CmpE102
Spring 2017
Programming Assignment 1

Due Date: 20 February
Goal: "Hello World" program in NASM

Instructions:

Step 1. Getting access to the Linux Lab

This class will use the Linux lab in E206. However, we will not be going to E206, we will be accessing remorely. If you prefer, you may use your own personal Linux installation.

In order to access the Linux systems, you will need a SSH client. The recommended one is `PuTTY`, obtainable from http://www.chiark.greenend.org.uk/~sgtatham/putty/. Or you can use `ssh`, if you are familiar with it.

You may also find it useful to use FTP. The recommended application is found at http://winscp.net/eng/download.php. Click on *Standalone application*.

If you use `PuTTY` to access the lab, you will have to use `vi` as your text editor. Knowing how to use `vi` is a useful skill which you should definitely learn at some point.

I believe that the italicized information is incorrect:

However, should you wish to postpone this valuable learning experience until a later time, you need a GUI in order to make use of `gedit`, the editor you don't need to learn how to use, or `emacs`, if you have an interest in that.

Go to http://www.tightvnc.com/. Download and install `TightVNC` viewer. (This is for Windows. For Mac, you might try Chicken of the VNC.)

Follow the instructions given in a separate document to connect to the lab.

Keep in mind that this requires you to be working from a terminal window. If you are using **`PuTTY`** that is what happens by default. If you are using `TightVNC`, right-click anywhere in the window and select "Open Terminal".

NOTE: Students in my other class have reported being able to connect using `PuTTY.` If you get the message "Access denied", it means that too many students in E206 have logged into the system and then walked away from the machine without logging out. Try another system.

Step 1. Using your own Linux system.

If the Linux lab cannot be reached, use Linux installed on your own system, as described in another document.

Step 2. Creating the assembly language file

Now use an editor to create a file named `eatsyscall.asm` containing the following text. (In `TightVNC`, select `Applications-►Accessories-►Text Editor`)

You must enter the text *exactly* as shown. Here is the text to enter:

```
;  Executable name : EATSYSCALL

;  Version         : 1.0
;  Created date    : 1/7/2009
;  Last update     : 2/18/2009
;  Author          : Jeff Duntemann
;  Description     : A simple program in assembly for Linux, using NASM
;    demonstrating the use of Linux INT 80H syscalls to display text.
;
;  Build using these commands:
;    nasm -f elf -g -F stabs eatsyscall.asm
;    ld -o eatsyscall eatsyscall.o
;

SECTION .data         ; Section containing initialized data
     EatMsg: db "Eat at Joe's!",10
     EatLen: equ $-EatMsg

SECTION .bss          ; Section containing uninitialized data

SECTION .text         ; Section containing code

global _start         ; Linker needs this to find the entry point!

_start:
     nop              ; This no-op keeps gdb happy...
     mov eax,4        ; Specify sys_write call
     mov ebx,1        ; Specify File Descriptor 1: Standard Output
     mov ecx,EatMsg   ; Pass offset of the message
     mov edx,EatLen   ; Pass the length of the message
     int 80H          ; Make kernel call

     mov eax,1        ; Code for Exit Syscall
     mov ebx,0        ; Return a code of zero
     int 80H          ; Make kernel call
```

Be careful to distinguish between zero and upper-case O, and one and lower-case L. In the font used above, they are not the same.

Step 3. Assembling and running the file

If you do not already have a terminal window open, open one. Make sure you are in the same directory as your file.

The first step of building an executable is assembling an object file from the source:

```
$ nasm -f elf -l eatsyscall.lst eatsyscall.asm
```

Note: The `$` is my prompt. You may have a different prompt.

This creates two files: object file `eatsyscall.o` and listing file `eatsyscall.lst`. The listing file shows how the assembly language is transformed into machine code.

The second step is producing the executable file itself from the object file by invoking the linker. For a 32-bit environment the command is:

```
$ ld -o eatsyscall eatsyscall.o
```

For a 64-bit environment the command is:

```
$ ld -o eatsyscall eatsyscall.o -melf_i386
```

This will build `eatsyscall` executable. You can now run the executable by typing its name. If not on your path, precede the name with `/.`

Submission Instructions

Bring your laptop to the class meeting on Feb. 20 and be ready to demonstrate the program.

References

Konstantin Boldyshev, ***Linux Assembly HOWTO***, http://www.faqs.org/docs/Linux-HOWTO/Assembly-HOWTO.html#AEN853, version of 2002/08/17 08:35:59

Duntemann, pg. 142