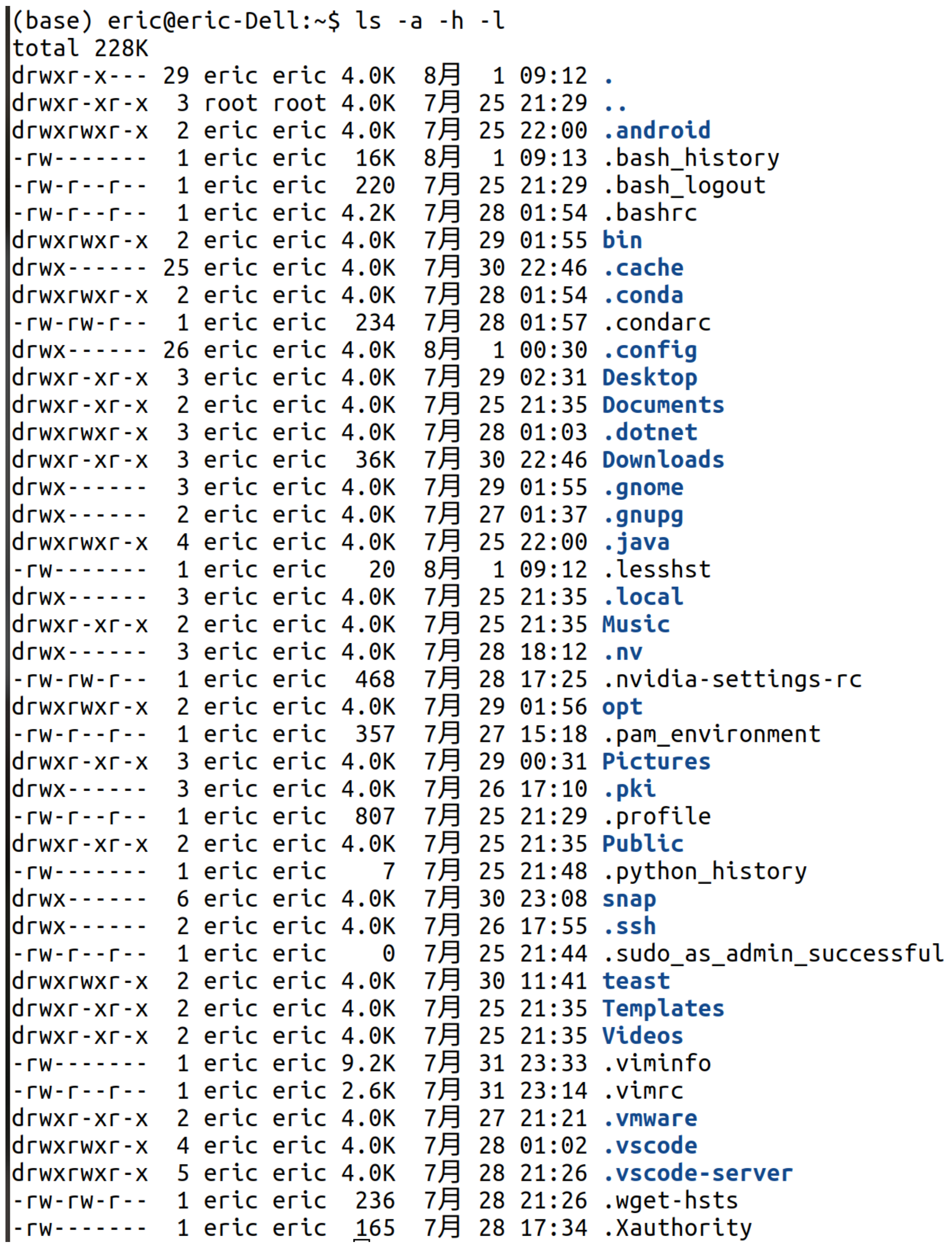
Homework 2

1. Read man ls and write an ls command that lists files in the following manner
    - Include all files, including hidden files
    - sizes are listed in human readable format (e.g, 454M instead of 454279954)
    - Files are ordered by recency
    - Output is colorized

   A sample output would look like this:

   ```
   -rw-r--r--   1 user group 1.1M Jan 14 09:53 baz
   drwxr-xr-x   5 user group  160 Jan 14 09:53 .
   -rw-r--r--   1 user group  514 Jan 14 06:42 bar
   -rw-r--r--   1 user group 106M Jan 13 12:12 foo
   drwx------+ 47 user group 1.5K Jan 12 18:08 ..
   ```

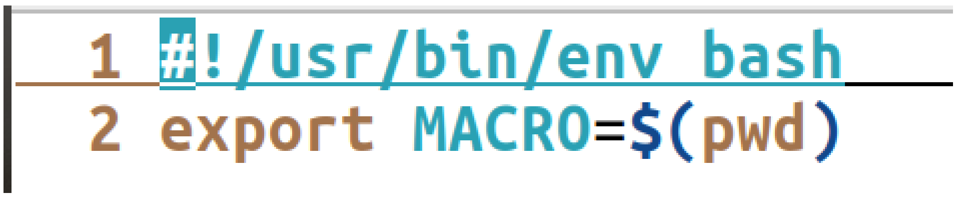
   ```
   ls -a -h -l
   ```



```
(base) eric@eric-Dell:~$ ls -a -h -l
total 228K
drwxr-x--- 29 eric eric 4.0K  8月  1 09:12 .
drwxr-xr-x  3 root root 4.0K  7月 25 21:29 ..
drwxrwxr-x  2 eric eric 4.0K  7月 25 22:00 .android
-rw-------  1 eric eric  16K  8月  1 09:13 .bash_history
-rw-r--r--  1 eric eric  220  7月 25 21:29 .bash_logout
-rw-r--r--  1 eric eric 4.2K  7月 28 01:54 .bashrc
drwxrwxr-x  2 eric eric 4.0K  7月 29 01:55 bin
drwx------ 25 eric eric 4.0K  7月 30 22:46 .cache
drwxrwxr-x  2 eric eric 4.0K  7月 28 01:54 .conda
-rw-rw-r--  1 eric eric  234  7月 28 01:57 .condarc
drwx------ 26 eric eric 4.0K  8月  1 00:30 .config
drwxr-xr-x  3 eric eric 4.0K  7月 29 02:31 Desktop
drwxr-xr-x  2 eric eric 4.0K  7月 25 21:35 Documents
drwxrwxr-x  3 eric eric 4.0K  7月 28 01:03 .dotnet
drwxr-xr-x  3 eric eric  36K  7月 30 22:46 Downloads
drwx------  3 eric eric 4.0K  7月 29 01:55 .gnome
drwx------  2 eric eric 4.0K  7月 27 01:37 .gnupg
drwxrwxr-x  4 eric eric 4.0K  7月 25 22:00 .java
-rw-------  1 eric eric   20  8月  1 09:12 .lesshst
drwx------  3 eric eric 4.0K  7月 25 21:35 .local
drwxr-xr-x  2 eric eric 4.0K  7月 25 21:35 Music
drwx------  3 eric eric 4.0K  7月 28 18:12 .nv
-rw-rw-r--  1 eric eric  468  7月 28 17:25 .nvidia-settings-rc
drwxrwxr-x  2 eric eric 4.0K  7月 29 01:56 opt
-rw-r--r--  1 eric eric  357  7月 27 15:18 .pam_environment
drwxr-xr-x  3 eric eric 4.0K  7月 29 00:31 Pictures
drwx------  3 eric eric 4.0K  7月 26 17:10 .pki
-rw-r--r--  1 eric eric  807  7月 25 21:29 .profile
drwxr-xr-x  2 eric eric 4.0K  7月 25 21:35 Public
-rw-------  1 eric eric    7  7月 25 21:48 .python_history
drwx------  6 eric eric 4.0K  7月 30 23:08 snap
drwx------  2 eric eric 4.0K  7月 26 17:55 .ssh
-rw-r--r--  1 eric eric    0  7月 25 21:44 .sudo_as_admin_successful
drwxrwxr-x  2 eric eric 4.0K  7月 30 11:41 teast
drwxr-xr-x  2 eric eric 4.0K  7月 25 21:35 Templates
drwxr-xr-x  2 eric eric 4.0K  7月 25 21:35 Videos
-rw-------  1 eric eric 9.2K  7月 31 23:33 .viminfo
-rw-r--r--  1 eric eric 2.6K  7月 31 23:14 .vimrc
drwxr-xr-x  2 eric eric 4.0K  7月 27 21:21 .vmware
drwxrwxr-x  4 eric eric 4.0K  7月 28 01:02 .vscode
drwxrwxr-x  5 eric eric 4.0K  7月 28 21:26 .vscode-server
-rw-rw-r--  1 eric eric  236  7月 28 21:26 .wget-hsts
-rw-------  1 eric eric  165  7月 28 17:34 .Xauthority
```

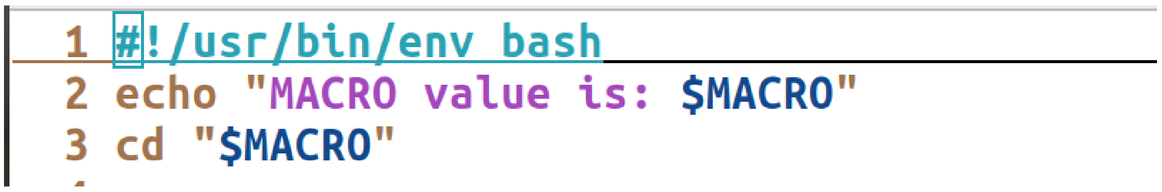
2. Write bash functions macro and polo that do the following. Whenever you exercise macro the current working directory should be saved in some manner, then when you execute polo, no matter what directory you are in, polo should cd you back to the directory where you execute macro. For ease of debugging, you can write the code in a file named marco.sh and (re)load the definitions to your shell by executing source marco.sh.

   macro.sh



```
#!/usr/bin/env bash
export MACRO=$(pwd)
```

   polo.sh



```
#!/usr/bin/env bash
echo "MACRO value is: $MACRO"
cd "$MACRO"
```

3. Say you have a command that fails rarely. In order to debug it, you need to capture its output but it can be time consuming to get a failure run. Write a bash script that runs the following script until it fails and captures its standard output and error streams to files and prints everything at the end. Bonus points if you can also report how many runs it took for the script to fail.

```
#!/usr/bin/env bash

n=$(( RANDOM % 100 ))

if [[ n -eq 42 ]]; then
   echo "Something went wrong"
   >&2 echo "The error was using magic numbers"
   exit 1
fi

echo "Everything went according to plan"
```

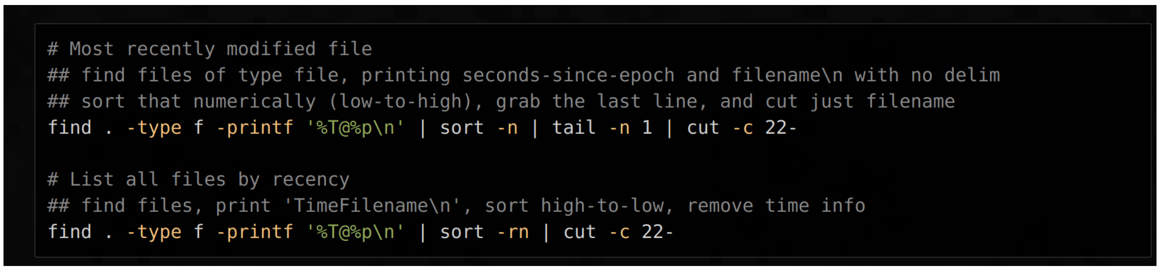
```
#!/usr/bin/env bash
cnt=1
while true; do
    source code.sh > error.log 2>&1
    if [[ $? -eq 1 ]]
    then
        break
    else
        echo "the shell runs good"
    fi
    echo "run $cnt times"
    cnt=$[$cnt+1]
done
```

4. As we covered in the lecture find's -exec can be very powerful for performing operations over the files we are searching for. However, what if we want to do something with all the files, like creating a zip file? As you have seen so far commands will take input from both arguments and STDIN. When piping commands, we are connecting STDOUT to STDIN, but some commands like tar take inputs from arguments. To bridge this disconnect there's the xargs command which will execute a command using STDIN as arguments. For

example, ls | xargs rm will delete the files in the current directory. Your task is to write a command that recursively finds all HTML files in the folder and makes a zip with them. Note that your command should work even if the files have spaces.(hint: check -d flag for xargs)

```
find . -name '*.html' | xargs -d '\n' tar czf archieve.zip
```

5. Write a command or script to recursively find the most recently modified file in a directory. More generally, can you list all files by recency?



```
# Most recently modified file
## find files of type file, printing seconds-since-epoch and filename\n with no delim
## sort that numerically (low-to-high), grab the last line, and cut just filename
find . -type f -printf '%T@%p\n' | sort -n | tail -n 1 | cut -c 22-

# List all files by recency
## find files, print 'TimeFilename\n', sort high-to-low, remove time info
find . -type f -printf '%T@%p\n' | sort -rn | cut -c 22-
```