CS 307 PA 3 Report

1. Input parsing

Argc and argv arguments in main are used to read numbers from console. The arguments are than cast to int with atoi() function. Than we check at both arguments are even and their sum is divisible by for. If any of the conditions do not hold main terminates with return 1.

2. Synchronization mechanisms

Barrier named “barrier” is used to wait for all the band when a driver is found. It is initilazied when main start and destroyed and re-intilazied each time a band rides with value of 4. Two sets of semaphores are used to coordinate functions. First set is sem, semPrint. They are used as mutex locks, and their initial value is one. Sem is used as global lock to coordinate threads and semprint is used to coordinate printing of line I have found a spot in a car among riders.

sem_A_waiters, sem_B_waiters are used to coordinate team members who wait for car. Waiters wait on sem_wait and when a rider is found it sem_posts enough times to let same waiters out of wait queue. They are initialized with value 0 so that only driver can let people out of wait queue by defult. Global variables A_waiters, B_waiters are used to coordinate number of waiters of each time. Values start as 4, Every time somebody waits one semaphore, they decrease value by one. And every time somebody post’s some semaphores they increase the value by number of semaphore’s they post. Details are explained below.

3. Main and Thread Logic

Main initialize all synchronization variables with values described above. Later it creates threads in a loop using numbers entered to terminal and passes a char variable to each of them to let them know of the team. All threads are kept on a queue. Main than waits on the queue for all threads to finish and when they do it destroys the barrier and terminates.

Threads keep a semaphore pointer for their friends (people on same team) and others (people on other team). First, they make this pointer point to correct semaphore based on their team. They also make an integer pointer point to appropriate global variables for number of friends waiting and number of others waiting numbers. Now Thread first checks if enough people are waiting to form a car ride. If at least one friend and 2 others are waiting (ie waiting numbers are less 4 for the team and less than 3 for others.) it forms a ride. It declares itself as the driver and posts semaphore for one friend and two others, so they escape the wait semaphore and ride with driver thread. And it also increases wait num for friends by one and wait num for others by 2 since we let them go. Else we check if at least 3 other friends are waiting (ie. Num wait for friends is less than 2). If they do thread again declares itself as the driver and posts friend semaphore three times to let 3 friends escape wait. Than it increases wait number for friends by three. If all two checks fail thread can’t declare itself as rider and must put itself on a wait queue and change the wait num so another driver my ride with it. First it waits number for friends by one, than it lets go of the global lock(sem) so other threads can run and it calls sem_wait to wait.

Once a thread escapes the sem_wait (or never enters it because she is driver), She prints that she has found a spot on a car (print is synchronized by semPrint described above). After threads wait on barrier so all four of the band can do the print. After all band prints only, driver declares herself as the driver and then she resets the barrier as described in part 2. The driver lets go of the global lock here.