CS105 – Computer Systems Spring 2021

Problem Session 11: Synchronization

Wednesday, April 14, 2020

1. Assume that LA county decides to one day create a functional public transit system, including a bus network. Riders come to a bus stop and wait for a bus. When the bus arrives, all the waiting riders board the bus one after another, but anyone who arrives while the bus is boarding has to wait for the next bus. The capacity of these hypothetical LA buses is 50 people; if there are more than 50 people waiting, some will have to wait for the next bus. When all the waiting riders have boarded, the bus may depart. If the bus arrives when there are no riders, it should depart immediately. Fix the following (incorrectly synchronized) code to guarantee that these constraints are all satisfied. You may assume that all passengers and bus drivers know how to use semaphores.

```
// global variables
int people_in_line = 0

sem_t bus = sem_init(50)
sem_t boarded = sem_init(1)
```

```
void bus(){                                         void passenger(){


    for(int i = 0; i < 50; i++){                        people_in_line++;
        V(bus);
        P(boarded);
    }


    people_in_line = max(people_in_line-50, 0);         P(bus);
                                                        boardBus();
                                                        V(boarded);

    depart();


}                                                   }
```

2. Use locks and condition variables to synchronize the bounded buffer example we discussed in class.

```
typedef struct {
    int *b;
    int n;
    int front;
    int rear;




}


void init(bbuf_t * ptr, int n){
    ptr->b = malloc(n*sizeof(int));
    ptr->n = n;
    ptr->front = 0;
    ptr->rear = 0;




}
```

```
void put(bbuf_t * ptr, int val){




    ptr->b[ptr->rear] = val;
    ptr->rear = (ptr->rear + 1) % ptr->n;



}

int get(bbuf_t* ptr){




    int val = ptr->b[ptr->front];
    ptr->front = (ptr->front + 1) % ptr->n;



}
```