Midterm 1 Solution Set.

Question 1

| | | Threatening | |
|---|---|---|---|
| | | OS | USER |
| Vulnerable | OS | Write good code | Dual mode (user/priv mode) Memory protection between OS and user processes (this is typically provided by the OS) |
| | USER | Encapsulate parts of malfunctioning OS, so that user can still run Reliability services | Memory protection between different user processes (this is typically provided by the OS) |

Question 2

1 T/F. Procedures executing as part of the same program can share data via the stack.

TRUE. This is how procedures pass arguments.

2. T/F. The operating system kernel stores interrupt state on the user stack.

FALSE. If interrupt state were stored on the user stack, a user program could modify it.

3. T/F. Only non-privileged instructions can be used to modify a page table base register (peanut butter register).

FALSE. Only privileged instructions can do this.

4. T/F. The X86 supports segments, or paging, but not both at the same time.

FALSE. Paged segments are described in V3 of the x86 reference manual.

5. Which one of the following three was not one of the early "killer apps" (eg, driving applications) for digital computing technology:
- Ballistics - Gambling - Space Exploration

Space Exploration was not a killer app.
WWII ballistics requirements drove the Eniac.
Gambling (odds-making at the race track) brought investment into Univac.

6. Which one of the following three technology inflection points offers the best insight into the rise of timesharing systems:
- the magnetic disk - the inexpensive CRT - large random access memories

The inexpensive CRT made it possible to equip users with their own high speed input/output device.

7. Fast processors and slow I/O systems motivated the invention of what operating system technique?

The best answer was spooling. If you put down “multiprogramming” and were marked off, please let your TA know.

8. Why was IBM able to dominate the computing industry during the latter half of the 20th century?

They already owned the customer/channel by virtue of their tabulator products from the first half of the century. Think “punch card.”

9. What is the purpose of the IDT on the X86?

The IDT (Interrupt Descriptor Table) is a dispatch table containing the entry address for interrupt handlers.

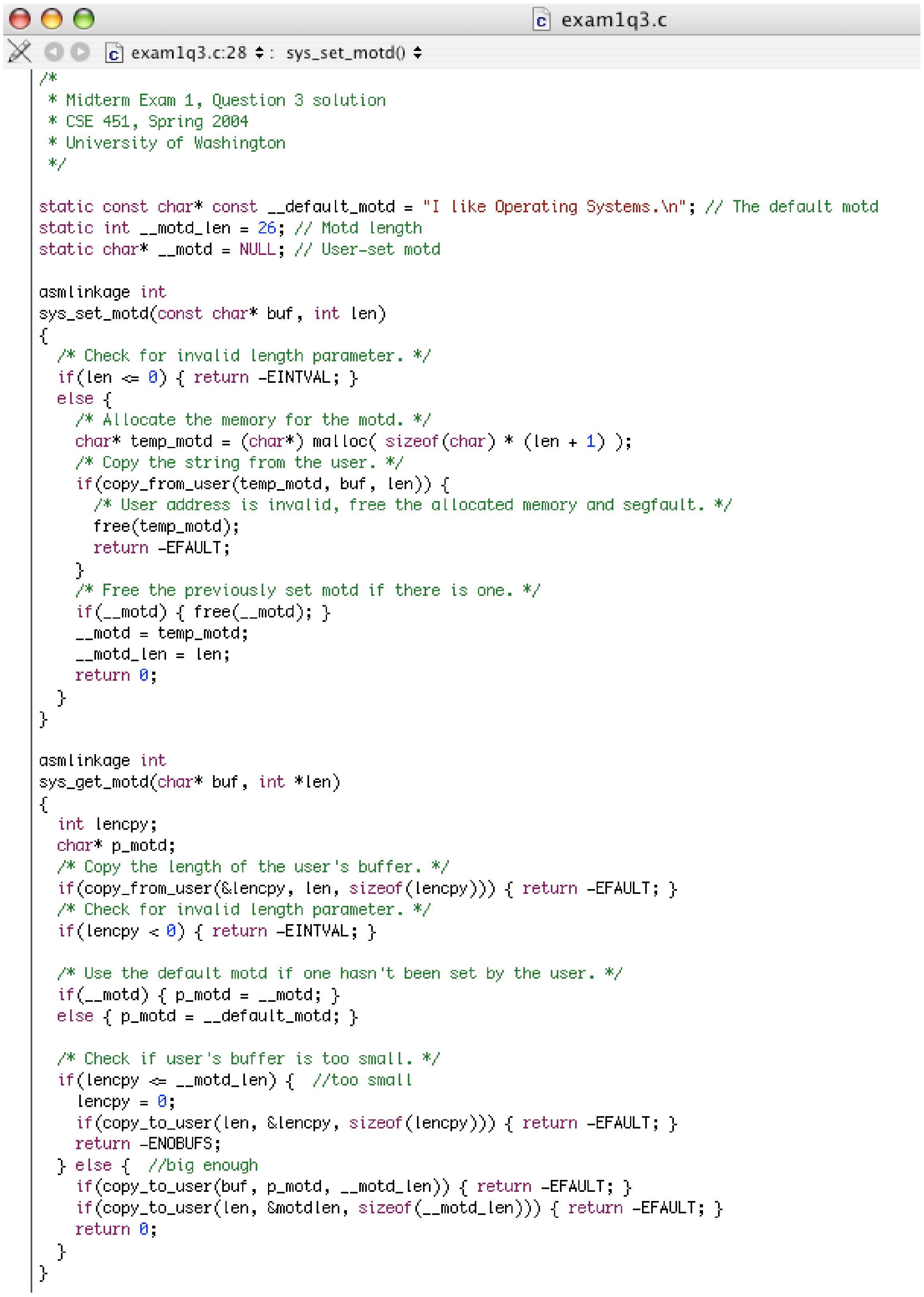
10. Why did Fortran have a COMMON declaration?

Small memories and no VM meant that programmers had to manually overlay their code modules for a big program. If a module needed to share data with another module (eg, a big array), it would declare the array to be common and therefore not overlaid.

QUESTION 3.

Lots of ways to solve this problem. The main things we were looking for were:

- did you get the linkage (function call naming) right
- did you show appropriate care (fear!) when accessing user space pointers
- did you free that which you malloced, and malloc that which you freed.



exam1q3.c

exam1q3.c:28 : sys_set_motd()

```c
/*
 * Midterm Exam 1, Question 3 solution
 * CSE 451, Spring 2004
 * University of Washington
 */

static const char* const __default_motd = "I like Operating Systems.\n"; // The default motd
static int __motd_len = 26; // Motd length
static char* __motd = NULL; // User-set motd

asmlinkage int
sys_set_motd(const char* buf, int len)
{
  /* Check for invalid length parameter. */
  if(len <= 0) { return -EINTVAL; }
  else {
    /* Allocate the memory for the motd. */
    char* temp_motd = (char*) malloc( sizeof(char) * (len + 1) );
    /* Copy the string from the user. */
    if(copy_from_user(temp_motd, buf, len)) {
      /* User address is invalid, free the allocated memory and segfault. */
      free(temp_motd);
      return -EFAULT;
    }
    /* Free the previously set motd if there is one. */
    if(__motd) { free(__motd); }
    __motd = temp_motd;
    __motd_len = len;
    return 0;
  }
}

asmlinkage int
sys_get_motd(char* buf, int *len)
{
  int lencpy;
  char* p_motd;
  /* Copy the length of the user's buffer. */
  if(copy_from_user(&lencpy, len, sizeof(lencpy))) { return -EFAULT; }
  /* Check for invalid length parameter. */
  if(lencpy < 0) { return -EINTVAL; }

  /* Use the default motd if one hasn't been set by the user. */
  if(__motd) { p_motd = __motd; }
  else { p_motd = __default_motd; }

  /* Check if user's buffer is too small. */
  if(lencpy <= __motd_len) {  //too small
    lencpy = 0;
    if(copy_to_user(len, &lencpy, sizeof(lencpy))) { return -EFAULT; }
    return -ENOBUFS;
  } else {  //big enough
    if(copy_to_user(buf, p_motd, __motd_len)) { return -EFAULT; }
    if(copy_to_user(len, &motdlen, sizeof(__motd_len))) { return -EFAULT; }
    return 0;
  }
}
```