Junsik Seo

Professor Greg Ozbirn

CS 4348

April 1, 2021

Project 2: Threads and Semaphores

* recp\_ready
  + Determine whether receptionist is ready
  + Initial value = 1
* pat\_recep
  + Deter whether patient is ready (in receptionist line)
  + Initial value = 0
* register
  + Notify a patient that receptionist register a patient
  + Initial value = 0
* pat\_wait
  + Notify a receptionist that patient leaves receptionist and sits in waiting room
  + Initial value = 0
* nurse\_pat
  + Notify a patient that nurse takes patient to doctor’s office
  + Initial value = 0
* pat\_doc
  + Notify a nurse that patient enters doctor’s office
  + Initial value = 0
* advice
  + Notify a patient that doctor gives an advice to patient
  + Initial value = 0
* mutex\_recep
  + Enforces mutual exclusion of adding and removing from the receptionist queue.
  + Initial value = 1
* mutex\_nurse
  + Enforces mutual exclusion of adding and removing from the nurse queue.
  + Initial value = 1
* mutex\_doc
  + Enforces mutual exclusion of adding and removing from the doctor queue.
  + Initial value = 1
* nurse\_ready[doctor’s number]
  + Determine whether nurse is ready
  + Initial value = 1
* recep\_nurse[doctor’s number]
  + Signaling a specific nurse that assigned by patient
  + Initial value = 0
* leave[patient’s number]
  + Determine whether patient leaves from the doctor’s office (for doctor)
  + Initial value = 0
* closedoor[doctor’s number]
  + Determine whether patient enters the doctor’s office (for nurse)
  + Initial value = 0

pseudocode for project2

/\* program project2 \*/

semaphore recp\_ready = 1;

Semaphore pat\_recep = 0;

Semaphore register = 0;

Semaphore pat\_wait = 0;

Semaphore nurse\_pat = 0;

Semaphore pat\_doc = 0;

Semaphore advice = 0;

Semaphore mutex\_recep = 1;

Semaphore mutex\_nurse = 1;

Semaphore mutex\_doc = 1;

Semaphore[] nurse\_ready;

Semaphore[] recep\_nurse;

Semaphore[] leave;

Semaphore[] closedoor;

Queue receptionist\_queue

Queue nurse\_queue

Queue doctor\_queue

void main(string [] args) {

doctor’s number=args[0]

patient’s number=args[1]

for ( i< patient’s number)

leave[i]=semaphore(0)

for (i< doctor’s number)

nurse\_ready[i]= semaphore(1)

recep\_nurse[i]=semaphore(0)

close\_door[i]= semaphore(0)

Thread[patient’s number] patient

Thread[doctor’s number] nurse

Thread[doctor’s number] doctor

For(i<doctor’s number)

doctor[i].start()

nurse[i].start()

Thread receptionist

Receptionist.start()

For(i<patient’s number)

patient[i].start()

}

class Patient(){

int ID

int doctor’s number

int maximum doctor’s number

void random\_assign (Integer number){

doctor’s number

}

int patient’s number(){

Return patient’s number

}

int doctor’s number(){

Return doctor’s number

}

void run(){

doctor’s number = random (0 ~ maximum doctor’s number);

patient\_enter();

wait(recp\_ready);

wait(mutex\_recep);

receptionist\_queue.add(this);

signal(mutex\_recep);

wait(register);

patient\_leave\_receptionist();

signal(pat\_wait);

wait(nurse\_pat);

patient\_enter\_doctor\_office();

signal(pat\_doc);

wait(advice);

patient\_receieve\_advice();

paitnet\_leave();

signal(leave[patient’s number]);

}

}

Class Receptionist {

Patient patient

int patient’s number

int doctor’s number

void run(){

wait(pat\_acquire);

wait(mutext\_recep);

patient= receptionist\_queue.poll;

register();

signal(mutex\_recep);

signal(register);

wait(pat\_wait);

wait(mutex\_nurse);

nurse\_queue.add(patient)

signal(mutex\_nurse);

signal(recep\_nurse[doctor’snumber]);

}

}

Class Nurse{

Patient patient

int nurse’s number

int patient’s number

void run(){

wait(recep\_nurse[nurse’s number]);

wait(nurse\_ready[nurse’s number]);

signal(recp\_ready);

wait(mutex\_nurse);

patient= nurse\_queue.poll;

Nurse\_take\_patinent();

signal(mutex\_nurse);

signal(nurse\_pat);

wait(pat\_doc);

wait(mutex\_doc);

doctor\_queue.add(patient)

signal(mutex\_doc);

signal(closedoor[nurse’s number]);

}

}

class Doctor(){

Patient Patient

int doctor’s number

int patient’s number

void run(){

wait(closedoor[doctor’s number]);

wait(mutex\_doc);

patient=doctor\_queue.poll();

Doctor\_listen\_ symptom();

signal(mutex\_doc);

signal(advice);

wait(leave[patient’s number]);

signal(nurse\_ready[doctor’s number]);

}

}

}