An Airline Check-in System Notes & Tips

Huan Wang

huanwang@uvic.ca

Notes

- Include corresponding header files
  - #include <pthread.h>: for multi-thread related system call
  - #include <sys/types.h>: e.g., thread id type “pthread_t”
  - …
- Compile
  - Example: **$gcc –pthread pt_test.c –o pt_test**
- Check the return values of key function calls
  - **errno**

The main thread

- **Create customer thread**
- Create all customer threads at once **VS** Create customer thread gradually according to the sequence of arrival time.
- **End of main thread**
- Use pthread_join() to wait for the termination of all customer threads. Clerk threads is not necessary (if any).
- The sequence of join doesn’t matter.

Customer thread

* 1. Get the customer information passed by pthread_create();
* 2. Simulate the arrival time: usleep();
  * **Example:** arrival time : 6 → 0.6 secs → (0.6 * 1,000,000) usecs
  * usleep(6 * 100, 000);
* 3. Pick the queue (based on the class info) to enter.
* 4. Enter queue operation:
  * 1. pthread_mutex_lock(); 2. enter queue operation; 3.length++;
  * 4. pthread_cond_wait();
* 5. Customer signaled by one of the four clerks:
  * 1. pthread_cond_wait() returns,
  * 2. check if "I am" in the head of queue, if NO, go back to pthread_cond_wait.
  * 3. pthread_mutex_unlock();

Customer thread

- 6. Ready to be served by the clerk who signaled me.

  i) Which clerk awoke me? Or who sent the signal?

  ii) Get current simulation time (needs to print the time information)

  iii) Update the overall_waiting_time, queue_waiting_time
-
- 7. Get served by clerk: usleeep();
- 8. Tell the clerk service is finished, you can serve other customers: Calling pthread_cond_signal();

Which clerk signaled me?

- Idea:
  - 1. Before sending signal, the clerk thread writes its clerk id into a global status variable for the queue it is going to signal (i.e., we have global variable for each queue).
  - 2. When customer is signaled, it checks the value of its queue's global status variable.
  - 3. if (clerkA_id == status) ➔ Clerk A send the signal!

Potential troubles

- 1. Multiple clerks send multiple signals to the same queue, at the same time.

  hint: set the queues with IDLE or BUSY status to prevent other clerks sending signals when a queue is busy.

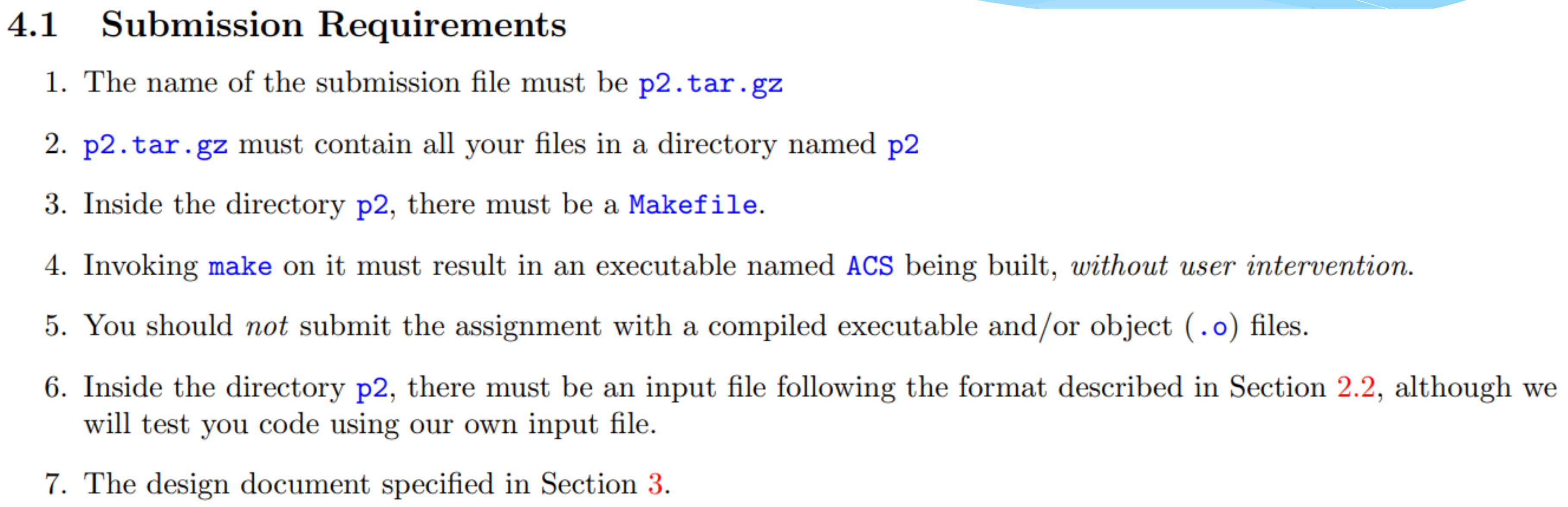
- 2. When using pthread_cond_broadcast to wake up all customers, the 2nd customer in the queue will become the head of the queue once the real 1st customer leaves the queue. Then the 2nd customer leaves the queue too.

  hint: once leaving the queue, the 1st customer has to lock the door to prevent other customers leaving.

Clerk Thread (if any)

- 1. Check if there are customers waiting in the two queues.
  - If there are customers ➔ return the queue id to send signal
  - No customer is waiting ➔ wait for a while, check it again
- Fetch the first customer from the picked queue.
  - pthread_cond_signal/broadcast(pickedqueue);
  - Need to lock/unlock the mutex of the queue
- Waiting for the customer to finish his service
  - pthread_cond_wait(); need mutex lock/unlock

Submission Requirements



4.1 Submission Requirements

1. The name of the submission file must be `p2.tar.gz`
2. `p2.tar.gz` must contain all your files in a directory named `p2`
3. Inside the directory `p2`, there must be a `Makefile`.
4. Invoking `make` on it must result in an executable named `ACS` being built, *without user intervention.*
5. You should *not* submit the assignment with a compiled executable and/or object (`.o`) files.
6. Inside the directory `p2`, there must be an input file following the format described in Section 2.2, although we will test you code using our own input file.
7. The design document specified in Section 3.

Announcement

- Next week will be a Q&A tutorial !