Local sort

It uses the system locale to determine the sorting order of letters. My guess is that with your locale, it ignores whitespace.

```
$ cat foo.txt
v 1006
v10 1
v 1011
$ LC_ALL=C sort foo.txt
v 1006
v 1011
v10 1
$ LC_ALL=en_US.utf8 sort foo.txt
v 1006
v10 1
v 1011
```

Git

High-level commands (porcelain)
We separate the porcelain commands into the main commands and some ancillary user utilities.

Main porcelain commands
git-add(1)
Add file contents to the index.

git-am(1)
Apply a series of patches from a mailbox.

git-archive(1)
Create an archive of files from a named tree.

git-bisect(1)
Use binary search to find the commit that introduced a bug.

git-branch(1)
List, create, or delete branches.

git-bundle(1)
Move objects and refs by archive.

git-checkout(1)
Switch branches or restore working tree files.

git-cherry-pick(1)
Apply the changes introduced by some existing commits.

git-citool(1)
Graphical alternative to git-commit.

git-clean(1)
Remove untracked files from the working tree.

git-clone(1)
Clone a repository into a new directory.

git-commit(1)
Record changes to the repository.

git-describe(1)
Describe a commit using the most recent tag reachable from it.

git-diff(1)
Show changes between commits, commit and working tree, etc.

git-fetch(1)
Download objects and refs from another repository.

git-format-patch(1)
Prepare patches for e-mail submission.

git-gc(1)
Cleanup unnecessary files and optimize the local repository.

git-grep(1)
Print lines matching a pattern.

git-gui(1)
A portable graphical interface to Git.

git-init(1)
Create an empty Git repository or reinitialize an existing one.

git-log(1)
Show commit logs.

git-merge(1)
Join two or more development histories together.

git-mv(1)
Move or rename a file, a directory, or a symlink.

git-notes(1)
Add or inspect object notes.

git-pull(1)
Fetch from and integrate with another repository or a local branch.

git-push(1)
Update remote refs along with associated objects.

git-rebase(1)
Reapply commits on top of another base tip.

git-reset(1)
Reset current HEAD to the specified state.

git-revert(1)
Revert some existing commits.

git-rm(1)
Remove files from the working tree and from the index.

git-shortlog(1)
Summarize git log output.

git-show(1)
Show various types of objects.

git-stash(1)
Stash the changes in a dirty working directory away.

git-status(1)
Show the working tree status.

git-submodule(1)
Initialize, update or inspect submodules.

git-tag(1)
Create, list, delete or verify a tag object signed with GPG.

git-worktree(1)
Manage multiple working trees.

gitk(1)
The Git repository browser.

Ancillary Commands
Manipulators:

git-config(1)
Get and set repository or global options.

git-fast-export(1)
Git data exporter.

git-fast-import(1)
Backend for fast Git data importers.

git-filter-branch(1)
Rewrite branches.

git-mergetool(1)
Run merge conflict resolution tools to resolve merge conflicts.

git-pack-refs(1)
Pack heads and tags for efficient repository access.

git-prune(1)

Prune all unreachable objects from the object database.

git-reflog(1)
Manage reflog information.

git-remote(1)
Manage set of tracked repositories.

git-repack(1)
Pack unpacked objects in a repository.

git-replace(1)
Create, list, delete refs to replace objects.

Interrogators:

git-annotate(1)
Annotate file lines with commit information.

git-blame(1)
Show what revision and author last modified each line of a file.

git-cherry(1)
Find commits yet to be applied to upstream.

git-count-objects(1)
Count unpacked number of objects and their disk consumption.

git-difftool(1)
Show changes using common diff tools.

git-fsck(1)
Verifies the connectivity and validity of the objects in the database.

git-get-tar-commit-id(1)
Extract commit ID from an archive created using git-archive.

git-help(1)
Display help information about Git.

git-instaweb(1)
Instantly browse your working repository in gitweb.

git-merge-tree(1)
Show three-way merge without touching index.

git-rerere(1)
Reuse recorded resolution of conflicted merges.

git-rev-parse(1)
Pick out and massage parameters.

git-show-branch(1)
Show branches and their commits.

git-verify-commit(1)
Check the GPG signature of commits.

git-verify-tag(1)
Check the GPG signature of tags.

git-whatchanged(1)
Show logs with difference each commit introduces.

gitweb(1)
Git web interface (web frontend to Git repositories).

Interacting with Others
These commands are to interact with foreign SCM and with other people via patch over e-mail.

git-archimport(1)
Import an Arch repository into Git.

git-cvsexportcommit(1)
Export a single commit to a CVS checkout.

git-cvsimport(1)
Salvage your data out of another SCM people love to hate.

git-cvsserver(1)
A CVS server emulator for Git.

git-imap-send(1)
Send a collection of patches from stdin to an IMAP folder.

git-p4(1)
Import from and submit to Perforce repositories.

git-quiltimport(1)
Applies a quilt patchset onto the current branch.

git-request-pull(1)
Generates a summary of pending changes.

git-send-email(1)
Send a collection of patches as emails.

git-svn(1)
Bidirectional operation between a Subversion repository and Git.

---

answer1.txt

1. man -k <keyword> for search of keyword in description part of man page of the commands. man -K <keyword> for search globally.

2. Use which cp and which wc. which command is used for searching of executables. cp locates at /usr/bin/cp, and wc locates at /usr/bin/wc.

3. Use command "echo $PATH" to find locations of executable programs, which is /usr/lib64/qt-3.3/bin, /u/eng/class/classzfu/perl5/bin, /usr/lib64/ccache, /usr/local/bin, /usr/bin, /usr/X11R6/bin, /u/eng/class/classzfu/bin. Then use "find <locations above> -name ? -executable", and X, w, [are found. Use whatis X, whatis w, and whatis [.
w (1)   - Show who is logged on and what they are doing.
[(1)   - bash built-in commands, see bash(1)
but nothing is shown for X (nothing show in man X, also), which is weird.

4. Use command "readlink /usr/bin/emacs", get "/etc/alternatives/emacs".

5. Use "man chmod".
g+s: permit group to save
o-x: deny the permission for others to execute

6. find -mtime -20
-mtime is the time of modification, which will be rounded down. -20 means less than 20 days after rounding down. Toooooo many files, so i cannot display the output.

7. find -type d -mtime -20
-type d specifies to search for directories.

output after searching in home directory:
.
./perl5
./Desktop
./Desktop/$RECYCLE.BIN
./Desktop/$RECYCLE.BIN/$R1C9I3J
./Desktop/$RECYCLE.BIN/$ROUUEOD
./Desktop/$RECYCLE.BIN/$RLY71XR
./Desktop/$RECYCLE.BIN/$ROVCRD6
./Desktop/$RECYCLE.BIN/$RFV6NNX
./Desktop/$RECYCLE.BIN/$RJ8GFZZ
./Desktop/$RECYCLE.BIN/$RJW15AH
./Desktop/$RECYCLE.BIN/$RRVOTVN
./Documents
./Downloads
./Downloads/$RECYCLE.BIN
./.cache/abrt
./.ccache
./.ccache/tmp
./.ccache/4
./.ccache/d
./.ccache/e
./.ccache/6
./.ccache/2
./.ccache/1
./.ccache/9
./.ccache/c
./.ccache/5
./.ccache/7

```
./.ccache/a
./week1
./hw1
./.emacs.d/auto-save-list
```

8. command: "which find", get "/usr/bin/find". command: "find /usr/bin -type l" There are so many lines, so the last command can be "find /usr/bin -type l | nl" to list the index of lines. The total number of symbolic links is 294 in /usr/bin.

9. command: cd /usr/bin, first to the specified directory. command: ls -t -l, list all regular files, directories and symbolic links in this directory in the order of modification time from new to old. Hence, the last file with - instead of d or l in the front is the oldest regular file, which is "libfreeblpriv3.so".

10. By reading the locale manuel, /usr/lib/locale/locale-archive is the usual default locale archive location. /usr/share/i18n/locales is the usual default path for locale definition.

11. command: emacs, then command: c-h a sort, which allows us to search for keyword in commands.
Output:

```
Buffer-menu-sort                M-x ... RET
   Sort Tabulated List entries by the column at point.
sort-columns                    M-x ... RET
   Sort lines in region alphabetically by a certain range of columns.
sort-fields                     M-x ... RET
   Sort lines in region lexicographically by the ARGth field of each
   line.
sort-lines                      M-x ... RET
   Sort lines in region alphabetically; argument means descending
   order.
sort-numeric-fields             M-x ... RET
   Sort lines in region numerically by the ARGth field of each line.
sort-pages                      M-x ... RET
   Sort pages in region alphabetically; argument means descending
   order.
sort-paragraphs                 M-x ... RET
   Sort paragraphs in region alphabetically; argument means descending
   order.
sort-regexp-fields              M-x ... RET
   Sort the text in the region region lexicographically.
tabulated-list-col-sort         M-x ... RET
   Sort Tabulated List entries by the column of the mouse click E.
tabulated-list-sort             M-x ... RET
   Sort Tabulated List entries by the column at point.
```

12. command: c-h c-h, to get what options that can be typed after c-h. command: c-h b, to find the key translations of key bindings. Then, command: c-x o, to move the cursor to the another window.

```
C-M-a           beginning-of-defun
C-M-b           backward-sexp
C-M-c           exit-recursive-edit
C-M-d           down-list
```

```
C-M-e           end-of-defun
C-M-f           forward-sexp
C-M-h           mark-defun
```

13. command: c-h k c-g
output:
C-g runs the command keyboard-quit, which is an interactive compiled Lisp function.

It is bound to C-g.

(keyboard-quit)

Signal a `quit' condition.
During execution of Lisp code, this character causes a quit directly.
At top-level, as an editor command, this simply beeps.

14. command: c-h f yank
description:
yank is an interactive compiled Lisp function.

It is bound to C-y, <S-insertchar>, <S-insert>, <menu-bar> <edit> <paste>.

(yank &optional ARG)

Reinsert ("paste") the last stretch of killed text.
More precisely, reinsert the most recent kill, which is the
stretch of killed text most recently killed OR yanked. Put point
at the end, and set mark at the beginning without activating it.
With just C-u as argument, put point at beginning, and mark at end.
With argument N, reinsert the Nth most recent kill.

When this command inserts text into the buffer, it honors the
`yank-handled-properties' and `yank-excluded-properties'
variables, and the `yank-handler' text property. See
`insert-for-yank-1' for details.

See also the command `yank-pop' (M-y).

15. the dired in emacs (c-x d) shows the "total used in directory 345912 available 11769808", ls -l does not shows that. And it seems that ls -l does not have the enough space to list the entries entirely.

---

lab2.log

1. webpage dictionary

sort /usr/share/dict/words > words
sort the words file and store the sorted file in the lab2 directionary

```
wget http://web.cs.ucla.edu/classes/spring17/cs35L/assign/assign2.html
```
get the course assignment webpage

```
tr -c 'A-Za-z' '[\n*]' < assign2.html > out1.html
```
-c get the complement set of 'A-Za-z', which is any char other than alphabetical letters. Replace the char with \n, which is newline character. * means that the number of newline char in set2 will matches that in set1.

```
tr -cs 'A-Za-z' '[\n*]' < assign2.html > out2.html
```
-s, squeeze-repeats replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character, so there with be no consecutive newlines. Hence, there is single empty line (the line with a single newline char) on the top of the file, since the original assign2.html starts with two non-alphabets, followed by a single word on each line.

```
tr -cs 'A-Za-z' '[\n*]' < assign2.html | sort > out3.html
```
sort the letters in alphabetical order

```
tr -cs 'A-Za-z' '[\n*]' < assign2.html | sort -u > out4.html
```
eliminate the repeated ones

```
tr -cs 'A-Za-z' '[\n*]' < assign2.html | sort -u | comm - words > out5.html
```
Compare the sorted letters in the webpage and the sorted dictionary and display them in three columns. Col1 is the word only in file1, col2 is the ones only in file2 col3 is the ones in both files.

```
tr -cs 'A-Za-z' '[\n*]' < assign2.html | sort -u | comm -23 - words > out6.html
```
Appear only words in file1, which is the webpage, but not in file2, words.

2. Hawaiian dictionary

```
wget http://mauimapp.com/moolelo/hwnwdseng.htm -O hawaiian.html
```
get the html file

```
grep -E '<td>.*</td>' hawaiian.html > 1.html
```
find lines with the <td>...</td> pattern in the webpage and store them into the 1.html file

```
sed '/<td><\/td>/d' 1.html > 2.html
```
get rid of lines with <td></td> with no words between the tags

```
sed 's/<td>\(.*\)<\/td>/\1/g' 2.html > 3.html
```
delete <td> </td> tags around the words and keep the words

```
sed -n 2~2p 3.html > 4.html
```
get every 2th line starting with line 2, -n will avoid the duplication of the original file 3.html

```
sed 's/<u>\(.\)<\/u>/\1/g' 4.html > 5.html
```
get the underlined single letter between <u> and </u>

```
tr , '\n' < 5.html > 6.html
```

treat , as the separation of two words

sed 's/^[\t]*//' 6.html > 7.html
delete tabs in front of each line

tr ' ' '\n' < 7.html | sed '/^$/d' > 8.html
' ' indicates separate words before and after the space, sed '/^$/d' is used to eliminate empty lines

sed "s/\`/'/g" 8.html > 9.html
replace ` (accent), with ' (apostrophe)

tr [:upper:] [:lower:] < 9.html > 10.html
replace every upper case letter with its corresponding lower case letter

sed "/[^pkmnwlhaeiou']/d" 10.html > 11.html
delete every word other than pkmnwlhaeiou'

sort -u 11.html > hwords
sort the Hawaiin words in order

3. buildworld

```
#!/bin/bash

grep -E '<td>.*</td>' |

sed '/<td><\/td>/d' |

sed 's/<td>\(.*\)<\/td>/\1/g' |

sed -n 2~2p |

sed 's/<u>\(.\)<\/u>/\1/g' |

tr , '\n' |

sed 's/^[ \t]*//' |

tr ' ' '\n' |

sed '/^$/d' |

sed "s/\`/'/g" |
```

```
tr [:upper:] [:lower:] |

sed "/[^pkmnwlhaeiou']/d" |

sort -u
```

4. check by using hwords

misspelled Eng with upper case cast to lower case:
```
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat assign2.html | tr -cs 'A-Za-z' '[\n*]' |
tr '[:upper:]' '[:lower:]' | sort -u | comm -23 - words | wc -w
38
```

store the misEng file:
```
cat assign2.html | tr -cs 'A-Za-z' '[\n*]' | tr '[:upper:]' '[:lower:]' | sort -u
| comm -23 - words > misEng
```

misspelled Eng without case casting:
```
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat assign2.html | tr -cs 'A-Za-z' '[\n*]' |
sort -u | comm -23 - words | wc -w
80
```

misspelled Hawaiian after case casting to lower:
```
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat assign2.html | tr -cs "\'pkmnwlhaeiou"
'[\n*]' | tr '[:upper:]' '[:lower:]' | sort -u | comm -23 - hwords | wc -w
199
```

store the misHaw file:
```
cat assign2.html | tr -cs "\'pkmnwlhaeiou" '[\n*]' | tr '[:upper:]' '[:lower:]' |
sort -u | comm -23 - hwords > misHaw
```

misspelled Eng but true for Hawaiian:
```
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat misEng | tr -cs "'pkmnwlhaeiou" '[\n*]' |
sort -u | comm -12 - hwords > EngHaw
[classzfu@lnxsrv09 ~/351/hw2/new2]$ wc -w < EngHaw
6
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat EngHaw
e
halau
i
lau
po
wiki
```

misspelled Haw but true for English:
```
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat misHaw | tr -cs 'A-Za-z' '[\n*]' | sort
-u | comm -12 - words > HaiEng
[classzfu@lnxsrv09 ~/351/hw2/new2]$ wc -w < HaiEng
```

```
109
[classzfu@lnxsrv09 ~/351/hw2/new2]$ cat HaiEng
a
ail
ain
ake
al
ale
alen
all
amine
amp
ample
an
aph
aul
awk
e
ea
ee
el
em
emp
en
ep
epa
h
ha
han
hap
he
hei
hell
hem
hen
hi
hin
ho
how
howe
ia
ie
ile
imp
in
ion
iou
k
keep
kin
l
lan
le
lea
li
```

like
line
link
ll
ln
lo
lowe
m
mail
man
me
men
mi
ml
mo
mp
n
name
ne
nee
no
non
nu
num
o
om
on
one
op
ope
open
owe
own
p
pe
pell
people
plea
pu
u
ui
ula
ule
ume
ump
un
uni
w
wa
wan
we
wh
wha
who
wi

wo

---

lab2
sameln (bash)

```
#!/bin/bash

dir=$1
restore="$IFS"
IFS=$'\n'
counter=0
declare -a arr

files=`find $dir -maxdepth 1 -type f | sed "s/\(.*\)\/\(.*\)/\2/g" | grep -E
"^[^\.]" | sort`
hfiles=`ls -a $dir | grep '^\.' | sort`

for hfile in $hfiles
do
    if [ -L "$dir/$hfile" ]; then
        :
    elif [ ! -f "$dir/$hfile" ]; then
        :
    elif [ ! -r "$dir/$hfile" ]; then
        echo "$hfile is not readable" 1>&2
    else
        arr[$counter]="$dir/$hfile"
        counter=$counter+1
    fi
done

for file in $files
do
    if [ ! -r "$dir/$file" ]
    then
        echo "$file is not readable" 1>&2
    else
        arr[$counter]="$dir/$file"
        counter=$counter+1
    fi
done

for (( i=0; i<$counter; i++ ))
do
    for (( j=i+1; j<$counter; j++ ))
    do
        cmp -s "${arr[$i]}" "${arr[$j]}"
        if [ $? == 0 ]
        then
            ln -fP "${arr[$i]}" "${arr[$j]}"
                arr[$j]=arr[$i]
        fi
```

```
    done
done

IFS=$restore
```

---

lab3
comm.py python

```
#!/usr/bin/python
import string
import sys
import locale
from optparse import OptionParser


def rawComm(file1, file2, arg1, arg2, arg3):
    tmp = []
    while len(file1) != 0 or len(file2) != 0:
        if len(file2) == 0 or (len(file1) != 0 and file1[0] < file2[0]):
            if not arg1:
                tmp.append(file1[0])
            file1.remove(file1[0])
        elif len(file1) == 0 or (len(file2) != 0 and file1[0] > file2[0]):
            if not arg2:
                tmp.append((1-arg1)*"\t" + file2[0])
            file2.remove(file2[0])
        else: # file1[0] == file2[0]
            if not arg3:
                tmp.append((2-arg1-arg2)*"\t"+file1[0])
            file1.remove(file1[0])
            file2.remove(file2[0])
    return tmp


def unsortComm(file1, file2, arg1, arg2, arg3):
    tmp = []
    for l1 in file1:
        for l2 in file2:
            if l1 == l2:
                if not arg3:
                    tmp.append((2-arg1-arg2)*"\t"+l1)
                file2.remove(l2)
                break
        else: #correspond to l1 == l2
            if not arg1:
                tmp.append(l1)
    if not arg2:
        for l2 in file2:
            tmp.append((1-arg1)*"\t"+l2)
    return tmp
```

```
def main():
    locale.setlocale(locale.LC_COLLATE, 'C')
    usage_msg = """%prog [OPTION]... FILE1 FILE2 Output three columns, containing
lines only in FILE1, lines only in FILE2, and lines in both files."""
    parser = OptionParser(usage=usage_msg)
    parser.add_option("-1", action='store_true', dest="arg1", help="suppress
first column of output")
    parser.add_option("-2", action='store_true', dest="arg2", help="suppress
second column of output")
    parser.add_option("-3", action='store_true', dest="arg3", help="suppress
third column of output")
    parser.add_option("-u", action='store_true', dest="argu", help="allow input
of unsorted files")
    options, args = parser.parse_args(sys.argv[1:])
    if len(args) != 2:
        parser.error("Wrong number of operands!")
    if args[0] == "-":
        inputfile1 = sys.stdin.readlines()
    else:
        f = open(args[0], 'r')
        inputfile1 = f.readlines()
        f.close()
    if args[1] == "-":
        inputfile2 = sys.stdin.readlines()
    else:
        f = open(args[1], 'r')
        inputfile2 = f.readlines()
        f.close()
    if bool(options.argu):
        rawData = unsortComm(inputfile1, inputfile2, bool(options.arg1),
bool(options.arg2), bool(options.arg3))
    else:
        rawData = rawComm(inputfile1, inputfile2, bool(options.arg1),
bool(options.arg2), bool(options.arg3))
    out = ''.join(rawData)
    sys.stdout.write(out)


if __name__ == "__main__":
    main()
```

---

lab3.txt

```
#There are two folders: coreutils-7.6, the untarred one from downloaded tar;
the other is coreutils-install, dir for install (make install).



Download the file and copy it to the hw3 directionary
$ cp ~/Downloads/coreutils-7.6.tar.gz .
```

```
unzip the compresssed file
$ tar -xzf coreutils-7.6.tar.gz

make a directionary for coretil to be installed
$ mkdir coreutils-install

goto the directionary of coreutils-7.6 and configure with prefix
$ ./configure --prefix=/u/eng/class/classzfu/351/hw3/coreutils-install

then build and install
$ make
$ make install

go the bin directionary of the coreutils-install folder
$ cd ../coreutils-install/bin

test to identify the problem to see the difference in output
$ export LC_ALL='en_US.UTF-8'
$ ls -l ls
-rwxr-xr-x 1 classzfu class 457296 Apr 22 16:49 ls
$ ./ls -l ls
-rwxr-xr-x 1 classzfu class 457296 2017-04-22 16:49 ls
the installed version generates YYYY-MM-DD date instead of traditional Unix one



then try to modify the ls.c file
emacs ../../coreutils-7.6/src/ls.c

edit the file based on the - and + sign, - for deletion and + for addition
delete case_long_iso_time_style: (or search for the line number by M+g g)
        C+s case_long_iso_time_style: C+a C+k
delete for loop:
        C+s for (i = 0 C+a DOWN C+space DOWN DOWN DOWN DOWN DOWN DOWN DOWN BackS
        LEFT Enter Tab
type the added contents:
        long_time_format[i] = dcgettext (NULL, long_time_format[i], LC_TIME);
save and close ls.c:
        C+x C+s C+x C+c

build the modified version but not copy to the final dir (coreutils-install)
$ cd ..
$ make

change to /src (the fixed version) and create a file with a year old
$ cd src
$ touch tmp -d "Apr 19 2016"
$ ./ls tmp -l
-rw-r--r-- 1 classzfu class 0 Apr 19  2016 tmp
```

---

hw3.txt

Q1. Why did Brady's patch remove the line - "case_long_iso_time_style:"? Was it necessary to remove that line? Explain.

Not necessary.
case_long_iso_time_style: is just a direction for goto to follow,
since no goto use this dir, it can be deleted but not necessary.

Q2. If your company adopts this patched version of Coreutils instead of the default one, what else should you watch out for? Might this new version of Coreutils introduce other problems with your application, perhaps in countries where users don't speak English and don't understand English-format dates?

The problem is the "dcgettext" function in the modified version of ls.c.
If no valid translation is found under the environment of a special locale,
it will return long_time_format[i].

Q3. What happens when this script is invoked on an empty file like /dev/null, and why?

```
$ python randline.py /dev/null
Traceback (most recent call last):
  File "randline.py", line 70, in <module>
    main()
  File "randline.py", line 64, in main
    sys.stdout.write(generator.chooseline())
  File "randline.py", line 34, in chooseline
    return random.choice(self.lines)
  File "/usr/lib64/python2.7/random.py", line 274, in choice
    return seq[int(self.random() * len(seq))]
    # raises IndexError if seq is empty
IndexError: list index out of range
```

Trace the error message from the top to the bottom, we can identify that it is the random.py that the randline.py imports causes the problem. I open the /usr/lib64/python2.7/random.py and go to line 274.

```
def choice(self, seq):
    """Choose a random element from a non-empty sequence."""
    return seq[int(self.random() * len(seq))]
    # raises IndexError if seq is empty seq cannot access a empty file
```

Hence, when seq, (in this case, which is the only argument /dev/null) is empty, seq[...] will be meaningly.

Q4. What happens when this script is invoked with Python 3 rather than Python 2, and why? (You can run Python 3 on the SEASnet hosts by using the command python3 instead of python.)

first to add /usr/local/cs/bin to the $PATH

```
$ export PATH=$PATH:/usr/local/cs/bin

then invoke randline.py with python3
$ python3 randline.py tmp.txt
  File "randline.py", line 65
    except IOError as (errno, strerror):
                      ^
SyntaxError: invalid syntax

The SyntaxError is caused by the difference between the syntax of python2 & 3.
In Python3, cannot read an exception as a tuple


--------------------------------------------------------------------------------
---------

lab4
sfrob.c

#include <stdio.h>
#include <stdlib.h>

int frobcmp(char const *a, char const *b) {
        if (a == NULL || b == NULL){
                fputs("Invalid input for frobcmp", stderr);
                exit(1);
        }
        while ((*a != ' ') & (*b != ' ')) {
                if (((*a) ^ 42) == ((*b) ^ 42)) {
                        a++;
                        b++;
                }
                else if (((*a) ^ 42) - ((*b) ^ 42) > 0)
                        return 1;
                else
                        return -1;
        }
        if (*a == ' ') {
                if (*b == ' ')
                        return 0;
                return -1;
        }
        return 1;
}

int frobcaller(const void *a, const void *b) {
  return frobcmp(*(const char**)a,*(const char**)b);
}

int main() {
        const int nAlloc = 10;
        int iArr = 0;
        int sizeArr = nAlloc;
        int iStr = 0;
        int sizeStr = 0;
```

```
        int i = 0;

        char chr;
        char **words = (char**)malloc(sizeof(char*) * nAlloc);

        if (words == NULL){ //check for words memory allocation
                fputs("Error in memory allocation\n", stderr);
                exit(1);
        }

        while (1) {
          chr = getchar();
          if (feof(stdin))
            break;
          if (ferror(stdin)) { //error in getchar
            fputs("Error in input\n", stderr);
            exit(1);
          }

                if (iArr + 1 > sizeArr) {
                        words = (char**)realloc(words, (sizeArr +
nAlloc)*sizeof(char*));
                        if (words == NULL){
                                fputs("Error in memory reallocation\n", stderr);
                                exit(1);
                        }
                        sizeArr += nAlloc;
                }
                if (iStr == 0) {
                        words[iArr] = (char*)malloc(sizeof(char) * nAlloc);
                        if (words[iArr] == NULL){
                                fputs("Error in memory allocation\n", stderr);
                                exit(1);
                        }
                        sizeStr = nAlloc;
                }
                if (iStr + 1 > sizeStr) {
                        words[iArr] = (char*)realloc(words[iArr], (sizeStr +
nAlloc)*sizeof(char));
                        if (words[iArr] == NULL){
                                fputs("Error in memory reallocation\n", stderr);
                                exit(1);
                        }
                        sizeStr += nAlloc;
                }

                if (chr == ' ') { //store input char
                        words[iArr][iStr] = chr;
                        iArr++;
                        iStr = 0;
                        sizeStr = 0;
                }
                else {
                        words[iArr][iStr] = chr;
                        iStr++;
```

```
                }
        }

        if (iStr != 0) { //in case stdin does not have a trailing space
                words[iArr][iStr] = ' ';
                iArr++;
                iStr = 0;
        }

        qsort(words, iArr, sizeof(char*), frobcaller);

        for (i; i < iArr; i++) {
                while (words[i][iStr] != ' ') {
                        if (putchar(words[i][iStr]) == EOF) { //check for error
in output
                                fputs("Error in output\n", stderr);
                                exit(1);
                        }
                        iStr++;
                }
                if (putchar(' ') == EOF) { //check for error in output
                        fputs("Error in output\n", stderr);
                        exit(1);
                }
                iStr = 0;
                free(words[i]);
        }
        free(words);

        return 0;
}
```

---

lab5
tr2b.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int norep (char *arr){
  int i;
  int j;
  for (i = 0; i < strlen(arr); i++){
    char tmp = arr[i];
    for (j = i + 1; j < strlen(arr); j++){
      if (tmp == arr[j]){
        return 0;
      }
    }
  }
  return 1;
```

```
}

int main(int argc, char *argv[]){
  if (argc != 3){ //first argument is ./tr2b
    fputs("needs 2 arguments\n", stderr);
    exit(1);
  }
  char *from = argv[1];
  char *to = argv[2];
  int sizefrom = strlen(from);
  int sizeto = strlen(to);
  char cget;
  int i;
  if (sizefrom != sizeto){
    fputs("two arguments are not of the same length\n", stderr);
    exit(1);
  }
  if (!norep(from)){
    fputs("repeated characters in arg1, (operand from)\n", stderr);
    exit(1);
  }
  while(1){
    cget = getchar();
    if (ferror(stdin)){
      fputs("error in input", stderr);
      exit(1);
    }
    if (feof(stdin))
      break;
    for (i = 0; i < sizefrom; i++){
      if (cget == from[i]){
        putchar(to[i]);
        break;
      }
    }
    if (i == sizefrom)
      putchar(cget);
  }
  return 0;
}
```

---

lab5
tr2u.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

int norep (char *arr){
  int i;
```

```
  int j;
  for (i = 0; i < strlen(arr); i++){
    char tmp = arr[i];
    for (j = i + 1; j < strlen(arr); j++){
      if (tmp == arr[j]){
        return 0;
      }
    }
  }
  return 1;
}

int main(int argc, char *argv[]){
  if (argc != 3){ //the first operand is ./tr2u
    fputs("needs 2 arguments\n", stderr);
    exit(1);
  }
  char *from = argv[1];
  char *to = argv[2];
  int sizefrom = strlen(from);
  int sizeto = strlen(to);
  char cget;
  int i;
  if (sizefrom != sizeto){
    fputs("two arguments are not of the same length\n", stderr);
    exit(1);
  }
  if (!norep(from)){
    fputs("repeated characters in arg1, (operand from)\n", stderr);
    exit(1);
  }
  while (read(0, &cget, 1) == 1){ //0 is the file descriptor for stdin
    for (i = 0; i < sizefrom; i++){
      if (cget == from[i]){
        write(1, &to[i], 1);
        break;
      }
    }
    if (i == sizefrom)
      write(1, &cget, 1);
  }
  return 0;
}
```

---

lab5
lab.txt

1, 2
Basic idea is to first check for errors, including reporting an error from and to are not the same length, or if from has duplicate bytes, and number of arguments is not 2.

Then get a character either by buffered input getchar or unbuffered input read.
Use nested for loop to check if the char is in the 'from' operand.
If so, output the corresponding char in the same position in 'to' operand. If not, output itself.

3.
$gcc tr2b.c -o tr2b
$gcc tr2u.c -o tr2u
$dd if=/dev/urandom of=infile.txt bs=5120 count=1000
        get 5120 bytes of characters generated from /dev/urandom, and repeated the generated chunk for 1000 times and store to infile.txt

(b) $strace -o tr2bstrace1.txt ./tr2b 'a' 'b' < infile.txt
(a) $strace -o tr2bstrace2.txt ./tr2b 'a' 'b' < infile.txt > tmp2.txt
(b) $strace -o tr2ustrace1.txt ./tr2u 'a' 'b' < infile.txt
(a) $strace -o tr2ustrace2.txt ./tr2u 'a' 'b' < infile.txt > tmp1.txt
//file tr2ustrace1.txt and tr2ustace2.txt are too large (430MB)

Open tr2bstrace1.txt, we see "read(0, "\v\212?DpC#\2108
\20d\336/l\276E\"\273\236\202\356z\344\356\2449\307\2\45;F\5"..., 65536) = 65536"
getchar calls read function to read in 65536 bytes into buffer at a time.

In comparison, the 'read' functionin tr2u reads 1 bytes into buffer at a time.
The rest of bytes are left unbuffered.

4.
[classzfu@lnxsrv07 ~/351/week5]$ time ./tr2u 'a' 'x' < infile.txt > tmp1.txt

real   0m6.181s
user   0m0.259s
sys   0m5.816s
[classzfu@lnxsrv07 ~/351/week5]$ time ./tr2b 'a' 'x' < infile.txt > tmp2.txt

real   0m0.239s
user   0m0.184s
sys   0m0.010s

real time - time elapseed between invovation and termination
user time - how much CPU time did your program in user space (outside the kernal)
sys time - how much time was spent in OS/supervisor mode (doing system call)

As expected, tr2u uses much more time, especially in 'sys', due to its myriads of system calls (write() and read()).

Problem encountered:
For the generated 5MB file, there are many EOF("\377") character in the middle of the file.
My original implementation of tr2b use the eof as the indication for the end of input.

As a result, the process of copy stop even the file does not end.

Solution:
I fix tr2b by using if(feof(stdin)) statment to indicate the end of input, and ferror(stdin) for the input error.

---

lab5
sfrob.txt

1. Measure any differences in performance between sfrob and sfrobu using the time command.
Run your program on inputs of varying numbers of input lines, and estimate the number of comparisons as a function of the number of input lines.

I write a bash file to generate lines with specificed number

```
#!/bin/bash

export LC_ALL='C'
> $2

for (( i=0; i<$1; i++))
do
    R=$(($RANDOM%7+1))
    dd if=/dev/urandom bs=$R count=1 status=none >> $2
    echo -n ' ' >> $2
done
```

```
[classzfu@lnxsrv07 ~/351/hw5]$ ./bashfile.sh 1000 l1000.txt
[classzfu@lnxsrv07 ~/351/hw5]$ ./bashfile.sh 10000 l10000.txt
[classzfu@lnxsrv07 ~/351/hw5]$ ./bashfile.sh 100000 l100000.txt
[classzfu@lnxsrv07 ~/351/hw5]$ ./bashfile.sh 1000000 l1000000.txt
```

For the number of comparisons as a function of the number of input lines, it should be based on the algorithm of qsort.
By using a counter in during comparsion, and print out as stdout.

line	comp (after rounding)
1000	8500
10000	120000
100000	1500000
1000000	19000000

comparison = lines*log(lines), nlogn algorithm with some constant that should be omitted in Big-O notation.

---

2. Use the time command to compare the overall performance of sfrob, sfrobu, sfrobs, sfrobu -f, and sfrobs -f.

generate file with specified size

```
[classzfu@lnxsrv07 ~/351/hw5]$ dd if=/dev/urandom of=1mb.txt bs=1024000 count=1
1+0 records in
1+0 records out
1024000 bytes (1.0 MB) copied, 0.109437 s, 9.4 MB/s
[classzfu@lnxsrv07 ~/351/hw5]$ dd if=/dev/urandom of=2mb.txt bs=1024000 count=2
2+0 records in
2+0 records out
2048000 bytes (2.0 MB) copied, 0.216905 s, 9.4 MB/s
[classzfu@lnxsrv07 ~/351/hw5]$ dd if=/dev/urandom of=5mb.txt bs=1024000 count=5
5+0 records in
5+0 records out
5120000 bytes (5.1 MB) copied, 0.565169 s, 9.1 MB/s
[classzfu@lnxsrv07 ~/351/hw5]$ dd if=/dev/urandom of=10mb.txt bs=1024000 count=10
10+0 records in
10+0 records out
10240000 bytes (10 MB) copied, 1.08226 s, 9.5 MB/s
[classzfu@lnxsrv07 ~/351/hw5]$ dd if=/dev/urandom of=20mb.txt bs=1024000 count=20
20+0 records in
20+0 records out
20480000 bytes (20 MB) copied, 2.17918 s, 9.4 MB/s
```

---

lab5
sfrobs (bash)

```
#!/bin/bash

export LC_ALL='C'
from="\52\53\50\51\56\57\54\55\42\43\40\41\46\47\
\44\45\72\73\70\71\76\77\74\75\62\63\60\
\61\66\67\64\65\12\13\10\11\16\17\14\15\2\3\0\1\6\7\4\5\32\33\30\31\36\
\37\34\35\22\23\20\21\26\27\24\25\152\153\
\150\151\156\157\154\155\142\143\140\141\146\147\144\
\145\172\173\170\171\176\177\174\175\162\163\160\161\
\166\167\164\165\112\113\110\111\116\117\114\115\102\
\103\100\101\106\107\104\105\132\133\130\131\136\137\
\134\135\122\123\120\121\126\127\124\125\252\253\250\
\251\256\257\254\255\242\243\240\241\246\247\244\245\
\272\273\270\271\276\277\274\275\262\263\260\261\266\
\267\264\265\212\213\210\211\216\217\214\215\202\203\
\200\201\206\207\204\205\232\233\230\231\236\237\234\
\235\222\223\220\221\226\227\224\225\352\353\350\351\
\356\357\354\355\342\343\340\341\346\347\344\345\372\
\373\370\371\376\377\374\375\362\363\360\361\366\367\
\364\365\312\313\310\311\316\317\314\315\302\303\300\
\301\306\307\304\305\332\333\330\331\336\337\334\335\
\322\323\320\321\326\327\324\325"
```

```
export LC_ALL='C'
to="\0\1\2\3\4\5\6\7\10\11\12\13\14\15\16\17\20\21\22\23\24\25\26\27\30\
\31\32\33\34\35\36\37\40\41\42\43\44\45\46\47\50\51\52\53\54\55\56\57\60\
\61\62\63\64\65\66\67\70\71\72\73\74\75\76\77\100\101\102\103\104\105\106\
\107\110\111\112\113\114\115\116\117\120\121\122\123\124\125\126\127\130\
\131\132\133\134\135\136\137\140\141\142\143\144\145\146\147\150\
\151\152\153\154\155\156\157\160\161\162\163\164\165\
\166\167\170\171\172\173\174\175\176\177\200\201\202\
\203\204\205\206\207\210\211\212\213\214\215\216\217\
\220\221\222\223\224\225\226\227\230\231\232\233\234\
\235\236\237\240\241\242\243\244\245\246\247\250\251\
\252\253\254\255\256\257\260\261\262\263\264\265\266\
\267\270\271\272\273\274\275\276\277\300\301\302\303\
\304\305\306\307\310\311\312\313\314\315\316\317\320\
\321\322\323\324\325\326\327\330\331\332\333\334\335\
\336\337\340\341\342\343\344\345\346\347\350\351\352\
\353\354\355\356\357\360\361\362\363\364\365\366\367\
\370\371\372\373\374\375\376\377"

export LC_ALL='C'
if [ "$1" = "-f" ]; then
        tr $from $to | sort -f | tr $to $from
else
        tr $from $to | sort | tr $to $from
fi
```

---

lab5
sfrobu.c

```
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>
#include <ctype.h>

void errHandler(const char* err, int size) {
        write(2, err, size);
        exit(1);
}

int frobcmp(char const *a, char const *b) {
        if (a == NULL || b == NULL) {
                char tmp[] = "Invalid input for frobcmp\n";
                errHandler(tmp, sizeof(tmp));
        }
        while ((*a != ' ') & (*b != ' ')) {
                if (((*a) ^ 42) == ((*b) ^ 42)) {
                        a++;
                        b++;
                }
                else if (((*a) ^ 42) - ((*b) ^ 42) > 0)
                        return 1;
```

```
                else
                        return -1;
        }
        if (*a == ' ') {
                if (*b == ' ')
                        return 0;
                return -1;
        }
        return 1;
}

int frobcmpupper(char const *a, char const *b) {
        if (a == NULL || b == NULL) {
                char tmp[] = "Invalid input for frobcmp\n";
                errHandler(tmp, sizeof(tmp));
        }
        while ((*a != ' ') & (*b != ' ')) {
                char first = (*a) ^ 42;
                char second = (*b) ^ 42;
                if (first >= -1)
                        first = toupper(first);
                if (second >= -1)
                        second = toupper(second);
                if (first == second) {
                        a++;
                        b++;
                }
                else if (first - second > 0)
                        return 1;
                else
                        return -1;
        }
        if (*a == ' ') {
                if (*b == ' ')
                        return 0;
                return -1;
        }
        return 1;
}

int frobcaller(const void *a, const void *b) {
        return frobcmp(*(const char**)a, *(const char**)b);
}

int frobcallerupper(const void *a, const void *b) {
        return frobcmpupper(*(const char**)a, *(const char**)b);
}

void to2d(const char* words1d, int n, int fswitch) {
        const int nAlloc = 10;
        int iArr = 0;
        int sizeArr = nAlloc;
        int iStr = 0;
        int sizeStr = 0;
        int i;
```

```
	char chr;
	char **words = (char**)malloc(sizeof(char*) * nAlloc);

	if (words == NULL) { //check for words memory allocation
		char tmp[] = "Error in memory allocation\n";
		errHandler(tmp, sizeof(tmp));
	}

	for (i = 0; i < n; i++) {
		chr = words1d[i];

		if (iArr + 1 > sizeArr) {
			words = (char**)realloc(words, (sizeArr + nAlloc) *
sizeof(char*));
			if (words == NULL) {
				char tmp[] = "Error in memory reallocation\n";
				errHandler(tmp, sizeof(tmp));
			}
			sizeArr += nAlloc;
		}
		if (iStr == 0) {
			words[iArr] = (char*)malloc(sizeof(char) * nAlloc);
			if (words[iArr] == NULL) {
				char tmp[] = "Error in memory reallocation\n";
				errHandler(tmp, sizeof(tmp));
			}
			sizeStr = nAlloc;
		}
		if (iStr + 1 > sizeStr) {
			words[iArr] = (char*)realloc(words[iArr], (sizeStr +
nAlloc) * sizeof(char));
			if (words[iArr] == NULL) {
				char tmp[] = "Error in memory reallocation\n";
				errHandler(tmp, sizeof(tmp));
			}
			sizeStr += nAlloc;
		}

		if (chr == ' ') { //store input char
			words[iArr][iStr] = chr;
			iArr++;
			iStr = 0;
			sizeStr = 0;
		}
		else {
			words[iArr][iStr] = chr;
			iStr++;
		}
	}

	if (iStr != 0) { //in case stdin does not have a trailing space
		words[iArr][iStr] = ' ';
		iArr++;
		iStr = 0;
```

```
        }

        if (fswitch)
                qsort(words, iArr, sizeof(char*), frobcallerupper);
        else
                qsort(words, iArr, sizeof(char*), frobcaller);

        for (i = 0; i < iArr; i++) {
                while (words[i][iStr] != ' ') {
                        char cwrite = words[i][iStr];
                        if (write(1, &cwrite, 1) == -1) { //check for error in
output
                                char tmp[] = "Error in output\n";
                                errHandler(tmp, sizeof(tmp));
                        }
                        iStr++;
                }
                char tmp = ' ';
                if (write(1, &tmp, 1) == -1) { //check for error in output
                        char tmp[] = "Error in output\n";
                        errHandler(tmp, sizeof(tmp));
                }
                iStr = 0;
                free(words[i]);
        }
        free(words);
}


int main(int argc, char *argv[]) {
        //check argument
        int fswitch = 0;
        if (argc > 2) {
                char tmp[] = "Can only take one option: -f\n";
                errHandler(tmp, sizeof(tmp));
        }
        else if (argc == 2) {
                if (strcmp(argv[1], "-f") == 0)
                        fswitch = 1;
                else {
                        char tmp[] = "Can only take one option: -f\n";
                        errHandler(tmp, sizeof(tmp));
                }
        }


        //check regular file and size
        struct stat buffer;
        long rawSize;
        const int nAlloc = 10;

        if (fstat(0, &buffer) == -1){
                char tmp[] = "Error in getting file status\n";
                errHandler(tmp, sizeof(tmp));
        }
```

```
        if (S_ISREG(buffer.st_mode))
                rawSize = buffer.st_size;
        else
                rawSize = nAlloc;

        char *words1d = (char*)malloc(sizeof(char)*rawSize);


        //read input
        int n = 0;
        char cget;
        int readstat;
        while ((readstat = read(0, &cget, 1)) == 1) {
                if (n == rawSize){
                        words1d = (char*)realloc(words1d, sizeof(char)*(rawSize +
nAlloc));
                        rawSize += nAlloc;
                }
                words1d[n] = cget;
                n++;
        }
        if (readstat == -1) {
                char tmp[] = "Error in reading input\n";
                errHandler(tmp, sizeof(tmp));
        }

        //convert to 2d array and qsort
        to2d(words1d, n, fswitch);

        return 0;
}
```

---

lab6
main.c pthread

```
#include "raymath.h"
#include "shaders.h"

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#include <pthread.h>

static int threadUsed;
static Vec3 camera_pos;
static Vec3 camera_dir;
static Vec3 bg_color;
static double camera_fov;
static double pixel_dx;
```

```
static double pixel_dy;
static double subsample_dx;
static double subsample_dy;
static scene_t scene;
static float **color2d;

static double dirs[6][3] =
{ {1,0,0}, {-1,0,0}, {0,1,0}, {0,-1,0}, {0,0,1}, {0,0,-1} };
static const int opposites[] = { 1, 0, 3, 2, 5, 4 };


static void
add_sphereflake( scene_t* scene, int sphere_id, int parent_id, int dir,
                 double ratio, int recursion_level )
{
    sphere_t* parent = &scene->spheres[parent_id];
    sphere_t* child = &scene->spheres[sphere_id];

    /* start at parents origin */
    mul( child->org, dirs[dir], (1.+ratio)*parent->rad );
    add( child->org, child->org, parent->org );
    child->rad = parent->rad * ratio;
    copy( child->color, parent->color );
    child->shader = parent->shader;
    scene->sphere_count++;
}

static int
recursive_add_sphereflake( scene_t* scene, int parent_id, int parent_dir,
                           int sphere_id, int dir,
                           int recursion_level, int recursion_limit )
{
    const double ratio = 0.35;

    add_sphereflake( scene, sphere_id, parent_id, dir, ratio, recursion_level );
    if( recursion_level > recursion_limit )
    {
        return sphere_id + 1;
    }

    /* six children, one at each cardinal point */
    parent_id = sphere_id;
    sphere_id = sphere_id + 1;
    for( int child_dir=0; child_dir<6; ++child_dir )
    {
        /* skip making spheres inside parent */
        if( parent_dir == opposites[child_dir] ) continue;
        sphere_id = recursive_add_sphereflake( scene, parent_id, parent_dir,
                                               sphere_id, child_dir,
                                               recursion_level + 1,
                                               recursion_limit );
    }
    return sphere_id;
}
```

```
static scene_t
create_sphereflake_scene( int recursion_limit )
{
    scene_t scene;
    Vec3 color;
    sphere_t* sphere;

    init_scene( &scene );

    // Pantone UC Gold 122
    add_light( &scene, 2, 5, 0, 0.996, 0.733, 0.212 );

    // Pantone UCLA Blue (50,132,191)
    add_light( &scene, -5, 3, -5, 0.196, 0.517, 0.749 );

    int max_sphere_count = 2 + powl( 6, recursion_limit + 2 );
    scene.spheres = realloc( scene.spheres,
                             max_sphere_count*sizeof( sphere_t ) );
    if( !scene.spheres )
    {
        fprintf( stderr, "Failed to get memory for sphereflake.  aborting.\n" );
        exit( -1 );
    }

//    sphere = &(scene.spheres[0]);
//    set( sphere->org, -0.5, -1.0, 0 );
//    sphere->rad = 0.75;
//    set( color, 0.85, 0.25, 0.25 );
//    copy( sphere->color, color );
//    sphere->shader = mirror_shader;


    /* center sphere is special, child inherent shader and color */
    sphere = &(scene.spheres[0]);
    scene.sphere_count++;
    set( sphere->org, 0, -1, 0 );
    sphere->rad = 0.75;
    set( color, 0.75, 0.75, 0.75 );
    copy( sphere->color, color );
    sphere->shader = mirror_shader;
    recursive_add_sphereflake( &scene,
                               0, /* parent is the first sphere */
                               -1, /* -1 means no dir, make all children */
                               1, /* next free sphere index */
                               2, /* starting dir */
                               0, /* starting recursion level */
                               recursion_limit );

    return scene;
}

static void
free_scene( scene_t* arg )
{
    free( arg->lights );
```

```
    arg->light_count = 0;
    free( arg->spheres );
    arg->sphere_count = 0;
}

/******
 * Constants that have a large effect on performance */

/* how many levels to generate spheres */
enum { sphereflake_recursion = 3 };

/* output image size */
enum { height = 131 };
enum { width = 131 };

/* antialiasing samples, more is higher quality, 0 for no AA */
enum { halfSamples = 4 };
/******/

/* color depth to output for ppm */
enum { max_color = 255 };

/* z value for ray */
enum { z = 1 };


inline
void errorHandler(const char* msg){
    fprintf(stderr, "%s\n", msg);
    exit(1);
}

static void initConst(){
    scene = create_sphereflake_scene( sphereflake_recursion );
    /* Write the image format header */
    /* P3 is an ASCII-formatted, color, PPM file */
    printf( "P3\n%d %d\n%d\n", width, height, max_color );
    printf( "# Rendering scene with %d spheres and %d lights\n",
scene.sphere_count, scene.light_count);
    set( camera_pos, 0., 0., -4. );
    set( camera_dir, 0., 0., 1. );
    camera_fov = 75.0 * (PI/180.0);
    set( bg_color, 0.8, 0.8, 1 );
    pixel_dx = tan( 0.5*camera_fov ) / ((double)width*0.5);
    pixel_dy = tan( 0.5*camera_fov ) / ((double)height*0.5);
    subsample_dx = halfSamples ? pixel_dx / ((double)halfSamples*2.0) : pixel_dx;
    subsample_dy = halfSamples ? pixel_dy / ((double)halfSamples*2.0) : pixel_dy;
}

void* threadFunc(void *arg){
    /* for every pixel */
    for( int px=*((int*)arg); px<width; px+=threadUsed){
        const double x = pixel_dx * ((double)( px-(width/2) ));
        for( int py=0; py<height; ++py )
        {
```

```
        const double y = pixel_dy * ((double)( py-(height/2) ));
        Vec3 pixel_color;
        set( pixel_color, 0, 0, 0 );

        for( int xs=-halfSamples; xs<=halfSamples; ++xs )
        {
            for( int ys=-halfSamples; ys<=halfSamples; ++ys )
            {
                double subx = x + ((double)xs)*subsample_dx;
                double suby = y + ((double)ys)*subsample_dy;

                /* construct the ray coming out of the camera, through
                 * the screen at (subx,suby)
                 */
                ray_t pixel_ray;
                copy( pixel_ray.org, camera_pos );
                Vec3 pixel_target;
                set( pixel_target, subx, suby, z );
                sub( pixel_ray.dir, pixel_target, camera_pos );
                norm( pixel_ray.dir, pixel_ray.dir );

                Vec3 sample_color;
                copy( sample_color, bg_color );
                /* trace the ray from the camera that
                 * passes through this pixel */
                trace( &scene, sample_color, &pixel_ray, 0 );
                /* sum color for subpixel AA */
                add( pixel_color, pixel_color, sample_color );
            }
        }

        /* at this point, have accumulated (2*halfSamples)^2 samples,
         * so need to average out the final pixel color
         */
        if( halfSamples )
        {
            mul( pixel_color, pixel_color,
                 (1.0/( 4.0 * halfSamples * halfSamples ) ) );
        }

        /* done, final floating point color values are in pixel_color */
        float scaled_color[3];
        scaled_color[0] = gamma( pixel_color[0] ) * max_color;
        scaled_color[1] = gamma( pixel_color[1] ) * max_color;
        scaled_color[2] = gamma( pixel_color[2] ) * max_color;

        color2d[px*height+py] = (float*) malloc (sizeof(float)*3);
        if (!color2d[px*height+py])
            errorHandler("Error in memory allocation");

        /* enforce caps, replace with real gamma */
        for( int i=0; i<3; i++)
            color2d[px*height+py][i] = max( min(scaled_color[i], 255), 0);
    }
}
```

```
    return NULL;
}

int
main( int argc, char **argv )
{
    int nthreads = argc == 2 ? atoi( argv[1] ) : 0;
    if( nthreads < 1 )
    {
      fprintf( stderr, "%s: usage: %s NTHREADS\n", argv[0], argv[0] );
      return 1;
    }

    initConst();
    threadUsed = nthreads < width ? nthreads : width;

    pthread_t *tid = (pthread_t*) malloc (sizeof(pthread_t) * threadUsed);
    if (!tid)
        errorHandler("Error in memory allocation");

    int *para = (int*) malloc (sizeof(int) * threadUsed);
    if (!para)
        errorHandler("Error in memory allocation");

    int i;
    for (i = 0; i < threadUsed; i++){
        para[i] = i;
    }

    color2d = (float**) malloc (sizeof(float*) * width * height);
    if (!color2d)
        errorHandler("Error in memory allocation");

    for (i = 0; i < threadUsed; i++){
        if (pthread_create (tid + i, NULL, threadFunc, para + i))
            errorHandler("Error in creating threads");
    }

    for (i = 0; i < threadUsed; i++){
        if (pthread_join(tid[i], NULL))
            errorHandler("Error in joining threads");
    }

    for (i = 0; i < width*height; i++){
        if ((i != 0) && (i % height == 0))
            printf("\n");
        /* write this pixel out to disk. ppm is forgiving about whitespace,
         * but has a maximum of 70 chars/line, so use one line per pixel
         */
        printf( "%.0f %.0f %.0f\n", color2d[i][0], color2d[i][1], color2d[i][2]
);
        free(color2d[i]);
    }
    printf("\n");
```

```
    free(color2d);
    free(para);
    free(tid);
    free_scene( &scene );

    if( ferror( stdout ) || fclose( stdout ) != 0 )
        errorHandler("Output error");

    return 0;
}
```

---

lab7
log.txt

1. setup steps

Open the virtual machine software running ubuntu system or directly run the system. Then use the terminal to execute the following commands.

```
$ dpkg --get-selections | grep openssh
```

Notice that still the openssh-server needs to be installed.

```
$ sudo apt-get install openssh-server
```

(the password for the ubuntu system user may be needed)
Press Y to continue
Type the dpkg command to confirm the installation

```
$ dpkg --get-selections | grep openssh
        openssh-client install
        openssh-server install
        openssh-sftp-server install
```

---

2. server steps

first generate the key pairs of the server

```
ubuntu@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa): (Press Enter directly to save key pairs to the default location)
Enter passphrase (empty for no passphrase): memes
Enter same passphrase again: memes
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:iNBnb9ElriP31nIQXFUD2EvyIWCEgvXaNuYMM5r8iN8 ubuntu@ubuntu
The key's randomart image is:
```

```
+---[RSA 2048]----+
|     o.  o+o =ooo.|
|    o ....+ B +  .|
|   . . +.. = = o  |
|    . +oo o . o   |
|     .=o=S .      |
|   . o O+.o o     |
|    +   o  + o    |
|   . +    . o     |
|  ..o E           |
+----[SHA256]-----+
```

To get the public key
        $ cat ~/.ssh/id_rsa.pub
To get the private key
        $ cat ~/.ssh/id_rsa id_rsa.pub

Add user and the corresponding password.
(We set the home directory to be mark and the username to be fzp)
(set the passphrase for username fzp to be memes1)
        $ sudo useradd -d /home/mark -m fzp
        $ sudo passwd
                memes1
                memes1

Change directory to the user directory with homedir_name mark:
        $ cd /home/mark

Create .ssh directory for new user
        $ sudo mkdir .ssh

Change owenership and permission on .ssh directionary
        $ sudo chown -R fzp .ssh (change ownership. client owner of the file)
        $ sudo chmod 700 .ssh

Check the server IP address:
        $ ipconfig
                inet adddress: 10.97.85.46

---

3. client steps

now use another computer with a different IP address to act as the client

1) generate public and private keys
        $ ssh-keygen

copy the public key to server for keey-based client authentication (instead of password-based one, in which password sent may be compromised)
passphrase 'memes' should be entered
        $ ssh-copy-id -i fzp@10.97.85.46

add private key to authentication agent (ssh-agent)
no prompt for passphrase then after

```
        $ ssh-add

The overall effect:
        ubuntu@ubuntu:~$ ssh-keygen
        Generating public/private rsa key pair.
        Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
ssh-copy-id fzp@10.97.85.46
        Enter passphrase (empty for no passphrase): memes1
        Enter same passphrase again: memes1
        Your identification has been saved in ssh-copy-id fzp@10.97.85.46.
        Your public key has been saved in ssh-copy-id fzp@10.97.85.46.pub.
        The key fingerprint is:
        SHA256:EJI5nHka0d49ZiHXEDs42zmH2TfQn+DOYaycTAqbVUQ ubuntu@ubuntu
        The key's randomart image is:
        +---[RSA 2048]----+
        |   .oB.    +E    |
        |    O.o...+...   |
        |     *..o+oo...  |
        |    . ...+**o....|
        |       .S+*oo=o..|
        |        = =o*... |
        |       o . = o   |
        |                 |
        |                 |
        +----[SHA256]-----+

2) SSH to server (user:fzp, pasword:memes1)
        ubuntu@ubuntu:~$ ssh fzp@10.97.85.46
        The authenticity of host '10.97.85.46 (10.97.85.46)' can't be
established.
        ECDSA key fingerprint is
SHA256:sgwjO6nmfiPIVpMXdyu4c9AWsqywSmlCEI9F1qEY1mA.
        Are you sure you want to continue connecting (yes/no)? yes
        Warning: Permanently added '10.97.85.46' (ECDSA) to the list of known
hosts.
        fzp@10.97.85.46's password: memes1
        Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-22-generic x86_64)

         * Documentation:  https://help.ubuntu.com
         * Management:     https://landscape.canonical.com
         * Support:        https://ubuntu.com/advantage



        The programs included with the Ubuntu system are free software;
        the exact distribution terms for each program are described in the
        individual files in /usr/share/doc/*/copyright.

        Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
        applicable law.

Now SSH with the option -X to enable X Window System for graphical interface
        ubuntu@ubuntu:~$ ssh -X fzp@10.97.85.46
        fzp@10.97.85.46's password: memes1
        Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-22-generic x86_64)
```

```
         * Documentation:  https://help.ubuntu.com
         * Management:     https://landscape.canonical.com
         * Support:        https://ubuntu.com/advantage


        Last login: Mon May 15 15:56:28 2017 from 10.97.85.48
```

Run the testing command to show X Window System works correctly, applications like firefox browser is poped up on the client

```
        $ xterm
        $ firefox
```

To check whether the firefox process is loaded on server instead of on the client, we can check the process running on the server by typing the following commands on the server

```
        $ ps -aux | grep firefox
                fzp 28121 12.7 2.8 990876 229780 pts/2 Sl+ 16:06 0:02
/usr/lib/firefox/firefox
                ubuntu 28203 0.0 0.0 21296 1028 pts/1 S+ 16:06 0:00 grep
--color=auto firefox
```

To see what user 'fzp' is running

```
        $ ps -aux | grep fzp
```

---

lab7
hw.txt

1. Secure
The network is secure because the asymmetic (public-key) encryption system ensures the confidentialiy of the communcation.
The message sent to either end is encrypted by the public key.
The only way to decrypt bytes going across the network is by the private key.
So, this system is different from symmetic key encryption.

(1) Secure
The key typed is insufficient for others to figure out the private key.
Even if I type the passphrase with the exact same letters or digits, the resulting keys are different.

(2) Insecure
If the system directionary is located on the USB drive, the private key along with the system is also stored on the USB.
Hence, they can get the private key to decrypt the bytes on the network if they are smart enough.

2.
The problem is that the public key is sent along with the tar file.

As a result, a hacker can replace the original public key to be a new public key that is used to along with his private key.
So, gpg --verify command only verifies that the public key inside the tar file can decrypt the detached signature and the resulting tar matches the existing tar.
Nothing to do with personal creation of the tar file.

The solution can be that post the public key on the Internet, so the receiver of the signature can verify the validation of the public key in the tar file.
If keys are the same, the gpg --verify command will work.
If not, it means that the public key sent in tar file is tampered.

---

lab8
randmain.c dynamic linking

```
#include "randcpuid.h"
#include <errno.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

static bool
writebytes (unsigned long long x, int nbytes)
{
  int ndigits = nbytes * 2;
  do
    {
      if (putchar ("0123456789abcdef"[x & 0xf]) < 0)
        return false;
      x >>= 4;
      ndigits--;
    }
  while (0 < ndigits);

  return 0 <= putchar ('\n');
}

/* Main program, which outputs N bytes of random data.  */
int
main (int argc, char **argv)
{
  /* Check arguments.  */
  bool valid = false;
  long long nbytes;
  if (argc == 2)
    {
      char *endptr;
      errno = 0;
      nbytes = strtoll (argv[1], &endptr, 10);
      if (errno)
```

```
      perror (argv[1]);
    else
      valid = !*endptr && 0 <= nbytes;
  }
if (!valid)
  {
    fprintf (stderr, "%s: usage: %s NBYTES\n", argv[0], argv[0]);
    return 1;
  }

/* If there's no work to do, don't worry about which library to use.  */
if (nbytes == 0)
  return 0;

/* Now that we know we have work to do, arrange to use the
   appropriate library.  */

void *dl_handle;
unsigned long long (*rand64) (void);
char* error;

if (rdrand_supported ())
  {
    dl_handle = dlopen("randlibhw.so", RTLD_LAZY);
    if((error = dlerror()) != NULL)
    {
      fprintf(stderr, "dlopen error - %s\n", error);
      return 1;
    }
    rand64 = dlsym(dl_handle, "rand64");
    if ((error = dlerror()) != NULL)
    {
      fprintf(stderr, "dlsym error - %s\n", error);
      return 1;
    }
  }
else
  {
    dl_handle = dlopen("randlibsw.so", RTLD_LAZY);
    if((error = dlerror()) != NULL)
    {
      fprintf(stderr, "dlopen error - %s\n", error);
      return 1;
    }
    rand64 = dlsym(dl_handle, "rand64");
    if ((error = dlerror()) != NULL)
    {
      fprintf(stderr, "dlsym error - %s\n", error);
      return 1;
    }
  }

int wordsize = sizeof rand64 ();
int output_errno = 0;
```

```
  do
    {
      unsigned long long x = rand64 ();
      int outbytes = nbytes < wordsize ? nbytes : wordsize;
      if (!writebytes (x, outbytes))
      {
        output_errno = errno;
        break;
      }
      nbytes -= outbytes;
    }
  while (0 < nbytes);

  if (fclose (stdout) != 0)
    output_errno = errno;

  if (output_errno)
  {
    errno = output_errno;
    perror ("output");
    return 1;
  }

  if (dlclose(dl_handle))
  {
    fprintf(stderr, "dlclose error - %s\n", dlerror());
    return 1;
  }

  return 0;
}
```