CMPT 218 Final Fall 2023

Dec 16, 2023

- Time: 12:00 pm — 3:00 pm
- Please write your answers on the space provided.
- This is a closed-book, no calculator, closed-notes exam.
- There are 20 questions.

Short Answer Questions

Please answer the following questions in 2-3 sentences *maximum*. You can only use the space provided. Do *not* just write what you think is relevant in hopes to get some partial credit. Verbose answers will result in point *reductions*.

1. **(2 pts)** Explain the difference between a hard link and a soft link.

2. **(2 pts)** What does CIA stand for?

3. **(2 pts)** Explain what a window of validity means.

4. **(2 pts)** What is the difference between a private and a public cryptographic algorithm?

5. **(2 pts)** What is the difference between a pipe and a FIFO?

6. **(2 pts)** What is the difference between file mapping and anonymous mapping in `mmap()`?

7. **(2 pts)** What is the purpose of `ftruncate()` for shared memory?

8. **(2 pts)** Provide an example function for buffered file I/O and also an example function for unbuffered file I/O.

9. **(2 pts)** Explain what an i-node is.

10. **(2 pts)** Suppose you have a vacation photo of yourself that you want to post on a social media website. However, you are worried that the photo might be downloaded later and edited (Photoshopped) by someone else. Thus, you want to provide extra information along with the photo (as part of your social media post) so that, as long as others get both the photo and the extra information, they can verify that the photo is not Photoshopped. What kind of crypto primitives and/or applications can you use to achieve this goal?

11. **(2 pts)** Explain the purpose of using the `epoll` API.

12. **(2 pts)** What is a passive socket?

13. (2 pts) What is the difference between a stream socket and a datagram socket?

14. (2 pts) Explain what strong collision resistance means.

15. (2 pts) Explain why there is a file offset for each open file.

Long Answer Questions

You can only use the space provided. All answers/code should be concise and to the point. Do *not* just write what you think is relevant in hopes to get some partial credit. Verbose answers will result in point *reductions*.

16. (4 pts) Suppose that the following two functions run in parallel as two different pthreads. The two functions are supposed to print out "ping" and "pong" alternatively, and "ping" should always be printed out before "pong". They synchronize with each other using two semaphores `sem_ping` and `sem_pong`.

```
sem_t sem_ping, sem_pong;

void *ping(void *arg)
{
  for (int i = 0; i < 10; i++) {
    // (1)
    printf(``ping\n'');
    // (2)
  }
  pthread_exit(NULL);
}

void *pong(void *arg)
{
  for (int i = 0; i < 10; i++) {
    // (3)
    printf(``pong\n'');
    // (4)
  }
  pthread_exit(NULL);
}
```

Note that `sem_wait()` and `sem_post()` are used to wait and signal a semaphore, respectively. What are the correct initial values of the semaphores and what are the correct uses of the semaphores in the four places marked with (1)-(4)?

17. (4 pts) Consider the following code:

```
int x = 0;     // A global variable that is shared among all threads
void update(int v1, int v2) {
  x += (v1 * v2)
}
```

Suppose we have two threads (`T1` and `T2`) executing `update(3, 4)` at the same time on two different CPU cores. The initial value of `x` is 0. What are the possible final results of `x`? Describe clearly yet succinctly how you reach these results (no points given if you only provide the final results).

18. (4 pts) Suppose we have a thread (called `T1`) running the following code:

```
T1_access() {
  access(&a);
  access(&b);
}
```

Here, `a` and `b` are two global variables, and `access` is a function that takes a pointer to the variable to access. Internally, `access` takes a lock for the variable being accessed. Suppose another thread `T2` runs the following code:

```
T2_access() {
  access(&b);
  access(&c);
  access(&a);
}
```

Now suppose that when `T1` and `T2` run simultaneously, a deadlock occurs.

a) **(2 pts)** Explain in 2–3 sentences why there could be a deadlock between `T1` and `T2`.

b) **(2 pts)** Suggest a change to `T2_access` to avoid the deadlock. You are not allowed to defined new variables/structures or use any other APIs/atomics/locks.

19. (4 pts) Consider the following code snippet:

```
int main() {
  struct sockaddr_in addr;
  int sfd;
  ssize_t num_read;
  char buf[64];

  sfd = socket(AF_INET, SOCK_STREAM, 0);

  memset(&addr, 0, sizeof(struct sockaddr_in));
  addr.sin_family = AF_INET;
  addr.sin_port = 8000;
  addr.sin_addr.s_addr = INADDR_ANY;

  bind(sfd, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
  listen(sfd, 32);

  for (;;) {
    accept(sfd, NULL, NULL);

    while ((num_read = read(sfd, buf, 64)) > 0)
      write(STDOUT_FILENO, buf, num_read);
  }
}
```

For simplicity, the code does not have any error checking, any cleanup code, or any headers. There are four problems with the code. Identify the problems and provide the corresponding fixes.

20. (4 pts) Write a program that creates a child process and communicates with the child process using a pipe. Here are the requirements:

a) The parent should write to its standard output to write to the pipe.

b) The child should read from its standard input to read from the pipe.

c) The parent should send a string "Hello, world!" to the child, and the child should print out the string to its own standard output.

Here are some of the useful constructs you can use.

a) `int pipe(int filedes[2])`: creates a pipe, where `filedes[0]` is the read end and `filedes[1]` is the write end.

b) `int dup2(int oldfd, int newfd)`: duplicates the `oldfd` file descriptor to `newfd`.

c) `STDOUT_FILENO`: the file descriptor for the standard output.

d) `STDIN_FILENO`: the file descriptor for the standard input.

For simplicity, do not worry about error handling.