EECS 338 Homework 2 *NOTE: Problem 2 was modified slightly on Saturday, January 27. Changes in red.*

General requirements:

- Due at 11 PM on the posted due date.
- Create a typed report for discussion questions.
- Upload a single, compressed file (e.g. zip) to Canvas that contains all required files (programs + report).
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.
- ***NEW: Include either a single makefile that compiles, all programs or a separate makefile for each.***

Special requirements:

- Students must use a Linux OS and the commands that are specified in the instructions. Mac users are welcome to experiment with Mac equivalents, but the report should show results using Linux, in order to be compatible with the instructions.
- For screen output, copy/paste actual text from the terminal window. Do not use screenshots.

Problems:

1. Run each of the following programs using the "time" command to analyze the wall (real) time, user mode time, and kernel (system) mode time. The command syntax is "time ./*program*", where *program* is the name of the program. **For each program, provide the output in your report and briefly explain why the "user" and "sys" times occurred.**

   a. Use "for.c" from HW #1. If desired, you may reduce the value of "N" for a more convenient (shorter) run time. Having a long run time is not important. An explanation for the times was discussed in class.
   b. Revise "for.c" such that "user" time is less than 50% of the "sys" time. Hint: try adding "printf" in a strategic location. You can adjust the value of "N" to be convenient.
   c. Revise "for.c" such that "user" time and "sys" time are each less than 2% of the "real" time. Hint: try adding "sleep". You can adjust the value of "N" to be convenient.

2. Create any C program you wish that prints the addresses of two variables: an integer and a *constant* integer. Then run "pmap" to visualize the memory usage, and answer the questions that follow. **Include a program output in your report.** The following example uses the %p format specifier to print a pointer value (address) and the "address of" operator (&) to obtain the pointer:

```
int a = 0; // Regular integer
const int b = 0; // Constant integer
// Print the pointer values (addresses) in hexadecimal
printf("%p, %p\n", &a, &b);
```

   Hypothetical output:

```
0x7ffd90ed9038, 0x7ffd90ed903c
```

   a. Run "top" and find your process in the list. How well does the total virtual memory (VIRT) reported by "top" compare to the amount displayed by "pmap"? Are they the same or different? **In your report, include the relevant output from "pmap" and "top", and describe the similarity or difference.** Note: some Linux computers may show the same values, and some may show different values. You are not required to explain any differences.

   b. In your report, include the output that shows the addresses of two variables. What is the name of the memory segment(s) (far right column) in which those variables are located? What are the permissions for that segment? Do the permissions agree with the concept of a "constant" (read-only)?

c. The following table lists hypothetical combinations of memory permissions. Which ones appear in the "pmap" output for your program? Briefly explain why they would appear or why they would not.

Permissions	*Appears? (yes or no)*	*Why or why not?*
`r--`		
`rw-`		
`r-x`		
`rwx`		
`--x`		

d. Suppose "pmap" produced the output line shown below for a memory segment. What is the maximum hexadecimal address for that memory segment? What is the maximum possible hexadecimal starting address for a 4-byte integer in that memory segment?

```
00007f7c01248000      8K rw---  [ anon ]
```

3. Use "lsmod" to list the kernel modules that are loaded, and use "dmesg" to see kernel messages. If desired, you can redirect the results to files. For example, you might use "lsmod > lsmod.txt" and "dmesg > dmesg.txt" so that you can use a text editor for searching. Then do the following:

a. In the text produced by "dmesg", find one message that you believe came from a kernel module that appeared in the "lsmod" list. Use "modinfo" to provide evidence that the output from "dmesg" is actually from that module. **In your report, show the following: (i) the line from "lsmod" that shows the module name, (ii) the corresponding message from "dmesg", (iii) the relevant output from "modinfo", and (iv) a brief explanation of why you believe that module produced that message.** Below is an example (you cannot use this one!):

From lsmod:
```
vmwgfx                233472  2
```

From dmesg:
```
[    9.100729] [drm] Initialized vmwgfx 2.9.0 20150810 for
0000:00:0f.0 on minor 0
```

From "`modinfo vmwgfx`"*:*
```
filename:       /lib/modules/4.4.0-109-
generic/kernel/drivers/gpu/drm/vmwgfx/vmwgfx.ko
description:    Standalone drm driver for the VMware SVGA device
```

Explanation: The message and the module info both refer to "drm" and "vmwgfx".

b. Choose one kernel module from the list and do an Internet search to find the purpose of that particular module. **In your report, provide the following: (i) a brief description of the purpose of the module, (ii) a URL for the information you found, and (iii) a brief explanation for how you know the information is related to that module.**

Rubric:

Item	**Points**
1(a) Program, output, and explanation	10

1(b) Program, output, and explanation	10
1(c) Program, output, and explanation	10
2(a) Output and explanation	10
2(b) Program output and memory name	5
2(b) Permissions and explanation	10
2(c) Appearance and explanation	10
2(d) Segment address and integer address	10
3(a) Output from lsmod, dmesg, and modinfo	10
3(a) Explanation	5
3(b) Description and URL	5
3(b) Explanation	5
Total	100