Project 2:

**Lab:**

#!/bin/sh

file\_list=`ls -a | sort`

SAVEIFS=$IFS #restore environment vars

IFS=$(echo -en "\n\b")

for file\_a in $file\_list;

do

for file\_b in $file\_list;

do

if [ $file\_a = $file\_b ]

then

continue

fi

echo "Processing:"${file\_a}" & "${file\_b};

filediff=comm $file\_a $file\_b -3

if [ z $filediff ];

then

echo "yes, same";

fi

done

done

IFS=$SAVEIFS

Commands:

1) tr -c 'A-Za-z' '[\n\*]'

The -c command is the complement, so this expression made every place where there was a non-alphabetical letter character into a new line.

2) tr -cs 'A-Za-z' '[\n\*]'

This command uses the -s flag in addition to the -c, so it will get rid of repeated ocurrences of spaces essentially outputing the same thing as before, except without all the new lines.

3) tr -cs 'A-Za-z' '[\n\*]' | sort

This command took the output from before (replacing all non-albphabets with new lines and eliminating repeating new lines) and then sorting that file alphabetically. It puts each word on a new line.

4) tr -cs 'A-Za-z' '[\n\*]' | sort -u

This command took the output from before (replacing all non-albphabets with new lines and eliminating repeating new lines) and then sorting that file alphabetically. It puts each word on a new line. Then due to the -u command, it eliminates all repeating instances of the same word when it sorts them.

5) tr -cs 'A-Za-z' '[\n\*]' | sort -u | comm - words

This command compares the sorted webpage file (that has been done with the regex commands above). The first column are the words unique to file 1 (webpage.txt), the 2nd column are words unique to file 2 (words), and column 3 is words that are the same on both pages.

6) tr -cs 'A-Za-z' '[\n\*]' | sort -u | comm -23 - words

This command compares the sorted webpage file (from the previous regex commands above), but surpresses columns 2 and 3 so that you only see column 1. This means you see all the words that are "incorrect" compared to our dictionary as these words are only seen in the webpage.txt file.

**Homework:**

**(buildwords)**

#!/bin/bash

#Find all the </td> from the input using <&0

grep '<td>\(..\*\)<\/td>' <&0 |

#Removes all HTML Tags

sed -e 's/<[^>]\*>//g' |

#Removes spaces and tags

sed 's/^[ \t]\+//g' |

#Removes every 2nd line because 1st line is English

sed '1~2d'# |

#replaces ASCII grave with apostrophe

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

#Converts upper case to lower

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

#Removes anything that is not Hawaiian characters

tr -cs [pk\'mnwlhaeiou] '\n' |

#Sorts the entire file and removes duplicate

sort –u

**Sameln:**

#!/bin/bash

#save the IFS before modifying anything

SAVEIFS=$IFS

set -f # take out the glob so that \*, etc, dont work as well

IFS=$'\n'

currDir=$1

declare -a fileArray

count=0

cd $currDir

#grabs all files that only start with the dot operator and sorts

dotFiles=`ls -a | grep "^\." | sort`

for file1 in $dotFiles

do

if [ ! -L "$file1" ] # check that file is not symbolic link

then

if [ -f "$file1" ] #check that file is a REGULAR

then

if [ -r "$file1" ] # readeable file

then

fileArray[$count]="$file1" #adds file1 to the array

let count=count+1

else

echo "The file, $file1, is unreadeable."

fi

fi

fi

done

#grabs all non-dot files that are left

nonDotFiles=`ls | sort`

for file2 in $nonDotFiles

do

if [ ! -L "$file2" ] #check not symbolic link

then

if [ -f "$file2" ]

then

if [ -r "$file2" ] then

fileArray[$count]="$file2" #adds file2 to the array

let count=count+1

else

echo "The file, $file2, is unreadeable."

fi

fi

fi

done

#compares each file to all the files after. Since it is sorted in alpha order

# it will always choose the file lexicographically first to use as the orig.

# and make anything after into a hard link

for (( i=0; i<$count-1; i++ ))

do

for (( j=$((i+1)); j<$count; j++ ))

do

cmp -s -- "${fileArray[$i]}" "${fileArray[$j]}"

if [ $? -eq "0" ] #if the files are the same, then link them

then

rm -- "${fileArray[$j]}" # remove the original, then link

ln -- "${fileArray[$i]}" "${fileArray[$j]}"

fi

done

done

#restore IFS after modifications, restore the glob after the program

set +f

IFS=$SAVEIFS

Project 3

Randline.py

def convertToList(fileName):

f = open(fileName, 'r') #reading in the file and converting it into a list

convertedList = list(f.readlines())

f.close()

return convertedList

def main():

version\_msg = "%prog 1.0"

usage\_msg = """%prog [OPTION/S] ... FILE1 ... FILE2

Compares two files like the comm POSIX command. Takes -1,2,3 and/or -u as options."""

#setting up the parser

parser = OptionParser(version=version\_msg, usage=usage\_msg)

parser.add\_option("-1", action="store\_true", dest="hideOne", help="Suppress column one, lines unique to file One", default=False)

parser.add\_option("-2", action="store\_true", dest="hideTwo", help="Suppress column two, lines unique to file Two", default=False)

parser.add\_option("-3", action="store\_true", dest="hideThree", help="Suppress column three, lines that are common to both files", default=False)

parser.add\_option("-u", action="store\_true", dest="areUnsorted", help="Compare two unsorted inputs", default=False)

options, args = parser.parse\_args(sys.argv[1:])

if len(args) != 2:

parser.error("You either have too few operands or too many. Remember the format is [options] file1 file2")

try:

if args[0] == '-': # one of the inputs MAY be a - which means to read in from standard input

fileOne = list(sys.stdin.readlines())

else:

fileOne = convertToList(args[0]) # convert the file into a list to sort it easily

if args[1] == '-':

fileTwo = list(sys.stdin.readlines())

else:

fileTwo = convertToList(args[1]) #convver the second file into a list to sort it easily

if options.areUnsorted == False : #this means that the files should be sorted to work. If not sorted, throw error

if (fileOne != sorted(fileOne) or fileTwo != sorted(fileTwo)) :

parser.error("Either one of your files is unsorted. Please sort them first and then put them into this program. Or use the -u command to compare unsorted files")

except IOError as err:

errorno, strerror = srr.args

parser.error("I/O error({0}): {1}".format(errorno, strerror))

commFiles = comm(fileOne, fileTwo, options)

commFiles.sort()

if \_\_name\_\_ == "\_\_main\_\_": #invoking the main method

main()

import random, sys, locale, string

from optparse import OptionParser

class comm: #creating a class comm that reads in the files

def \_\_init\_\_(self, fOne, fTwo, inputOptions):

self.fOne = fOne

self.fTwo = fTwo

self.inputOptions = inputOptions

def sort(self):

#first we will cover the option that both the files are sorted

currF1 = 0

currF2 = 0

fileTabMultiplier = 0 #since python supports multiplying strings to repeat it, I am using this to multiply tabs for proper spacing

if self.inputOptions.hideOne == True and self.inputOptions.hideTwo == True:

fileTabMultiplier = -2

elif self.inputOptions.hideOne == True and self.inputOptions.hideTwo == False:

fileTabMultiplier = -1

elif self.inputOptions.hideOne == False and self.inputOptions.hideTwo == True:

fileTabMultiplier = -1

elif self.inputOptions.areUnsorted == True:

currF1 = 0

currF2 = 0

col2 = []

col3 = []

for word2 in self.fTwo:

#go through all the words in file2 and compare to file1 to make sure unique to file2

while currF1 < len(self.fOne):

if self.fOne.count(word2) == 0:

col2.append(word2)

break

elif word2 != self.fOne[currF1]:

currF1 +=1

elif word2 == self.fOne[currF1]:

if col3.count(word2) < min(self.fOne.count(word2), self.fTwo.count(word2)):

col3.append(word)

if self.inputOptions.hideThree == True:

sys.stdout.write("")

else:

sys.stdout.write("\t"\*(fileTabMultiplier+2) + self.fTwo[currF2])

if self.fTwo.count(word2) > self.fOne.count(word2):

col2.append(word2)

break

currF1 = 0

only the indents matter, no brackets!

Project 4

**Lab**

3) I used the command "git tag > git-tags.txt" to put the tags used for

Diffutils into the file git-tags.txt

5) To checkout the correct version of DiffUtils, I used the command " git

checkout v3.0 -b diffUtils3.0".

6) To patch, I used the command "patch -p1 < quote-patch.txt". It prompted me for some files and all I needed to do was hit "return" and then press "y" to skip over those files. The patch worked flawlessly without any errors (ie. I had no problem applying the patch). There were a few failed hunks, and the rejected parts were saved to various files, like "README.rej" and "README-hacking.rej".

7) I used "git status" to check the status of the current git directory, and

I saw that there were several files that were needed to be stages for commit.

# modified: NEWS

# modified: README

# modified: TODO

# modified: doc/diagmeet.note

# modified: ms/config.bat

# modified: ms/config.site

# modified: po/en.po

# modified: src/analyze.c

# modified: src/cmp.c

# modified: src/context.c

# modified: src/diff.c

# modified: src/diff.h

8) To work with reverting back changes, I read up on the Emacs commands like

vc-diff and vc-revert and diff-apply-hunk.

I ran git status to check that indeed these files are no longer needed to be added to the stage for commit because git detects no changes in them (since I reverted them).

11) I typed "git status" again, and once again only those six files were

showing again. However there were many files in the "untracked" portion of

the status bar, and I do not need these. Thus I should delete them with the

command "rm \_\_fileName\_\_". I also do NOT want to remove my lab files like

git-log.txt and git-tags.txt and quote-patch.txt, so I copied them to

another folder. Alternatively, I can also do **git -clean -f.**

12) I ran the command "git diff | wc -l" and saw that my git diff file was

312 lines long. Since this is between 300 and 400, I feel that it is

appropiate and since I followed the directions to the best of my ability, I

believe this should work. I used the command "git diff >

quote-3..0-patch.txt" to put the diff output into a new file.

13) I then looked into README-hacking to see the next instructions

regarding patching the file. These steps included running ./bootstrap, ./

configure, make and make check. Thus I did the following commands ->

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

./bootstrap

wget http://web.cs.ucla.edu/classes/fall16/cs35L/assign/diffutils-gets.diff

-O patch2

patch -p0 < patch2

./configure

make

make check

After doing the above steps, I typed in "src/diff . -" and "src/diff --help" and saw

that the ascii grave character ` was changed into a ' everywhere, which is

exactly what we wanted.

14) To check my version of the patch with the original, I ran the command

"cp -r diffutils/ diffutils-3.0" and "cp -r diffutils/ diffutils-3.0-patch"

What this has done is create two new directories with the same code base.

However, I need to revert back to the older patch in the 3.0 folder, so I

used "git checkout v3.0" to change it to the pre-patch version.

I ran the command "./diffutils/src/diff -pru diffutils-3.0

diffutils-3.0-patch > quote-3.0-test.txt"

15) I ran the command "./diffutils/src/diff quote-3.0-test.txt

quote-3.0-patch.txt" and saw that there were some differences that were outputted. This meant that the files were NOT identical, and after loooking through the code, it appears that the differences are not innocous as the strings in the code appear to be different (resulting in different behavior than what we wanted in the patch).

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

./bootstrap

wget http://web.cs.ucla.edu/classes/fall16/cs35L/assign/diffutils-gets.diff

-O patch2

patch -p0 < patch2

./configure

make

make check

**Homework:**

1) Touch hw4.txt

2) I used "cd diffutls-3.0" to go into the directory where I had the v3.0

checked out, and then I used the command "git checkout v3.0" to make sure I checked out the right one. Then I used the command "git branch quote" to create a new branch for the homework assignment. I then used the command "git checkout quote" to checkout this new branch to use.

3) I ran the command, "patch -p1 < quote-3.0-patch.txt" to patch this new branch. For every file that was being patched, it asked me, "Reversed or previously applied patch detected! Assume -R [n]". I replied y to all of these (this meant that patch would try to revert the old patches). Then I ran the command again "patch -p1 < quote-3.0-patch.txt" and got the files to properly patch.

5) I changed 6 different files (the .c files), so I wrote a small

description of which files were changed, and what I did for each file. This was the content of my file:

2016-10-21 Sahil Manoj Gandhi <gandhi@lnxsrv07.seas.ucla.edu>

\* src/cmp.c - Changed ascii grave to apostrophe in comments.

\* src/diff.c - Changed ascii grave to apostrope in comments.

\* src/diff3.c - Changed ascii grave to apostrophe in comments.

\* src/dir.c - Changed ascii grave to apostrophe in comments.

\* src/diff.c - Changed ascii grave to apostrophe in comments.

\* src/util.c - Changed ascii grave to apostrophe in comments.

6) I added the files to my workspace by doing "git add ." and then doing

"git commit -F ChangeLog" to commit my changes with the ChangeLog as my commit message. I could check that my commit actually worked by typing "git log" and it showed my most recent commit as being:

7) I used the command "git format-patch -1 --stdout HEAD >

formatted-patch.txt". I used "wc -l formatted-patch.txt" to see the amount of lines it contains, and it was 656 lines.

8) I checked out v3.0 by running "git checkout v3.0" and then made a new temporary branch called "partner" by running "git checkout -b partner". Now I copied over my friend's file () to my system and then used the command "git am < formatted-patch.txt" to see if it worked. The output of that command was as followed:

git am < formatted-patch1.txt

Applying: 2016-10-21 Jonathan Michael Zaturensky <

zaturens@lnxsrv09.seas.ucla.edu>

I then invoked the following commands:

./boostrap

./configure

./make

./make check

Now I used the following comppands to check whether his patch actually works:

src/diff . - -> this outputted the correct use of '-' rather than `-'

src/diff --help -> this also outputted the correct use of 'FILE1 FILE2'

rather than `FILE1 FILE2'

sections.

10) To get gitk to work on my windows machine, I first installed Xming to host an X server on my local machine. I then used the X11 form of putty to ssh into the lnxsrv02 and was able to go to the directory: "~eggert/src/gnu/emacs" and execute the commmand "/usr/local/cs/bin/gitk". This appropiately opened up the gui that was needed, and I was able to answer the questions. The rest of my work is in the screenshot and the gitk-description.txt file.

Project 5:

**Lab**

2) Then unzip using "tar -xvf coreutils-with-bug.tar.gz".

9) I tried reproducing this error in my home directory of the SEASnet linux servers by running the commands seen above in step 7. Even using the ls from /usr/bin/ls, I still see the erronous date that it was created in some time in 2054. This is due to the filesystem of the linux server.

The local file system has a 32 bit signed stamp while the SEASnet NFS filesystem has a 32 bit unsigned stamp. If we instead repeat those steps in number 7 in the SEASNET file system, then we would see the correct output. I tried that in my system's /tmp/ directory. In this case, the

output was correct, the really old file showed the appropiate date rather than some date that is VERY far in the future.

#if ! defined TIMESPEC\_H

# define TIMESPEC\_H

+#include <limits.h>

# include <sys/types.h>

# if TIME\_WITH\_SYS\_TIME

@@ -45,8 +46,15 @@

static inline int

timespec\_cmp (struct timespec a, struct timespec b)

{

- int diff = a.tv\_sec - b.tv\_sec;

- return diff ? diff : a.tv\_nsec - b.tv\_nsec;

+ long longDiff = a.tv\_sec - b.tv\_sec;

+ int currDiff = 0;

+ if (longDiff > INT\_MAX)

+ currDiff = INT\_MAX;

+ else if (longDiff < INT\_MIN)

+ currDiff = INT\_MIN;

+ else

+ currDiff = a.tv\_sec - b.tv\_sec;

+ return currDiff ? currDiff : a.tv\_nsec - b.tv\_nsec;

}

**Homework**

#include <stdio.h>

#include <stdlib.h>

int frobcmp(const void\* chara, const void\* charb) // each \* points to an array of non-space bytes + space

{

char const \* wordOne = \*(char const \*\*) chara; char const \* wordTwo = \*(char const \*\*) charb;

int currPos = 0;

while (wordOne[currPos] != ' ' && wordTwo[currPos] != ' ')

{

int oneChar = (wordOne[currPos])^'\*';

int twoChar = (wordTwo[currPos])^'\*';

if (oneChar - twoChar == 0)

{

++currPos; continue;

}

if (oneChar > twoChar)

return 1;

if (oneChar < twoChar)

return -1;

}

if (wordOne[currPos] == ' ' && wordTwo[currPos] != ' ') // this means that wordOne ran out first, so -1

return -1;

if (wordOne[currPos] != ' ' && wordTwo[currPos] == ' ' ) return 1;

return 0;

}

int main()

{

int allocSize = 8;

int arraySize = 0;

int currLength = 0;

char previousChar = ' ';

char currentChar = ' ';

char \* currWord = (char \*) malloc(allocSize);

if (currWord == NULL)

{

fprintf(stderr, "Array of current words failed to be allocated");

exit(1);

}

int x = getchar(); // getChar will allow us to be able to get the right weird characters

currentChar = (char)x;

while (x != EOF)

{

if (currentChar == ' ' && previousChar == ' ')

{

x = getchar();

currentChar = (char)x;

continue;

}

++currLength;

if (currLength == allocSize) {

currWord = realloc(currWord, (currLength\*2)\*sizeof(char));

if (currWord == NULL)

{

fprintf(stderr, "Memory reallocation failed for currWord");

exit(1);

}

allocSize\*=2;

}

if (currentChar == ' ') {

currWord[currLength-1] = currentChar; arraySize++; }

else {

currWord[currLength - 1] = currentChar;

}

previousChar = currentChar;

x = getchar();

currentChar = (char)x;

}

if (currLength == allocSize) {

currWord = realloc(currWord, (currLength\*2)\*sizeof(char));

if (currWord == NULL)

{

fprintf(stderr, "Memory reallocation failed for currWord");

exit(1);

}

allocSize\*=2;

}

if (previousChar != ' ') {

currWord[currLength] = ' ';

arraySize++;

}

char\*\* strarr = (char\*\*) malloc(arraySize\*sizeof(char\*));

if (strarr == NULL)

{

fprintf(stderr, "Memory allocation for strarr failed");

exit(1);

}

int currWordPos = 0;

char\* traverseWords = currWord;

while (currWordPos < arraySize)

{

strarr[currWordPos] = traverseWords;

if (currWordPos == (arraySize -1))

break;

while (\*traverseWords != ' ')

{

++traverseWords;

}

++traverseWords;

++currWordPos;

}

qsort(strarr, arraySize, sizeof(char\*), frobcmp);

char\* getWord;

int i =0;

for (i = 0; i < arraySize; ++i)

{

getWord = strarr[i];

putchar(\*getWord);

while (\*getWord != ' ') {

getWord++;

putchar(\*getWord);

}

}

if (ferror(stdout)) {

fprintf(stderr, "putchar() failed to output the char!\n");

exit(1);

}

free(strarr); free(currWord);

}

Project 6

**Homework**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

// XOR 42, aka '\*', to compare the decoded with the encoded

unsigned long totalComparisonCount = 0;

int frobcmp(const void\* chara, const void\* charb) {

totalComparisonCount++;

unsigned char const \* wordOne = \*(unsigned char const \*\*) chara;

unsigned char const \* wordTwo = \*(unsigned char const \*\*) charb;

int currPos = 0;

// breaks appropiately)

while (wordOne[currPos] != ' ' && wordTwo[currPos] != ' ')

{

int oneChar = (wordOne[currPos])^'\*';

int twoChar = (wordTwo[currPos])^'\*';

if (oneChar - twoChar == 0)

{

++currPos; continue;

}

if (oneChar > twoChar)

return 1;

if (oneChar < twoChar)

return -1;

}

if (wordOne[currPos] == ' ' && wordTwo[currPos] != ' ')

return -1;

if (wordOne[currPos] != ' ' && wordTwo[currPos] == ' ' ) return 1;

return 0;

}

int main()

{

int allocSize;

struct stat fileStat;

if (fstat(0, &fileStat) < 0)

{

fprintf(stderr, "Unable to run fstat on the current file");

exit(1);

}

if (S\_ISREG(fileStat.st\_mode))

allocSize = fileStat.st\_size+1; else

allocSize = 1\*sizeof(unsigned char \*);

int arraySize = 0;

int currLength = 0;

unsigned char previousChar = ' ';

unsigned char currentChar = ' ';

unsigned char \* currWord = (char \*) malloc(allocSize \* sizeof(unsigned char)); // pointer to the current word

if (currWord == NULL)

{

fprintf(stderr, "Array of current words failed to be allocated");

exit(1);

}

while (read(0, &currentChar, sizeof(unsigned char)) > 0)

{

if (currentChar == ' ' && previousChar == ' ')

{

continue;

}

++currLength;

if (currLength == allocSize) {

currWord = realloc(currWord, (currLength\*2)\*sizeof(unsigned char));

if (currWord == NULL)

{

fprintf(stderr, "Memory reallocation failed for currWord");

exit(1);

}

allocSize\*=2;

}

if (currentChar == ' ') {

currWord[currLength-1] = currentChar; arraySize++; }

else {

currWord[currLength - 1] = currentChar;

}

previousChar = currentChar;

}

if (currLength == allocSize) {

currWord = realloc(currWord, (currLength\*2)\*sizeof(unsigned char));

if (currWord == NULL)

{

fprintf(stderr, "Memory reallocation failed for currWord");

exit(1);

}

allocSize\*=2;

}

if (previousChar != ' ') {

currWord[currLength] = ' ';

arraySize++;

}

unsigned char\*\* strarr = (unsigned char\*\*) malloc(arraySize\*sizeof(unsigned char\*));

if (strarr == NULL)

{

fprintf(stderr, "Memory allocation for strarr failed");

exit(1);

}

int currWordPos = 0;

unsigned char\* traverseWords = currWord;

while (currWordPos < arraySize)

{

strarr[currWordPos] = traverseWords;

if (currWordPos == (arraySize -1))

break;

while (\*traverseWords != ' ')

{

++traverseWords;

}

++traverseWords;

++currWordPos;

}

qsort(strarr, arraySize, sizeof(unsigned char\*), frobcmp);

unsigned char\* getWord;

int i =0;

for (i = 0; i < arraySize; ++i)

{

getWord = strarr[i];

write(1, getWord, sizeof(unsigned char));

//putchar(\*getWord);

while (\*getWord != ' ') {

getWord++;

write(1, getWord, sizeof(unsigned char));

}

}

if (ferror(stdout)) {

fprintf(stderr, "putchar() failed to output the char!\n");

exit(1);

}

free(strarr); free(currWord);

fprintf(stderr, "\nComparisons: %lu\n", totalComparisonCount);

return 0;

}

Lab 7:

Lab:

#include <pthread.h>

int nthreads;

scene\_t scene;

float scaled\_color[width][height][3]; void\* routine(void \* startPos)

{

int startingPosition = \*(int\*) startPos;

Vec3 camera\_pos;

set( camera\_pos, 0., 0., -4. );

Vec3 camera\_dir;

set( camera\_dir, 0., 0., 1. );

const double camera\_fov = 75.0 \* (PI/180.0);

Vec3 bg\_color;

set( bg\_color, 0.8, 0.8, 1 );

const double pixel\_dx = tan( 0.5\*camera\_fov ) / ((double)width\*0.5);

const double pixel\_dy = tan( 0.5\*camera\_fov ) / ((double)height\*0.5);

const double subsample\_dx

= halfSamples ? pixel\_dx / ((double)halfSamples\*2.0)

: pixel\_dx;

const double subsample\_dy

= halfSamples ? pixel\_dy / ((double)halfSamples\*2.0)

: pixel\_dy;

/\* for every pixel \*/

for( int px=startingPosition; px<width; px+=nthreads)

{

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 \*/

scaled\_color[px][py][0] = gamma( pixel\_color[0] ) \* max\_color;

scaled\_color[px][py][1] = gamma( pixel\_color[1] ) \* max\_color;

scaled\_color[px][py][2] = gamma( pixel\_color[2] ) \* max\_color;

/\* enforce caps, replace with real gamma \*/

for( int i=0; i<3; i++)

scaled\_color[px][py][i] = max( min(scaled\_color[px][py][i], 255), 0);

}

}

return NULL;

}

int main( int argc, char \*\*argv )

{

nthreads = argc == 2 ? atoi( argv[1] ) : 0;

if( nthreads < 1 )

{

fprintf( stderr, "%s: usage: %s NTHREADS\n", argv[0], argv[0] );

return 1;

}

scene = create\_sphereflake\_scene( sphereflake\_recursion );

pthread\_t threads[nthreads]; // make an array with the size of the number of threads that we went

int startPos[nthreads]; // store the ints here for the start pos because otherwise there is a race condition!! (learned that the hard way =( )

for (int i = 0; i < nthreads; i++){

startPos[i] = i;

int createThread = pthread\_create(&threads[i], NULL, routine, (void\*)(startPos + i));

if (createThread)

fprintf(stderr, "Could NOT create a new thread and an error was thrown");

}

// now that you have the threads doing their tasks ... if you dont join them here, then one thread might finish

for (int i = 0; i < nthreads; i++){

pthread\_join(threads[i], NULL);

}

for (int px = 0; px < width; px++){

for (int py = 0; py < height; py++){

/\* 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", scaled\_color[px][py][0], scaled\_color[px][py][1], scaled\_color[px][py][2]);

}

printf("\n");

}

free\_scene( &scene );

if( ferror( stdout ) || fclose( stdout ) != 0 )

{

fprintf( stderr, "Output error\n" );

return 1;

}

return 0;

}

Lab 8:

**Lab:**

Run with ldd ./exec so that you can see which dynamically linked libraries are being called

Homework:

#include "randlib.h"

#include <immintrin.h>

/\* Initialize the hardware rand64 implementation. \*/

void rand64\_init (void)

{

}

/\* Return a random value, using hardware operations. \*/

extern unsigned long long rand64 (void)

{

unsigned long long int x;

while (! \_rdrand64\_step (&x))

continue;

return x;

}

/\* Finalize the hardware rand64 implementation. \*/

void rand64\_fini (void)

{

}

#include "randlib.h"

#include <stdio.h>

#include <stdlib.h>

/\* Software implementation. \*/

/\* Input stream containing random bytes. \*/

static FILE \*urandstream;

/\* Initialize the software rand64 implementation. \*/

\_\_attribute\_\_ ((\_\_constructor\_\_))

static void rand64\_init (void)

{

urandstream = fopen ("/dev/urandom", "r");

if (! urandstream)

abort ();

}

/\* Return a random value, using software operations. \*/

extern unsigned long long rand64 (void)

{

unsigned long long int x;

if (fread (&x, sizeof x, 1, urandstream) != 1)

abort ();

return x;

}

/\* Finalize the software rand64 implementation. \*/

\_\_attribute\_\_ ((\_\_destructor\_\_))

static void rand64\_fini (void)

{

fclose (urandstream);

}

Main:

/\* 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]);

exit(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 (\*initialize) (void);

unsigned long long (\*rand64) (void);

void (\*finalize) (void);

void\* dl\_handle;

char\* error;

if (rdrand\_supported ())

{

dl\_handle = dlopen("randlibhw.so", RTLD\_LAZY);

if (!dl\_handle){

fprintf(stderr, "dlopen() Error with randlibhw.so %s\n", dlerror());

exit(1);

}

initialize = dlsym (dl\_handle, "rand64\_init");

error = dlerror();

if (error!= NULL)

{

fprintf(stderr, "Error with opening rand64\_init for hardware %s\n", error);

exit(1);

}

rand64 = dlsym (dl\_handle, "rand64");

error = dlerror();

if (error!= NULL)

{

fprintf(stderr, "Error with opening rand64 for hardware %s\n", error);

exit(1);

}

finalize = dlsym (dl\_handle, "rand64\_fini");

error = dlerror();

if (error!= NULL)

{

fprintf(stderr, "Error with opening rand64\_fini for hardware %s\n", error);

exit(1);

}

//finalize = hardware\_rand64\_fini;

fprintf(stdout, "HW\n");

}

else

{

dl\_handle = dlopen("randlibsw.so", RTLD\_LAZY);

if (!dl\_handle){

fprintf(stderr, "dlopen() Error with randlibhw.so %s\n", dlerror());

exit(1);

}

rand64 = dlsym (dl\_handle, "rand64");

error = dlerror();

if (error!= NULL)

{

fprintf(stderr, "Error with opening rand64 for software %s\n", error);

exit(1);

}

fprintf(stdout, "SW\n");

}

int wordsize = sizeof rand64 ();

int output\_errno = 0;

int ceiling = nbytes%wordsize;

if (ceiling != 0)

nbytes = nbytes + (wordsize - ceiling);

do

{

unsigned long long x = rand64 ();

fprintf(stdout, "%llu\n", x);

nbytes -= wordsize;

}

while (0 < nbytes);

if (fclose (stdout) != 0)

output\_errno = errno;

if (output\_errno)

{

errno = output\_errno;

perror ("output");

finalize ();

exit(1);

}

dlclose(dl\_handle);

error = dlerror();

if (error != NULL)

{

fprintf(stderr, "Error closing the dl handle%s\n", error);

exit(1);

}

exit(0);

}

randlibsw.so: randlibsw.c

$(CC) $(CFLAGS) randlibsw.c -shared -fPIC -o randlibsw.so

randlibhw.so: randlibhw.c

$(CC) $(CFLAGS) randlibhw.c -shared -fPIC -o randlibhw.so

randmain: randcpuid.o randmain.o

$(CC) $(CFLAGS) -ldl -Wl,-rpath=$(PWD) -o randmain randcpuid.o randmain.o

Lab 9:

Step 1: Installing open SHH

I first had to install openssh and did the following commands to get ssh and get the server/client:

dpkg --get-selections | grep openssh

sudo apt-get install openssh-server

sudo apt-get install openssh-client

Step 2: I had to generate the ssh keys for my client, and did so by typing:

ssh-keygen // The keys are saved into ∼/.ssh/is\_rsa and is saved in the format id\_rsa.pub)

Step 3: I tried to find my IP by runing ipconfig, and then tried to ping my partner's IP address to check for validity.

ipconfig // my IP was 10.97.85.106, his was 10.97.85.101

ping 10.97.85.101

Step 4: While my partner waited, I began to create the user on my machine so that my partner could ssh into that:

sudo useradd -d /home/kunjan -m kunjan1

cd /home/kunjan1/ // verified that kunjan1 exists

sudo passwd kunjan1 // created a password for kunjan1

cd /home/kunjan1/ // go to the user I just created

sudo mkdir .ssh // created the ssh directory and changed the ownership + permission

sudo chown -R kunjan1 .ssh // recursively change ownership to kunjan1

sudo chmod 700 .ssh // 700 = read, write, execute

Step 4: After my partner attempted to ssh, I disabled the password based authentication by running:

emacs /etc/ssh/sshd\_config

And then changed the line "PasswordAuthentication -> yes" to being no. This way he could log in without using his

password next time.

Step 5: Now after my partner did step 1-3 for me, I began doing some steps so I could log onto his machine with my

username:

ssh-copy-id -i sahil1@10.97.85.101 // copied public key over the server

ssh-add // added the private key to ssh-agent

Step 6: Now I can log on to his machine:

ssh sahil1@10.97.85.101

I ran a few commands like "man chmod", and "mkdir SahilTest", and more to see that I was indeed able to log on to his

computer and the changes stuck. He tried going to my user, and saw that the changes were there, so we knew it was

successful!

I then took out Kunjan's read, write, and execute permissions for my user on his computer by running:

chmod o-rwwx "$HOME"

Now that user was only for me!! Additionally, I also tried running it with x-forwarding, by typing the following

command, and also ran "firefox" to see that indeed, the firefox window popped up!:

ssh -X sahil1@10.97.85.101

firefox

Sahil Gandhi

11/22/2016

TA: Thuy Vu

Lab: Monday/Wednesday 4-6 PM

1) If while we were creating the lab portion the other team could see all of the bytes accross the networks, then

the resulting network IS still secure. This is true because the private key is still in our control, not in the

control of the other team. All they have access to is the bytes, and in their encrypted format, they have no idea what

is happenening.

A) If they had access to our keyboard strokes then it would not be safe anymore since they would be able to know which

words we typed for say the username or password or to open certain programs, and then could use those known keys to

generate a decrypter for the rest of the characters.

B) If instead they had access to the physical control of the USB (we are booting off of the USB), then the network is

NO LONGER secure either ... they can get their hands on the encryption key and then figure out everything that we have

typed. Perhaps if the key has a password associated with it, then it might be harder to get access, but even then ...

they could copy the key, and try cracking the password in their own time and then come back to decrypt everything you

have been typing.

2) gpg --verify does not really verify WHO created the tar file in question, but rather if the signature was changed

recently (meaning someone else changed the tar file and their signature was added (hence the change)) ie ... it can

only verify document integrity, not document origin. This means that if I intercepted my friends program and using his

key, created some malicious code and sent it out to the public ... if the public tries to use gpg --verify, it will

say that my friend created those files and is the "hacker". However, in reality I was the one who created the

malicious code, and gpg --verify can't do anything about this.

Possible solutions to this problem:

a) Directly talk to possible customers/users of the code so that they can verify that it is you. This is fine if you

are not distributing the code to many people, but for massively distributed programs, this would not be the best way

to solve the gpg --verify issue.

b) Another way to solve this problem is if you have some kind of trusted third party that can generate a certificate

that can verify the identity of the sender based on some previously issued digital certificate. This digital

certificate would tell who the real owner of the code is, and if that differs from the signer, then we know there is

some kind of an error/mishap/trickery going on. One modern example of that is the Certificate Authority (CA) which

includes its own data on the sender so that it can verify the code being sent is from the correct individual/

organization.