**Assignment – 8 (OS)**

**U20CS110**

**Krishna Pandey**

**1: The Sleeping-Barber Problem.**

**A barbershop consists of a waiting room with n chairs and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber.**

1. **Write a program to coordinate the barber and the customers.**

Answer:

We use 3 semaphores. Semaphore customers counts waiting customers; semaphore

barbers is the number of idle barbers (0 or 1); and mutex is used for mutual exclusion. A

shared data variable customers1 also counts waiting customers. It is a copy of customers.

But we need it here because we can’t access the value of semaphores directly. We also

need a semaphore cutting which ensures that the barber won’t cut another customer’s

hair before the previous customer leaves.

// shared data

semaphore customers = 0;

semaphore barbers = 0;

semaphore cutting = 0;

semaphore mutex = 1;

int customer1 = 0;

void barber() {

while(true) {

wait(customers); //sleep when there are no waiting customers

wait(mutex); //mutex for accessing customers1

customers1 = customers1 - 1;

signal(barbers);

signal(mutex);

cut\_hair();

}

}

void customer() {

wait(mutex); //mutex for accessing customers1

if (customers1 < n) {

customers1 = customers1 + 1;

signal(customers);

signal(mutex);

wait(barbers); //wait for available barbers

get\_haircut();

}

else { //do nothing (leave) when all chairs are used.

signal(mutex);

}

}

cut\_hair(){

waiting(cutting);

}

get\_haircut(){

get hair cut for some time;

signal(cutting);

}

**(2) Consider the Sleeping-Barber Problem with the modification that there are k barbers and k barber chairs in the barber room, instead of just one. Write a program to coordinate the barbers and the customers.**

**Answer:**

// shared data

semaphore waiting\_room\_mutex = 1;

semaphore barber\_room\_mutex = 1;

semaphore barber\_chair\_free = k;

semaphore sleepy\_barbers = 0;

semaphore barber\_chairs[k] = {0, 0, 0, …};

int barber\_chair\_states[k] = {0, 0, 0, …};

int num\_waiting\_chairs\_free = N;

boolean customer\_entry( ) {

// try to make it into waiting room

wait(waiting\_room\_mutex);

if (num\_waiting\_chairs\_free == 0) {

signal(waiting\_room\_mutex);

return false;

}

num\_waiting\_chairs\_free--; // grabbed a chair

signal(waiting\_room\_mutex);

// now, wait until there is a barber chair free

wait(barber\_chair\_free);

// a barber chair is free, so release waiting room chair

wait(waiting\_room\_mutex);

wait(barber\_room\_mutex);

num\_waiting\_chairs\_free++;

signal(waiting\_room\_mutex);

// now grab a barber chair

int mychair;

for (int I=0; I<k; I++) {

if (barber\_chair\_states[I] == 0) { // 0 = empty chair

mychair = I;

break;

}

}

barber\_chair\_states[mychair] = 1; // 1 = haircut needed

signal(barber\_room\_mutex);

// now wake up barber, and sleep until haircut done

signal(sleepy\_barbers);

wait(barber\_chairs[mychair]);

// great! haircut is done, let’s leave. barber

// has taken care of the barber\_chair\_states array.

signal(barber\_chair\_free);

return true;

}

void barber\_enters() {

while(1) {

// wait for a customer

wait(sleepy\_barbers);

// find the customer

wait(barber\_room\_mutex);

int mychair;

for (int I=0; I<k; I++) {

if (barber\_chair\_states[I] == 1) {

mychair = I;

break;

}

}

barber\_chair\_states[mychair] = 2; // 2 = cutting hair

signal(barber\_room\_mutex);

// CUT HAIR HERE

cut\_hair(mychair);

// now wake up customer

wait(barber\_room\_mutex);

barber\_chair\_states[mychair] = 0; // 0 = empty chair

signal(barber\_chair[mychair]);

signal(barber\_room\_mutex);

// all done, we’ll loop and sleep again

}

}