SMPL

Chang-Gun Lee (cglee@snu.ac.kr)

Assistant Professor

The School of Computer Science and Engineering

Seoul National University

Recall

- What to do
  - Generic (system independent)
    - scheduleEvent(event, time)
    - event = getEvent();
    - random number generators
  - System-specific
    - Define events
    - Define the operation steps for each event
    - Determine the most appropriate data structure
- The generic part can be pre-built as a library
  - SMPL provides such a library

SMPL

- SMPL is a set of C functions for building event-based, discrete-event simulation models
- SMPL was written by M.H. MacDougall and is described in Simulating Computer Systems, Techniques and Tools, The MIT Press, 1987

SMPL Entities

- Facilities
- Tokens
- Events

Facilities

- A facility typically represents some work-performing resource of system being modeled, such as CPU and memory in computer system, BUS in computer networks and lock in operating system
- SMPL provides functions to define *facilities*, *reserve*, *release*, and *preempt* them and *interrogate* their status
- The Interconnection of facilities is not explicit, but can be determined by the model's routing of tokens between facilities
- A system comprises a collection of interconnected facilities

Tokens

- Tokens represents the active entities of the system
- The dynamic behavior of the system is modeled by the movement of tokens through set of facilities
- A token may represent a task in a computer system model, a packet in communication model or memory access in a memory bus subsystem model
- In SMPL a token may reserve (preempt) a facility or schedule activity of various duration
- A token can be a single integer (customer id), an structure (enter time, size, …)

Events

- A change of state of any system entity, active or passive is an event
- Some events are Task arrival, CPU completion interval, Process departure

SMPL functions (for start)

- smpl(m,s)
  - int m; char *s
  - initialize the simulation subsystem for a simulation run (clear SMPL data structure, initialize time to zero)
  - When m=1, SMPL provides an interactive interface to model execution.
- reset()
  - Clear all accumulated measurements (Not time)

SMPL functions (for resources) - 1

- F=facility (s,n)
  - char *s; int n
  - This function creates and names a facility with n servers
- R=request(f,tkn,pri)
  - int f, tkn, pri
  - It requests that a server of facility f be reserved for token tkn with pri priority
  - If the facility is not busy, a server is reserved for the requesting tokens (The first idle server will be chosen). Return 0.
  - Each facility has a queue. When facility is busy a queue entry (i.e., tkn and current event) is constructed for the request. Return 1.
  - A request which initially finds the facility busy is called a blocked request

SMPL functions (for resources) - 2

- R=preempt(f,tkn,pri)
  - int f, tkn, pri
  - If the facility is not busy or all the tokens in the busy facility have greater priority, it works like request function
  - If the facility is busy, the server with the lowest priority reserving token is located. If this priority is equal or greater than the requester, the request (current event and token) is queued and 1 is returned
  - Otherwise, the located token is suspended (preempted) and the server will be reserved for requester.
  - The suspended token is put on top of the same priority queue with its remaining time

SMPL functions (for resources) - 3

- release(f,tkn)
  - int f, tkn
  - This function releases the server facility f reserved by token tkn
  - Next, the facility queue is examined and if it isn’t empty the entry at its head is dequeued and associated event (i.e., requestFacility) is resecheduled at the current time
  - If the entry is for a preempted request, the released server is reserved for the dequeued token and the associated event (i.e., releaseFacility) is scheduled to occur at a time equal to the current time plus remaining time

SMPL functions (for statistics)

- N=inq(f)
  - Number of tokens currently in queue
- R=status(f)
  - The facility status (busy or idle)
- U=u(f)
  - Mean facility utilization
- B=B(f)
  - Mean busy period
- L=lq(f)
  - Mean queue length

SMPL functions (for event management)

- schedule(ev, time, tkn)
  - int ev, tkn; real time;
  - This function schedules an event. ev is the event number, time is the inter-event time and tkn is a token associated with event
  - Then an event entry for this event is constructed
- cause(ev, tkn)
  - int *ev, *tkn
  - Removes the entry at the head of the event list, advances the simulation time to the event occurrence time and return the event number ev and token tkn
- Tkn = cancel(ev)
  - int tkn
  - Search in the event list for event ev and remove it from list and return its token

SMPL functions

- t = real time()
  - return the current simulation time
- r = real ranf()
  - returns a psuedo-random number uniformly distributed in the range 0, 1
- i=stream(n)
  - int n
  - select a stream (seed) (1<= n <= 15)
  - or identify the selected stream (if n=0)
- r=expntl(x)
- r=erlang(x,s)
- r=hyperx(x,s)
- r=uniform(real a, real b)
- k=ranfom(int i, int j)
- r=normal(x,s)

Simulating M/M/1 with SMPL

```
#include <stdio.h>      // Needed for printf()
#include "smpl.h"       // Needed for SMPL

void main(void)
{
  real Ta = 200;        // Mean interarrival time
(seconds)
  real Ts = 100;        // Mean service time
  real te = 1.0e6;      // Total simulation time
  int customer = 1;     // Customer id
  int event;  // Event (1 = arrival, 2 = request, 3 =
completion)
  int server;           // Handle for server facility

  // Initialize SMPL subsystem
  smpl(0, "M/M/1 Queue");

  // Initialize server facility (single server)
  server=facility("server", 1);

  // schedule a kick-off event
  schedule(1, 0.0, customer);
```

```
while (time() < te)
  {
    cause(&event,&customer);
    switch(event)
    {
     case 1:  // *** Arrival
       schedule(2, 0.0, customer);
       schedule(1, expntl(Ta), customer);
       break;

     case 2:  // *** Request Server
       if (request(server, customer, 0) ==  0)
         schedule(3, expntl(Ts), customer);
       break;

     case 3:  // *** Release server
       release(server, customer);
       break;
    }
  }
  report();
}
```

Homework 2

- Draw the following three graphs by SMPL simulation
  - traffic load (0-0.95) vs. average # of jobs in the system
  - traffic load (0-0.95) vs. average queue length
  - traffic load (0-0.95) vs. average queueing delay for each job
- Compare the three graphs with the graphs of Homework 1
- When simulating the system with a traffic load 1.0 and 1.5,
  - Observe what happens in your simulation
  - Discuss how to handle such situation