CS 451/551 Quiz 6 Annotated Solution

Date: May 4, 2016 **Max Points**: 15

Important Reminder As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

1. Which one of the following IPC mechanisms is a good choice when efficiency is a primary concern?

   a) Anonymous pipes.

   b) FIFO's.

   c) Shared memory.

   d) Message queues.

   e) Files.

   Answer: (c)

   All of the above facilities require kernel involvement during set up, but shared memory does not require any kernel involvement during the actual IPC. All the alternatives require a user-space to system-space transition during the IPC. Hence shared memory is a leading contender for an efficient IPC mechanism.

2. Which one of the following statements about threads within a single process is necessarily **false**?

   a) It is always possible for any thread to cancel any other arbitrary thread.

   b) If control returns from `main()`, then all threads are terminated.

   c) If a thread is detached, then it's exit status will not be available to other threads.

   d) Code which is MT-safe can be executed correctly by multiple concurrent threads.

   e) Each thread has its own stack.

   Answer: (a)

   A thread can cancel another thread only if the subject thread permits cancellation. The other statements are true because:

   - Control returning from `main()` terminates the entire process and consequently all contained threads.
   - If a thread is detached, then it is not possible for other threads to join with it and access it's termination status.

- The definition of MT-safe is that the code can be run correctly by concurrent threads.
- Each thread has a separate locus of control, it needs to have its own stack.

3. Which one of the following statements about the correspondence between pthread mutexes and condition variables is necessarily **true**?

   a) There is a 1:1 correspondence between each condition variables and its controlling mutex.

   b) Each condition variable can have one-or-more controlling mutexes.

   c) Each mutex can control one-or-more condition variables.

   d) Each condition variable can have one-or-more controlling mutexes and each mutex can control one-or-more condition variables.

   e) A condition variable can be used without any corresponding mutex.

   Answer: (c)

   A condition variable must wait and be signalled under the same mutex, but a mutex can control multiple conditions (a queue can be full or empty).

4. Which one of the following constitutes a correct use of condition variables to check a shared `condition()` within a mult-threaded context?

```
a)  pthread_mutex_t mutex = PTHREAD_MUTEX_INIITIALIZER;
    pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
    if (!condition()) pthread_cond_wait(&cond, &mutex);
    //process condition

b)  pthread_mutex_t mutex = PTHREAD_MUTEX_INIITIALIZER;
    pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
    while (!condition()) pthread_cond_wait(&cond, &mutex);
    //process condition

c)  pthread_mutex_t mutex = PTHREAD_MUTEX_INIITIALIZER;
    pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
    pthread_mutex_lock(&mutex);
    while (!condition()) pthread_cond_wait(&cond, &mutex);
    //process condition
    pthread_mutex_unlock(&mutex);

d)  pthread_mutex_t mutex = PTHREAD_MUTEX_INIITIALIZER;
    pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
    pthread_mutex_lock(&mutex);
    if (!condition()) pthread_cond_wait(&cond, &mutex);
    //process condition
    pthread_mutex_unlock(&mutex);
```

e)
```
pthread_mutex_t mutex = PTHREAD_MUTEX_INIITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_cond_wait(&cond, &mutex);
//process condition
```

Answer: (c)

A condition variable should be waited for only when its controlling mutex is locked; this precludes (a), (b) and (e). It is necessary the condition should be checked again after the wait; this preclused (d). This leaves (c).

5. Which one of the following sequence of calls for creating and using a streaming **server** socket is valid?

   a) `socket(); bind(); listen(); accept()`.

   b) `socket(); bind(); listen(); connect(); accept()`.

   c) `socket(); bind(); connect(); accept()`.

   d) `socket(); connect(); listen(); accept()`.

   e) `socket(); connect()`.

   Answer: (a)

   `connect()` is used only for client sockets.

6. Which one of the following sequence of calls for creating and using a streaming **client** socket is valid?

   a) `socket(); bind(); listen(); accept()`.

   b) `socket(); bind(); listen(); connect(); accept()`.

   c) `socket(); bind(); connect(); accept()`.

   d) `socket(); connect(); listen(); accept()`.

   e) `socket(); connect()`.

   Answer: (e)

   The correct sequence of calls is `socket(); connect()`. `bind()`, `listen()` and `accept()` are only used for server sockets.

7. Assuming no errors and that `printf()` always runs atomically, what will be printed by the following program?

```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
void *fn(void *p) {
  int *ip = p;
  pthread_mutex_lock(&lock);
  *ip += 5;
  pthread_mutex_unlock(&lock);
  return NULL;
}

int main() {
  int v = 5;
  pthread_t t;
  pthread_create(&t, NULL, fn, &v);
  fn(&v);
  printf("%d\n", v);
  return 0;
}
```

a) `10` followed by `15`.

b) Only `15` will always be printed.

c) One of `5`, `10` or `15` will always be printed.

d) Only `10` will always be printed.

e) One of `10` or `15` will always be printed.

Answer: (e).

[During the exam it was announced that if alternatives overlapped, the more accurate alternative should be chosen.]

The `main()` thread always calls `fn()`; hence `v` will always be incremented at least once to `10`. However there is a race condition between execution of the auxiliary thread and the execution of the `printf()` by the main thread (it is also possible that the auxiliary thread cannot even increment `v` before the main thread exits, terminating the entire process including the auxiliary thread). Hence the auxiliary thread may or may not have a chance to increment `v` before the `printf()`. Hence the output will be one of `10` or ‘15.