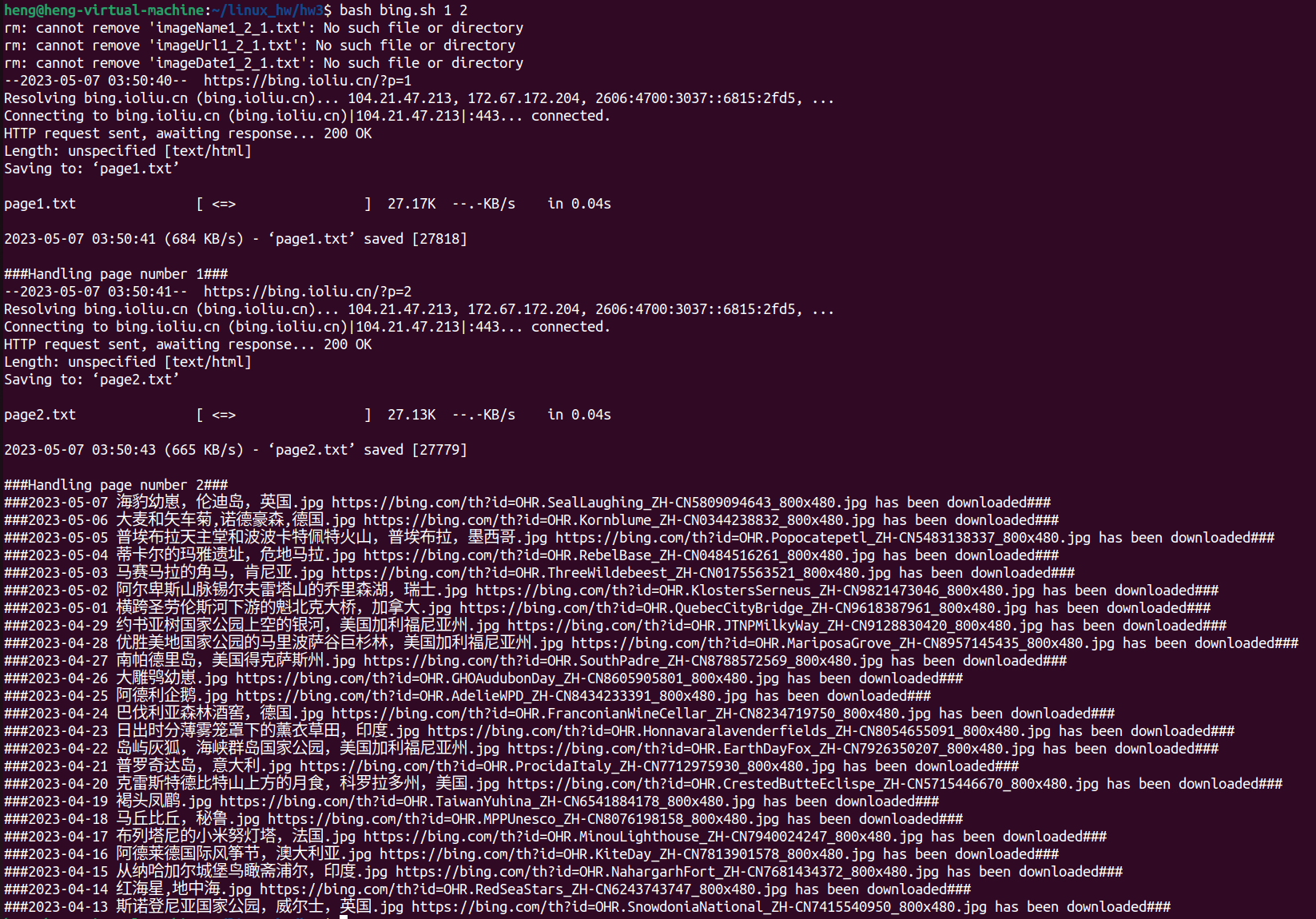
打开文件夹，可以看到所有图片。其中有些图片无法正确显示，这是网站自身的原因。



网站中有些图片无法显示，也无法下载。



然后，再次下载第1-2页的图片，可以看到程序提示这两页的图片已下载，不会重复下载。



**实验心得**

在本次实验中，设计和编写了更加复杂的shell程序，实现了更高级的功能，更加掌握shell脚本语言，也令我了解到shell脚本功能的广泛。实验难度较大，主要在于如何实现多进程并发。在编写程序过程中，曾想过用进程编号来命名文件以避免冲突，但通过ps指令发现，同一程序的所有进程使用了同一个名字，在shell代码中难以区分，所以采用自己分配ID的方式实现。但是，这样的问题在于可以并发的进程数量有限，也存在被分配了同一id的风险。需要继续学习进程方面的知识，找到解决方法，完成更好的多进程同步。

**源代码**

|  |
| --- |
| #Range of page number specified  page\_num1=$1  page\_num2=$2  #Default range, 1-218  if [ -z $1 ]  then  page\_num1=1  fi  if [ -z $2 ]  then  page\_num2=218  fi  for i in `seq 1 10`  do  if [ ! -f imageName$page\_num1\\_$page\_num2\\_$i.txt ]  then  curID=$i  break  fi  done  #Remove any previous records and create new files  rm imageName$page\_num1\\_$page\_num2\\_$curID.txt  touch imageName$page\_num1\\_$page\_num2\\_$curID.txt  rm imageUrl$page\_num1\\_$page\_num2\\_$curID.txt  touch imageUrl$page\_num1\\_$page\_num2\\_$curID.txt  rm imageDate$page\_num1\\_$page\_num2\\_$curID.txt  touch imageDate$page\_num1\\_$page\_num2\\_$curID.txt  rm pagevisited.txt  #If no such file, create one. Record pages of pictures that have been visited.  if [ ! -f pagevisited.txt ]  then  touch pagevisited.txt  fi  #Create directory of pictures  path="bing"  if [ ! -d "./$path" ]  then  mkdir "./$path"  fi  #Get picture info by pages  for i in `seq $page\_num1 $page\_num2`  do  #Check if the page has been visited  flag=0  for visited in `cat pagevisited.txt`  do  [ $i = $visited ] && flag=1  done  #The page has not been visited  [ $flag -eq 0 ] && {  #Mark the page "visited"  echo $i >> pagevisited.txt  wget -O page$i.txt -c https://bing.ioliu.cn/?p=$i  echo "###Handling page number $i###"  #Getting links of pictures  cat page$i.txt | grep -Eo '<img[^>]\*src="https://[^"]\*[^"]\*"[^>]\*>' | grep -Eo 'src="[^"]\*"' | sed 's/src="//g' | sed 's/"//g' >> imageUrl$page\_num1\\_$page\_num2\\_$curID.txt  #Getting names of pictures  cat page$i.txt | grep -Eo '<h3>(.|\n)\*?</h3>' | sed 's/<[^>]\*>/ /g' | sed 's/([^)]\*)/ /g' | sed 's/分享/\n/g' | sed 's/[0-9a-zA-Z]\*/ /g' | sed 's/- -/ /g' | sed 's/\ //g' >> imageName$page\_num1\\_$page\_num2\\_$curID.txt  #Getting dates of pictures  cat page$i.txt | grep -o '20[0-9][0-9]-[0-9][0-9]-[0-9][0-9]' >> imageDate$page\_num1\\_$page\_num2\\_$curID.txt  rm page$i.txt  }  done  #Getting links of pictures from file  a=0  for line in `cat imageUrl$page\_num1\\_$page\_num2\\_$curID.txt | uniq`  do  urls[$a]=$line  a=`expr $a + 1`  done  #Getting dates of pictures from file  b=0  for line in `cat imageDate$page\_num1\\_$page\_num2\\_$curID.txt | uniq`  do  times[$b]=$line  b=`expr $b + 1`  done  #Downloading pictures  i=0  for name in `cat imageName$page\_num1\\_$page\_num2\\_$curID.txt`  do  if [ ! -f ./$path/${times[$i]}" ""$name".jpg"" ]  then  echo "###Downloading ${times[$i]}" ""$name".jpg"" ${urls[$i]}###"  sudo wget -O ./$path/${times[$i]}" ""$name"\_tmp.jpg"" ${urls[$i]}  if [ $? -eq 0 ]  then  mv ./$path/${times[$i]}" ""$name"\_tmp.jpg"" ./$path/${times[$i]}" ""$name".jpg""  echo "###Has downloaded `expr $i + 1` pictures###"  else  rm ./$path/${times[$i]}" ""$name"\_tmp.jpg""  echo "###Failed to download ${times[$i]}" ""$name".jpg""###"  fi  else  echo "###${times[$i]}" ""$name".jpg"" ${urls[$i]} has been downloaded###"  fi  i=`expr $i + 1`  done  rm imageName$page\_num1\\_$page\_num2\\_$curID.txt  rm imageUrl$page\_num1\\_$page\_num2\\_$curID.txt  rm imageDate$page\_num1\\_$page\_num2\\_$curID.txt |