CS543

Homework #4

SUBMITTED BY:

Agrima Jindal

1. Briefly sketch the design of a threads-based discrete-event simulator.

(1) Using pictures, try to explain how the language CSIM implements the transfer of control between “sim” and “cust” in the simulation of the single-server queue.

SCHED

SIM

CUST

yields

holds & yields

holds & yields

yields

(2) Explain how the “hold” operation is implemented

Assuming process executing the hold function has activation record E and the current simulation time is E.time = clock. The hold function advance the event occurrence time to t (time to hold) in the future, and insert event E back to the calendar, then yield to the scheduler function.

void hold (float t)

{

E.time = clock + t;

Insert E in simulation calendar;

yield (scheduler);

}

(3) Explain what you will use as an event-activation record

Activation record E:

|  |
| --- |
| E.thread (process which is given control at E.time) |
| E.time (time of event occurrence) |

(4) You will need to provide a “create-wrapper” (to be invoked by the simulator user) which envelopes the actual create primitive, or you will use CSIM's approach and provide a “create” function.

The simulation process:

|  |
| --- |
| FACILITY f;  EVENT done;  sim ()  {  int i; float t;  create(“sim”);  f = facility(“facility”);  done = event (“done”);  for (i=0; i<1000; i++) {  cust ();  hold(expon(seed, mean));  }  wait(done);  print statistics;  }  cust ()  {  create(“cust”);  reserve(f);  hold(expon(seed, mean));  release(f);  if (cust is #1000)  set done = true;  } |

1. Implement a threads-based C-language discrete-event simulator. Use pthreads for this project; you will only need the three basic thread operations:

*create(x)*: create a thread whose name is x

*yield(x)*: yield to thread x

*setpri(x; j)*: set the priority of thread x to an integer j, 1<= j <=2. Note that at any given time only the highest priority thread can run. If x = SELF, the invoking thread sets its own priority.

Write a library of functions that will support a facility with a single server and a fcfs queueing discipline, just as done in CSIM. The facility will support reserve and release methods. Also implement an event with set, clear and wait methods.

Objects:

1. Calendar – record future activity of threads
2. Facility – serve the customers
3. Simevent – only the “done” event in this simulation
4. Asim – the thread operations

Functions:

1. Calendar

* init():
* initialize all the entries in the calendar with a thread id NULL and occur time INF

1. Facility

* init():
* initialize the facility status into FREE
* empty the waiting list for the facility
* clear the statistics (queue length, delay etc.)
* reserve():
* if the facility is FREE, change it to OCCUPIED
* else the thread itself to the waiting list and call yield()
* release():
* if the waiting list is not empty, extract the first thread on the queue and put it in the calendar
* else change the facility status to FREE
* notice\_arrive():
* record the current simulation time as the arrival time of the thread
* notice\_leave():
* record the current simulation time as the depart time of the thread
* report():
* print the statistics (queue length, delay etc.)

1. Simevent

* init():
* initialize the event status to 0
* set():
* set the event status to 1
* clear():
* set the event status to 0
* wait():
* busy wait until event status is 1

1. Asim

* create(funcname):
* create a thread with function funcname and set its priority to MINPRIO
* insert the created thread to calendar
* yield (tid):
* set SELF priority to MINPRIO
* set tid priority to MAXPRIO
* call sched\_yield()
* setpri (tid, pri):
* if tid == SELF, set pthread\_self() priority to pri
* else set tid priority to pri
* hold(t):
* calculate thread\_time = simclock + t;
* insert the SELF thread back into calendar with occur time thread\_time
* yield to scheduler

Simulation Process:

|  |
| --- |
| main() {  create the “sim” thread with maximum priority;  create the “scheduler” thread with maximum priority;  pthread\_join(“sim” thread);  pthread\_join(“scheduler” thread);  pthread\_exit(NULL);  } |

|  |
| --- |
| sim() {  s = new asim;  f = new facility;  done = new simevent;  done->clear();  for (i = 0; i < NARS; i++) {  s->create(cust);  s->hold(IATM);  }  done->wait();  f->report();  } |

|  |
| --- |
| cust() {  f->notice\_arrive();  f->reserve();  s->hold(SVTM);  f->release();  cnt --;  f->notice\_leave();  if (cnt == 0) {  done->set();  }  } |

|  |
| --- |
| scheduler() {  while(1) {  if (calendar is not empty) {  remove the thread with earliest time stamp from calendar;  change the thread’s priority into MAXPRIO;  }  sched\_yield();  }  } |

Simulation Result:

In the demonstration simulation, there are two customers arriving with inter-arrival time 5 and service time 3, the simulation result is shown below:

