Name: ID#: X.500:

CSCI 2021: Practice Final Exam

Fall 2022

University of Minnesota

Exam period: 20 minutes
Points available: 40

Background: Nearby are several C files along with two attempts to compile them on the left. Study these and answer the questions that follow.

```
> gcc vf_weak_var.c vf_strong_func.c vf_main.c   # COMPILE 1
/usr/bin/ld: warning: size of symbol ‘foo’ changed from 4 to 14
/usr/bin/ld: warning: type of symbol ‘foo’ changed from 1 to 2
> file a.out
a.out: ELF 64-bit LSB pie executable, x86-64, version
> ./a.out
-573193927
4
> rm a.out

> gcc vf_strong_var.c vf_strong_func.c vf_main.c # COMPILE 2
/usr/bin/ld: multiple definition of ‘foo’;
collect2: error: ld returned 1 exit status
> file a.out
a.out: cannot open ‘a.out’ (No such file or directory)
```

```
// FILE: vf_main.c
#include <stdio.h>
int foo(int x);
int main(){
  printf("%d\n",foo);
  printf("%d\n",foo(2));
  return 0;
}

// FILE: vf_strong_func.c
int foo(int x){
  return 2*x;
}

// FILE: vf_strong_var.c
int foo = 0;

// FILE: vf_weak_var.c
int foo;
```

Problem 1 (10 pts): Why does `COMPILE 1` succeed while `COMPILE 2` fails? Mention pertinent properties of ELF files in your answer.

Problem 2 (10 pts): Nearby is the output of `pmap` showing page table virtual memory mapping information for a running program called `memory_parts`. Answer the following questions about this output.

(A) The mapped memory references something called `libc-2.26.so`. Describe this entity and what kind of information you would expect to find at the mapped locations.

(B) Why does `pmap` only show a limited number of virtual addresses? What would happen if the program attempted to access an address not listed in the output? Example: address `0x00` is not in the listing.

```
> pmap 7986
7986:   ./memory_parts
00005579a4abd000      4K r-x-- memory_parts
00005579a4cbd000      4K r---- memory_parts
00005579a4cbe000      4K rw--- memory_parts
00005579a4cbf000      4K rw---   [ anon ]
00005579a53aa000    132K rw---   [ heap ]
00007f441f2e1000   1720K r-x-- libc-2.26.so
00007f441f48f000   2044K ----- libc-2.26.so
00007f441f68e000     16K r---- libc-2.26.so
00007f441f692000      8K rw--- libc-2.26.so
00007f441f694000     16K rw---   [ anon ]
00007f441f698000    148K r-x-- ld-2.26.so
00007f441f88f000      8K rw---   [ anon ]
00007f441f8bb000      4K r---- gettysburg.txt
00007f441f8bc000      4K r---- ld-2.26.so
00007f441f8bd000      4K rw--- ld-2.26.so
00007f441f8be000      4K rw---   [ anon ]
00007fff96ae1000    132K rw---   [ stack ]
00007fff96b48000     12K r----   [ anon ]
00007fff96b4b000      8K r-x--   [ anon ]
 total             4276K
```

Problem 3 (10 pts): A recent project involved parsing an ELF file to access its symbol table. While parsing the file, entries in the Section Header Table and Symbol Table had fields associated with string names but these fields were just numbers rather than string names.
(A) How did one actually obtain the name of a Section Header or Symbol in ELF files? Give code snippets AND a verbal explanation.

(B) The designers of the ELF format chose this scheme for including strings in ELF files. Describe how this design saves space and provides flexibility in the file by contrasting it with the use of a fixed character array for names like `char sh_name[128]`.

Problem 4 (10 pts): We have seen that a common use of `mmap()` is to map files into the virtual memory space of a program to make it easy for them to processed. However, this is only one of the uses for `mmap()` which is a fundamental tool for programs to interact with the Operating System and hardware. The Loader is the program which will take the disk image of a program like `a.out` and load it into memory to run. Discuss how `mmap()` can be used by the loader to place the required sections of ELF files in memory and establish areas such as the Stack and Heap for that program. In this, mention what important data structure about a program `mmap()` manipulates.