CS 460 Strace/Ltrace lab script

```
./CS460_SysCalls /etc/passwd
./CS460_SysCalls /etc/passwd


file ./CS460_SysCalls
ldd ./CS460_SysCalls

# turn off address randomization   Addr Space Layout Randomization
# not permanent
echo 0 | sudo tee /proc/sys/kernel/randomize_va_space

# turn it back on
echo 2 | sudo tee /proc/sys/kernel/randomize_va_space

make runStrace
make runLtrace

strace -e raw=mmap ./CS460_SysCalls /etc/passwd > strace.out 2>
strace_raw.err
```

Study the output of ltrace.

2. 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**?

3. 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?

Process Layout in Memory

ELF File

http://www.cs.stevens.edu/~jschauma/810/elf.html

Open debugger

```
 gdb ./CS460_SysCalls

(gdb) break main
(gdb) break 76 # line the prints HeapChar
(gdb) break 186 # line number containing return from main!

(gdb) run /etc/passwd # should pause at return

(gdb) set disassemble-next-line on
(gdb) info reg
(gdb) info stack
(gdb) info frame

(gdb) stepi # until you hit callq printInt

# what is happening previous to callq?
(gdb) stepi # in printInt
(gdb) disas $pc

# what is happening with %edi
# %rbp
# %rsp

(gdb) stepi # until printf@plt # plt: procedure linking table
(gdb) stepi # a few times to _dl_runtime_….
(gdb) break printf
(gdb) cont
(gdb) disas printf
(gdb) info reg # $rsi
(gdb) x $rbp
(gdb) x $rbp-4
(gdb) stepi # until mov %rsi, 0x28(%rsp)
(gdb) x 0x28 + ($rsp)
(gdb) cont # goes to main.c:76
(gdb) stepi # until printf
(gdb) info break
(gdb) del 4 # delete break point 4 (printf)
(gdb) cont # got to return statement
(gdb) print $rbp
(gdb) print $rsp
```

New Terminal

find PID
ps -ef | grep CS460

cat /proc/PID/maps

pmap PID

We also see the [heap] Does that address make sense?

readelf -a CS460_SysCalls | less

readelf -d CS460_SysCalls # .dynamic section

readelf -l CS460_SysCalls # segments

readelf -S CS460_SysCalls # sections

memory page:

getconf PAGESIZE