Alan Padilla Chua

AXP141330

Design

Semaphores:

1. door = 0 – Depending on the number of permits people may enter elevator once the people enter the others must wait for more permits to enter. The permits granted will be 7 for the max capacity.
2. Elevatorfull = 0 – Binary semaphore that Person threads signals telling elevator that it is full
3. enterLobby = 49 – 49 people enter lobby
4. enterMutex = 1 – enter semaphore for mutual exclusion because array list is being modified
5. capacity = 0 – semaphore waits for signal to allow people in elevator
6. wantedFloor[49] = {0} – semaphore array each thread waits for their specific semaphore to know that they are at their floor
7. leftElevator = 0 – semaphore that elevator thread waits in order to signal next wantedFloor[] semaphore
8. exitMutex = 1 – leave semaphore for mutual exclusion because array list is being modified

Pseudocode:

Elevator Methods:

Void OpenDoor() = Output that the door is open at the current floor;

Void CloseDoor() = Outputs that door is closed

Int gotoFloor(){

Smallest floor = 11

while the arraylist is not empty

loop through every element of arraylist

if currents element request floor is smaller than the smallest floor

new smallest floor is the current requested floor

send smallest floor

if list is empty send 1 so elevator returns to lobby

}

Void checkFinal() {

If the number that as exited the elevator is equal to max people in lobby (49)

Output that the Simulation is complete

Exit the program

}

processFloor(){

get next floor = gotofloor

while the floor is not 1

set the current floor to the next floor

output the the door for the elevator has opend at the floor

if the list is not empty

while(the lowest floor has not changed)

loop through the array list of people (i++)

if a persons requested floor matches the current floor

signal(wantedFloor[i])

wait(leftElevator)

check if lowest floor has changed

close the door

get next floor

once next floor is 1

clear the arraylist

set the floor to the lobby

}

void Elevator {

while(true) {

OpenDoor();

Signal(door,7) :// signal that door is open giving 7 permits

Signal(capacity,7) ;//signal that it only wants 7 people to enter elevator

wait(Elevatorfull); // wait for person thread to signal that elevator has reached capacity

CloseDoor();

processFloors();

checkFinal();

}

}

Person methods:

Void enterElevator(Person){

add person to elevator arraylist

increase count

}

Void leaveElevator(Person){

Remove person from elevator arraylist

Decrease count

Increase person left count

}

Void checkifFull(){

If current count is equal to the max capacity of elevator

Signal(Elevatorfull)

}

Void Person(){

Wait(enterLobby) // 49 people(threads) enter the lobby

Wait(capacity) // elevator must signal that it want 7 people in the elevator

Wait(door) // wait for door to signal open but also only 7 people may enter

Wait(enterMutex) // get the enterMutex permit

enterElevator(this) // the current person threads enters elevator

signal(enterMutex) // signals to next person that its okay to enter

checkIfFull()

wait(wantedFloor(this.id))

wait(exitMutex)

leaveElevator(this)

signal(exitMutex)

signal(leftElevator)

}