CS 460 Programming Assignment One Linux Tools, System Calls

Due Feb 13, 2018, 11:59pm Email me your SVN/Git repository name
Due Feb 15, 2018, **1:00 pm** Final Project (in class demo!) **50 points**

Goal:
Refresh your skills on: Eclipse, Subversion, the Linux command line, C, Makefiles, argv/argc, and learn about Linux System Calls and some Linux command line tools. We also want to understand how processes are created in Linux and how executable files are formatted.

Description:
You will write a program that will make a number of system calls to determine information about the system your program is running on. You'll need to display this information to the screen as specified below. Further, and most important, you will need to run your program using **`strace`** and **`ltrace`** to investigate how Linux is executing your program. You will need to submit answers to the questions at the end of this handout. The answers must be written in full English sentences in a text file named **Answers.txt** in the root of your Eclipse project.

Your Program:

The system calls you will need to invoke are:

nanosleep()	access()
sysinfo()	uname()

You will also need to use errno/perror() to find and display error messages produced by the system calls. You will need to determine which header files supply each of the above functions.

Your program must do the following, in the following order:

1. You must write and call a function named printInt(). This function must take exactly one parameter of type integer. The function must only print the parameter to the screen. Call the function with the value 17. This must be the first executable line in main().
2. Dynamically allocate one byte. Print the address (in hexadecimal) of that byte.
3. Pause for 1.5 seconds using nanosleep
4. Call **uname** and display all of the returned information to the screen using printf.
5. You must check to make sure exactly one file name is passed as a command line argument. If this is not the case you need to print a usage message and terminate.
6. Call **access** to determine if you have both **read** and **write** access to the file passed as a command line argument. Use **perror**() to print an error if you don't have access.
7. Call **sysinfo** and display all of the returned information to the screen (separated by \n). *You can ignore the load information.*
8. Print the address of: main(), cos() [from math.h], and printInt().

   Error handling: For nanosleep, check for errors and then decode the error using errno.
   For all other system calls, check for errors and then use perror() to display an error message.
   NOTE: Using perror writes to stderr so your error output may not be in the order you expect or the order on this handout!

Sample Output:

```
chadd@coffee:~> ./CS460_SysCalls /etc/passwd
17
HeapChar: 1b9a010

nanosleep ------------------------------------

UNAME ------------------------------------
Linux | coffee | 4.4.104-39-default | #1 SMP Thu Jan 4 08:11:03 UTC 2018 (7db1912) |
x86_64 | (none)

access Read and Write  ------------------------------------
/etc/passwd with R_OK | W_OK: Permission denied

sysinfo  ------------------------------------

uptime: 2067971
Total Ram: 16769466368
Free Ram: 1796116480
Shared Ram: 359645184
Buffer Ram: 956989440
Total Swap: 8585736192
Free Swap: 8153493504
Processes: 845
Total High Memory: 0
Free High Memory: 0
Memory Unit Size: 1


Math cos address: 400790
Main address: 4009b3
printInt address: 4008fd
```

Usage Output:

```
coffee$ ./CS460_SystemCalls
USAGE: ./CS460_SysCalls filename
```

NOTE: Your numbers/data will vary!

Subversion/Git (or how do I submit my work?):

You must store your source code in a Subversion repository on zeus or a private git repository on GitHub or GitLab.

Subversion - one repository for the entire semester

You need to create an empty, private repository as follows:

```
zeus$ svnadmin create /home/login/SVNREPOS_CS460S18/
```

You can connect to this through Eclipse using the address:

svn+ssh://***login***@zeus.cs.pacificu.edu/home/***login***/SVNREPOS_CS460S18/

Git - one repository per project

Create an empty, private repository on GitHub/GitLab named CS460S18_SysCalls.

Create an Eclipse project as described in the attached sheet and use SSH Keys to connect to the GitLab/GitHub repository.

Name your Eclipse project **CS460_SysCalls_PUNetID**. I will check out your code using the following command:

```
zeus$ svn co -r {2018-02-15T13:05} svn+ssh://zeus.cs.pacificu.edu/home/PUNetID/SVNREPOS_CS460S18/
CS460_SysCalls_PUNetID
```

This will pull out the last revision made previous to 1:05pm on Feb 15, 2018.

Makefile

You will need to build a Makefile for this Eclipse project. You need the following make targets:

```
CS460_SysCalls: build the executable file CS460_SysCalls (using -g)
runTest:        run './CS460_SysCalls /etc/passwd'
runStrace:      run 'strace ./CS460_SysCalls /etc/passwd > strace.out
2>strace.err'
runLtrace:      run 'ltrace ./CS460_SysCalls /etc/passwd > lstrace.out
2>ltrace.err'
runTime:        run 'time ./CS460_SysCalls /etc/passwd'
clean:          remove any executable and object files
```

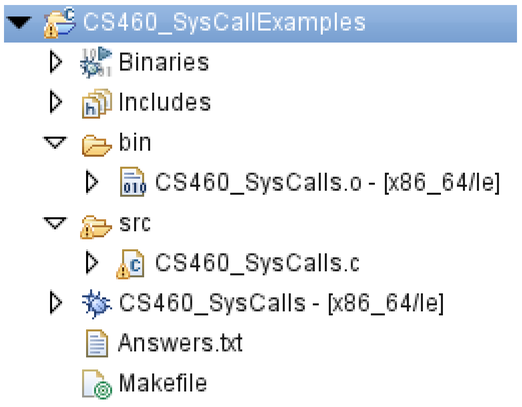
I will compile your code (on Zeus) using the command:

```
zeus$ gmake clean ; gmake
```

I will test your code using the command (among others):

```
zeus$ gmake runTest
```

Eclipse



Your Eclipse project must look very similar to the project shown here. Note the executable file and Answers.txt are at the root of the project. You may add an **include** directory. You must adhere to proper Computer Science department coding standards!

Useful resources

http://linux.die.net/man   snprintf   fflush(stdout)

Questions:

Answer the following questions in the file Answers.txt in your Eclipse Project. Use proper English, complete sentences, and sensible formatting. Put your name at the top of the file.

1. How many hours did you spend coding this project?

2. Which part of this project caused you the most difficulty?

3. Given the output of runTime, does nanosleep seem accurate?

4. Study the output of **`ltrace`**.
   1. Does the call to **main** show up? If so where, if not, why not?
   2. Does the call to the **printInt** show up? If so where, if not, why not?
   3. What does **access("/etc/passwd", 6) = -1** mean? Where does the **6** come from? What does the **-1** mean?
   4. Do you see calls to **puts**? Did you call **puts**? Why do you think **puts** is there?
   5. Why does **printf** show up in addition to **puts**?

5. Study the output of **`strace`**.
   1. Why is Linux trying to open libc.so? Where is libc.so found?
   2. Which file descriptor is assigned to libc.so?
   3. Where is that file descriptor being used? When does libc.so get closed?
   4. What do the parameters to mmap do?
   5. What are execve() and brk(0) doing?
   6. What does brk(0xSomeHexValue) do?
   7. How does the value in write(1, "HeapChar:….") related to the hex values returned by brk(0) above and the call to brk() in question 6?
   8. Where does `write(2, "/etc......"` come from? Where does the 2 come from?
   9. Where does write(1, “access.....” come from? Where does the 1 come from? How many times does write(1 appear? Does this make sense?
   10. How to the addresses of main, cos, and printInt differ?

   The answers to these questions are the most important part of this assignment!

This took me less than 250 non-comment lines of code. The most time consuming part of this assignment is the Questions section and the **research** you will need to do.