Ahmet Bilal Yildiz 27925

CS307- PA3-RIDE SHARE

General Information:

In this homework, the main thread creates the child fan threads with given number and the child threads are ridesharing.

For the implementation of child threads POSIX pthread_t,

For the implementation of semaphores POSIX sem_t ,

For the implementation of mutexes POSIX pthread_mutex_t,

For the implementation of barriers POSIX pthread_barrier are used.

Flow of Threads (Pseudo code)

1. Main thread starts to run.
2. numA and numB are assigned to given inputs.
3. if given inputs are invalid main thread terminates, else continue with the next steps
4. A thread list and all child threads are initialized.
5. For each one of the numA fan,
    a. the corresponding child thread is created with functionA.
6. For each one of the numB fan,
    a. the corresponding child thread is created with functionB.
7. For each thread in thread list,
    a. the corresponding child thread is joined.
8. Free all the dynamically allocated memory, give terminate message and terminate.

Ahmet Bilal Yildiz 27925

FunctionA (This function is used in team A threads)

1. Lock the mutex
2. Display the init message
3. Update the waiting info for waitingA
4. Check whether the conditions are satisfied for creating a band.
5. if conditions are satisfied (either 4 team A fans or 2 team A and 2 team B fans):
    - **open the waiting sems** (with sem_post)
    - be the captain
6. else:
    - unlock the mutex
    - **wait** until the band condition is satisfied (with sem_wait)
7. After if/else period, give the mid message and wait for **barrier** (wait other threads to give mid messages)
8. if current thread is captain:
    - give the end message
    - unlock the mutex

FunctionB (This function is used in team B threads)

1. Lock the mutex
2. Display the init message
3. Update the waiting info for waitingB
4. Check whether the conditions are satisfied for creating a band.
5. if conditions are satisfied (either 4 team B fans or 2 team A and 2 team B fans):
    - **open the waiting sems** (with sem_post)
    - be the captain
6. else:
    - unlock the mutec
    - **wait** until the band condition is satisfied (with sem_wait)
7. After if/else period, give the mid message and wait for **barrier** (wait other threads to give mid messages)
8. if current thread is captain:
    - give the end message
    - unlock the mutex

Ahmet Bilal Yildiz 27925

Syncronization

As it can be seen from the pseudo code syncronization is provided by the a mutex, two semaphores (1 for team A threads and 1 for team B threads), and a barrier.

Mutex is used to prevent mixture in the update of the global variables that stores the information of waiting fans and also to prevent mixture while printing messages to the console.

Semaphores are used to ensure a fan thread waits for other threads untill a band condition is satisfied. When the condition is satisfied, related semaphores open and related fan threads are continue.

Barrier is used to ensure that all threads give their mid messages, in other words after entering a car, a fan must wait the other fans to enter the car before the car moves it must be full which ensured by the barrier.

Correctness

I think my solution is correct because the implementation of the threds and syncronization are correct.

Main thread create child threads as many as the given numbers and ensures that all the child threads are finished before termination by joining all of them. If the given numbers are invalid then terminates after give a message.

Each child thread lock mutex before update the globally shared variable and unlock mutex before terminate.

Each child thread wait for the band conditions to satisfy, then goes on.

Each child thread waits for other threads to go on after displaying the mid message.

Therefore, for each thread there are init and mid messages, all related init messages are before the mid messages, for each band there is one end message (car drive message), between 2 end messages there are 4 mid messages, for each thread the messages are in init-mid-end(if exists) order, the thread gives end message is one of the threads that gives mid message beforehand and band contions are valid combinations.