Brennan McFarland
January 30, 2018
Homework 2

1a. Reducing N to 500, the output of time (omitting the “Looping” line from for.c itself) was:
real 0m0.397s
user 0m0.396s
sys 0m0.000s
All of the time was apparently spent in user mode, meaning little to no time (a fraction of a millisecond could have been truncated) was spent in kernel/system mode. This is expected as for.c simply runs a loop and makes no system calls in its code.

1b. Reducing N to 50 and adding a printf(“this is text printed that is causing the program to enter kernel mode”) inside the innermost loop produces that same phrase repeated many times over (omitted in the following for conciseness) and the following output for time:
real 0m0.953s
user 0m0.024s
sys 0m0.052s
The difference from 1a is that printf is a system call, which causes the process to enter kernel mode. In other words, it references the system library containing the printf command which has unrestricted access to memory. Due to the length of the text to print (I tested and a longer length leads to longer time in sys mode) and the comparatively smaller overhead of iterating through the loops, this causes the process to spend more than two-thirds of its time in kernel mode.

1c. Reducing N to 2 and replacing the printf with a sleep(1) command (and a preceding fflush(stdout) to get rid of buffered user input) yields (with the “Looping” message omitted):
real 0m8.007s
user 0m0.000s
sys 0m0.000s
Now the process spends most of its time executing the sleep command, causing it to be dormant and neither taking up user or system time, which have both been rounded to 0ms in the displayed output.

2a. The line corresponding to the process displayed by top is:
6433 user 20 0 4344 636 568 S 0.0 0.0 0:00.00 printaddresses.
And pmap:
0000562279372000 4K r-x-- printaddresses.o
0000562279572000 4K r---- printaddresses.o
0000562279573000 4K rw--- printaddresses.o

000056227a0ba000 132K rw--- [anon]
00007ff54bd1a000 1784K r-x-- libc-2.24.so
00007ff54bed8000 2044K ----- libc-2.24.so
00007ff54c0d7000 16K r---- libc-2.24.so
00007ff54c0db000 8K rw--- libc-2.24.so
00007ff54c0dd000 16K rw--- [anon]
00007ff54c0e1000 152K r-x-- ld-2.24.so
00007ff54c2e7000 8K rw--- [anon]
00007ff54c303000 12K rw--- [anon]
00007ff54c306000 4K r---- ld-2.24.so
00007ff54c307000 4K rw--- ld-2.24.so
00007ff54c308000 4K rw--- [anon]
00007fffd783c000 132K rw--- [stack]
00007fffd7963000 8K r---- [anon]
00007fffd7965000 8K r-x-- [anon]
ffffffffff600000 4K r-x-- [anon]
total 4348K

Though slightly different at 4344K from top and 4348K from pmap, the virtual memory values are nearly identical, as would be expected as they are both measurements of the process memory usage.

2b. The program itself outputs:
integer: 0x7fffd785be20 constant: 0x7fffd785be24
Since the variables' addresses are greater than 00007fffd783c000 but less than 00007fffd7963000, they would both appear to be in the block
00007fffd783c000 132K rw--- [stack]
Named [stack], which has read and write permissions. This does not seem to agree with the concept of a concept being read-only, but could be justified by the fact that the name indicates that this is the program stack, which the program has full access to read and write even if it will not do so because of the language.

2c.

Permissions	Appears?	Why
r--	yes	The program uses some read-only memory, likely memory addresses or system constants (though not the

		constant we declared, see above)
rw-	yes	The integer variable needs to be able to be read from and written to
r-x	yes	This is probably system calls like the printf command, which can be read from and executed but not written to as it is a system library
rwx	no	The program only uses data memory that can be read/written and instructions that can be read/executed
--x	no	Apparently whatever the program can execute it can also read, so long as it doesn't modify anything

2d. 00007f7c01248000 + 8K =

```
    00007f7c01248000
  + 0000000000002000
    00007f7c0124a000
```

So the maximum address in that memory segment is `00007f7c01249fff`

`00007f7c0124a000-4 = 00007f7c01249ffc`

So the maximum possible hexadecimal starting address for a 4-byte integer in that segment is `00007f7c01249ffc`

3a.

Lsmod:

pinctrl_intel 20480 1 pinctrl_sunrisepoint

Dmesg:

[0.106213] pinctrl core: initialized pinctrl subsystem

Modinfo:

filename: /lib/modules/4.10.0-42-generic/kernel/drivers/pinctrl/intel/pinctrl-intel.ko

license: GPL v2

description:	**Intel pinctrl/GPIO core driver**
author:	Mika Westerberg <mika.westerberg@linux.intel.com>
author:	Mathias Nyman <mathias.nyman@linux.intel.com>
srcversion:	0570A25AD0B11B73AC76412
depends:	
intree:	Y
vermagic:	4.10.0-42-generic SMP mod_unload

Explanation: The pinctrl_intel appears to be a driver and a part of a GPIO pin control subsystem which the message from dmesg was announcing to have initialized.

3b. Intel_pch_thermal
https://cateee.net/lkddb/web-lkddb/INTEL_PCH_THERMAL.html
This module enables support for thermal reporting on some intel PCHs (Platform Controller Hub, the motherboard chipset), providing hardware temperature readings and allowing for things like automatic shutdown when the system overheats.
This information is directly taken from the Linux Kernel Driver Database's section on the intel_pch_thermal driver.