```
/* simulate an M/M/1 queue
  (an open queue with exponential service times and interarrival intervals)
*/

#include "csim.h"
#include <stdio.h>

#define SVTM 1.0 /*mean of service time distribution */
#define IATM 2.0 /*mean of inter-arrival time distribution */
#define NARS 5000 /*number of arrivals to be simulated*/

FACILITY f; /*pointer for facility */
EVENT done; /*pointer for counter */
TABLE tbl; /*pointer for table */
QTABLE qtbl; /*pointer for qhistogram */
int cnt; /*number of active tasks*/
void cust();
void theory();

void sim() /*1st process - named sim */
{
int i;

set_model_name("M/M/1 Queue");
create("sim"); /*required create statement*/

f = facility("facility"); /*initialize facility*/
done = event("done"); /*initialize event*/
tbl = table("resp tms"); /*initialize table */
qtbl = qhistogram("num in sys", 10l); /*initialize qhistogram*/

cnt = NARS; /*initialize cnt*/
for(i = 1; i <= NARS; i++) {
hold(expntl(IATM)); /*hold interarrival*/
cust(); /*initiate process cust*/
}
wait(done); /*wait until all done*/
report(); /*print report*/
theory(); /*print theoretical res*/
mdlstat();
}

void cust() /*process customer*/
{
double t1;

create("cust"); /*required create statement*/

t1 = clock; /*time of request */
note_entry(qtbl); /*note arrival */
reserve(f); /*reserve facility f*/
hold(expntl(SVTM)); /*hold service time*/
release(f); /*release facility f*/
record(clock-t1, tbl); /*record response time*/
```

```
note_exit(qtbl); /*note departure */
cnt--; /*decrement cnt*/
if(cnt == 0)
set(done); /*if last arrival, signal*/
}

void theory() /*print theoretical results*/
{
double rho, nbar, rtime, tput;

printf("\n\n\n\t\t\tM/M/1 Theoretical Results\n");

tput = 1.0/IATM;
rho = tput*SVTM;
nbar = rho/(1.0 - rho);
rtime = SVTM/(1.0 - rho);

printf("\n\n");
printf("\t\tInter-arrival time = %10.3f\n",IATM);
printf("\t\tService time       = %10.3f\n",SVTM);
printf("\t\tUtilization        = %10.3f\n",rho);
printf("\t\tThroughput rate    = %10.3f\n",tput);
printf("\t\tMn nbr at queue    = %10.3f\n",nbar);
printf("\t\tMn queue length    = %10.3f\n",nbar-rho);
printf("\t\tResponse time      = %10.3f\n",rtime);
printf("\t\tTime in queue      = %10.3f\n",rtime - SVTM);
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

M/M/1 Queue

Mon Feb 15 17:35:00 2010

Ending simulation time: 10041.661
Elapsed simulation time: 10041.661
CPU time used (seconds): 0.140

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
facility	fcfs	0.99206	0.494	0.49793	0.99059	1.98943	5000

TABLE 1: resp tms

minimum	0.000145	mean	1.989433
maximum	14.273079	variance	3.813342
range	14.272934	standard deviation	1.952778
observations	5000	coefficient of var	0.981575

QTABLE 1: num in sys

initial	0	minimum	0	mean	0.990590
final	0	maximum	13	variance	1.937727
entries	5000	range	13	standard deviation	1.392022
exits	5000			coeff of variation	1.405246

	number	total time	proportion	cumulative proportion	
	0	5081.38161	0.506030	0.506030	**********
	1	2426.95194	0.241688	0.747718	**********
	2	1238.22169	0.123308	0.871027	*****
	3	667.95025	0.066518	0.937545	***
	4	350.00001	0.034855	0.972399	*
	5	152.62571	0.015199	0.987599	*
	6	69.33696	0.006905	0.994504	.
	7	25.09331	0.002499	0.997003	.
	8	9.84005	0.000980	0.997982	.
	9	10.69388	0.001065	0.999047	.
>=	10	9.56521	0.000953	1.000000	.

```
M/M/1 Theoretical Results


Inter-arrival time =      2.000
Service time       =      1.000
Utilization        =      0.500
Throughput rate    =      0.500
Mn nbr at queue    =      1.000
Mn queue length    =      0.500
Response time      =      2.000
Time in queue      =      1.000
CSIM MODEL STATISTICS

CPU time used (sec):         0.140
Events processed:            17485
Memory allocated:            17392 bytes
Calls to malloc:                47
Processes
     Started:                 5001
     Saved:                   9105
     Terminated:              5000
     High water mark:           14
Stacks
     Allocated:               5015
     High water mark:         1832 words
     Average:                  130 words
     Maximum:                  131 words
     Current:                  129 words
```

```
/* simulate an M/M/1 queue
  (an open queue with exponential service times and interarrival intervals)
*/

#include "csim.h"
#include <stdio.h>

#define SVTM 1.0 /*mean of service time distribution */
#define IATM 2.0 /*mean of inter-arrival time distribution */
#define NARS 5000 /*number of arrivals to be simulated*/

FACILITY f; /*pointer for facility */
EVENT done; /*pointer for counter */
CLASS sim_cl; /*process class for sim */
CLASS cust_cl; /*process class for cust */
TABLE tbl; /*pointer for table */
QTABLE qtbl; /*pointer for qhistogram */
int cnt; /*number of active tasks*/
void cust();
void theory();

void sim() /*1st process - named sim */
{
int i;

set_model_name("M/M/1 Queue");
create("sim"); /*required create statement*/

f = facility("facility"); /*initialize facility*/
done = event("done"); /*initialize event*/
sim_cl = process_class("sim"); /*initialize sim process class*/
cust_cl = process_class("customer"); /*initialize cust process*/
tbl = table("resp tms"); /*initialize table */
qtbl = qhistogram("num in sys", 10l); /*initialize qhistogram*/
collect_class_facility(f); /*turn on class stats for f*/
set_process_class(sim_cl);

cnt = NARS; /*initialize cnt*/
for(i = 1; i <= NARS; i++) {
hold(expntl(IATM)); /*hold interarrival*/
cust(); /*initiate process cust*/
}
wait(done); /*wait until all done*/
report(); /*print report*/
theory(); /*print theoretical res*/
mdlstat();
}

void cust() /*process customer*/
{
double t1;

create("cust"); /*required create statement*/
set_process_class(cust_cl); /*put this process in cust_cl*/
```

```
t1 = clock; /*time of request */
note_entry(qtbl); /*note arrival */
reserve(f); /*reserve facility f*/
hold(expntl(SVTM)); /*hold service time*/
release(f); /*release facility f*/
record(clock-t1, tbl); /*record response time*/
note_exit(qtbl); /*note departure */
cnt--; /*decrement cnt*/
if(cnt == 0)
set(done); /*if last arrival, signal*/
}

void theory() /*print theoretical results*/
{
double rho, nbar, rtime, tput;

printf("\n\n\n\t\t\tM/M/1 Theoretical Results\n");

tput = 1.0/IATM;
rho = tput*SVTM;
nbar = rho/(1.0 - rho);
rtime = SVTM/(1.0 - rho);

printf("\n\n");
printf("\t\tInter-arrival time = %10.3f\n",IATM);
printf("\t\tService time       = %10.3f\n",SVTM);
printf("\t\tUtilization        = %10.3f\n",rho);
printf("\t\tThroughput rate    = %10.3f\n",tput);
printf("\t\tMn nbr at queue    = %10.3f\n",nbar);
printf("\t\tMn queue length    = %10.3f\n",nbar-rho);
printf("\t\tResponse time      = %10.3f\n",rtime);
printf("\t\tTime in queue      = %10.3f\n",rtime - SVTM);
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

M/M/1 Queue

Mon Feb 15 17:37:58 2010

Ending simulation time: 10041.661
Elapsed simulation time: 10041.661
CPU time used (seconds): 0.150

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
facility	fcfs	0.99206	0.494	0.49793	0.99059	1.98943	5000
> class customer		0.99206	0.494	0.49793	0.99059	1.98943	5000

PROCESS CLASS SUMMARY

id	name	number	lifetime	hold count	hold time	wait time
0	default	1	0.00000	0.00000	0.00000	0.00000
1	sim	5001	2.00793	0.99980	2.00791	0.00002
2	customer	5000	1.98943	1.00000	0.99206	0.99738

TABLE 1: resp tms

minimum	0.000145	mean	1.989433
maximum	14.273079	variance	3.813342
range	14.272934	standard deviation	1.952778
observations	5000	coefficient of var	0.981575

QTABLE 1: num in sys

initial	0	minimum	0	mean	0.990590
final	0	maximum	13	variance	1.937727
entries	5000	range	13	standard deviation	1.392022
exits	5000			coeff of variation	1.405246

number	total time	proportion	cumulative proportion	
0	5081.38161	0.506030	0.506030	**********
1	2426.95194	0.241688	0.747718	**********
2	1238.22169	0.123308	0.871027	*****
3	667.95025	0.066518	0.937545	***
4	350.00001	0.034855	0.972399	*
5	152.62571	0.015199	0.987599	*

```
             6      69.33696     0.006905     0.994504   .
             7      25.09331     0.002499     0.997003   .
             8       9.84005     0.000980     0.997982   .
             9      10.69388     0.001065     0.999047   .
 >=         10       9.56521     0.000953     1.000000   .



M/M/1 Theoretical Results


Inter-arrival time =      2.000
Service time       =      1.000
Utilization        =      0.500
Throughput rate    =      0.500
Mn nbr at queue    =      1.000
Mn queue length    =      0.500
Response time      =      2.000
Time in queue      =      1.000
CSIM MODEL STATISTICS

CPU time used (sec):         0.160
Events processed:            17485
Memory allocated:            17720 bytes
Calls to malloc:                64
Processes
     Started:                 5001
     Saved:                   9105
     Terminated:              5000
     High water mark:           14
Stacks
     Allocated:               5015
     High water mark:         1832 words
     Average:                  130 words
     Maximum:                  131 words
     Current:                  129 words
```

```
/* illustrate use of process classes and simulate an M/M/1 queue
  (an open queue with exponential service times and interarrival intervals)
*/

#include "csim.h"
#include <stdio.h>

#define SVTM 1.0 /*mean of service time distribution */
#define IATM 2.0 /*mean of inter-arrival time distribution */
#define NARS 5000 /*number of arrivals to be simulated*/

FACILITY f; /*pointer for facility */
EVENT done; /*pointer for counter */
TABLE tbl; /*pointer for table */
QTABLE qtbl; /*pointer for qhistogram */
CLASS cl[2]; /* pointers to classes */
int cnt; /*number of active tasks*/
FILE *fp;
void cust();
void make_facility_report();

void sim() /*1st process - named sim */
{
int i;

fp = fopen("xxx.out", "w");
set_output_file(fp);
set_model_name("M/M/1 Queue");
create("sim"); /*required create statement*/

f = facility_ms("facility", 2l); /*initialize facility*/
collect_class_facility(f); /*collect class statistics*/
done = event("done"); /*initialize event*/
tbl = table("resp tms"); /*initialize table */
qtbl = qhistogram("num in sys", 10l); /*initialize qhistogram*/
cl[0] = process_class("class 0"); /*initialize cust class */
cl[1] = process_class("class 1");

cnt = NARS; /*initialize cnt*/
for(i = 1; i <= NARS; i++) {
hold(expntl(IATM)); /*hold interarrival*/
cust(); /*initiate process cust*/
}
wait(done); /*wait until all done*/
report(); /*print report*/
make_facility_report(f);
}

void cust() /*process customer*/
{
double t1;

create("cust"); /*required create statement*/
set_process_class(cl[random(0l,1l)]);
```

```
t1 = clock; /*time of request */
note_entry(qtbl); /*note arrival */
reserve(f); /*reserve facility f*/
hold(expntl(SVTM)); /*hold service time*/
release(f); /*release facility f*/
record(clock-t1, tbl); /*record response time*/
note_exit(qtbl); /*note departure */
cnt--; /*decrement cnt*/
if(cnt == 0)
set(done); /*if last arrival, signal*/
}

void make_facility_report(f)
FACILITY f;
{
long i, n;

fprintf(fp, "\n\tfacility %s\n", facility_name(f));
fprintf(fp, "service disp    %s\n", service_disp(f));
fprintf(fp, "service time    %.3f\n", serv(f));
fprintf(fp, "utilization     %.3f\n", util(f));
fprintf(fp, "throughput      %.3f\n", tput(f));
fprintf(fp, "queue length    %.3f\n", qlen(f));
fprintf(fp, "response time   %.3f\n", resp(f));
fprintf(fp, "completions     %ld\n",  completions(f));
fprintf(fp, "preempts        %ld\n", preempts(f));
n = num_servers(f);
fprintf(fp, "num servers     %d\n", n);
if(n > 1) {
for(i = 0; i < n; i++) {
fprintf(fp, "\nserver %ld\n", i);
fprintf(fp, "    service time    %.3f\n",
server_serv(f, i));
fprintf(fp, "    utilization     %.3f\n",
server_util(f, i));
fprintf(fp, "    throughput      %.3f\n",
server_tput(f, i));
fprintf(fp, "    completions     %ld\n",
server_completions(f, i));
}
}
for(i = 0; i < 2; i++) {
fprintf(fp, "\nclass %ld\n", i);
fprintf(fp, "    service time     %.3f\n", class_serv(f, cl[i]));
fprintf(fp, "    utilization      %.3f\n", class_util(f, cl[i]));
fprintf(fp, "    throughput       %.3f\n", class_tput(f, cl[i]));
fprintf(fp, "    queue length     %.3f\n", class_qlen(f, cl[i]));
fprintf(fp, "    response time    %.3f\n", class_resp(f, cl[i]));
fprintf(fp, "    completions      %ld\n", class_completions(f, cl[i]));
}
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

M/M/1 Queue

Mon Feb 15 17:39:52 2010

Ending simulation time: 9865.713
Elapsed simulation time: 9865.713
CPU time used (seconds): 0.140

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
facility	fcfs	1.01112	0.512	0.50681	0.55222	1.08960	5000
> server 0		1.00237	0.355	0.35436			3496
> server 1		1.03146	0.157	0.15245			1504
> class class 0		1.01134	0.254	0.25077	0.27304	1.08881	2474
> class class 1		1.01091	0.259	0.25604	0.27918	1.09038	2526

PROCESS CLASS SUMMARY

id	name	number	lifetime	hold count	hold time	wait time
0	default	5001	1.97275	0.99980	1.97241	0.00034
1	class 0	2474	1.08881	1.00000	1.01134	0.07747
2	class 1	2526	1.09038	1.00000	1.01091	0.07947

TABLE 1: resp tms

minimum	0.000145	mean	1.089604
maximum	10.791286	variance	1.181387
range	10.791142	standard deviation	1.086916
observations	5000	coefficient of var	0.997533

QTABLE 1: num in sys

initial	0	minimum	0	mean	0.552218
final	0	maximum	8	variance	0.648783
entries	5000	range	8	standard deviation	0.805471
exits	5000			coeff of variation	1.458611

number	total time	proportion	cumulative proportion	
0	5852.15006	0.593181	0.593181	**********
1	2971.51407	0.301196	0.894377	**********
2	763.59921	0.077399	0.971776	***

```
            3     195.60913     0.019827     0.991603   *
            4      58.82568     0.005963     0.997566   .
            5      18.28632     0.001854     0.999419   .
            6       4.52843     0.000459     0.999878   .
            7       1.02624     0.000104     0.999982   .
            8       0.17391     0.000018     1.000000   .


facility facility
service disp    fcfs
service time    1.011
utilization     0.512
throughput      0.507
queue length    0.552
response time   1.090
completions     5000
preempts        0
num servers     2

server 0
    service time    1.002
    utilization     0.355
    throughput      0.354
    completions     3496

server 1
    service time    1.031
    utilization     0.157
    throughput      0.152
    completions     1504

class 0
    service time     1.011
    utilization      0.254
    throughput       0.251
    queue length     0.273
    response time    1.089
    completions      2474

class 1
    service time     1.011
    utilization      0.259
    throughput       0.256
    queue length     0.279
    response time    1.090
    completions      2526
```

```
/* example of preempt-resume service discipline */

#include "csim.h"

FACILITY f;
void job();

void sim()
{
int i;

create("sim");
f = facility_ms("f", 21);
set_servicefunc(f, pre_res);
for(i = 0; i < 1000; i++) {
job(i);
hold(expntl(2.0));
}
wait(event_list_empty);
report();
}

void job(n)
int n;
{
create("job");
set_priority((long)n);
use(f, expntl(1.5));
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

Mon Feb 15 17:52:15 2010

Ending simulation time: 1957.940
Elapsed simulation time: 1957.940
CPU time used (seconds): 0.020

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
f	pre_res	1.54317	0.788	0.51074	0.92397	1.80908	1000
> server	0	1.48727	0.490	0.32943			645
> server	1	1.64473	0.298	0.18131			355

```
/* round robin service without priorities */

#include "csim.h"

FACILITY f;
TABLE st;

void high_task();
void low_task();

void sim()
{
int i;

create("sim");

f = facility("f");
set_servicefunc(f, rnd_rob);
set_timeslice(f, 0.5);
st = table("serv tm");

high_task();
for(i = 0; i < 3; i++)
low_task();
wait(event_list_empty);
report();
}

void high_task()
{
int i;

create("high");
set_priority(2L);
for(i = 0; i < 5; i++) {
hold(1.0);
use(f, 1.0);
record(1.0, st);
}
}

void low_task()
{
int i;

create("low");
set_priority(1L);
for(i = 0; i < 5; i++) {
use(f, 1.0);
record(1.0, st);
hold(1.0);
}
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

Mon Feb 15 17:47:17 2010

Ending simulation time: 21.000
Elapsed simulation time: 21.000
CPU time used (seconds): 0.000

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
f	rnd_rob	0.50000	0.952	1.90476	2.54762	1.33750	40

TABLE 1: serv tm

minimum	1.000000	mean	1.000000
maximum	1.000000	variance	0.000000
range	0.000000	standard deviation	0.000000
observations	20	coefficient of var	0.000000

```
/* round robin service with priorities */

#include "csim.h"

FACILITY f;
TABLE st;

void high_task();
void low_task();

void sim()
{
int i;

create("sim");

f = facility("f");
set_servicefunc(f, rnd_pri);
set_timeslice(f, 0.5);
st = table("serv tm");

high_task();
for(i = 0; i < 3; i++)
low_task();
wait(event_list_empty);
report();
}

void high_task()
{
int i;

create("high");
set_priority(2l);
for(i = 0; i < 5; i++) {
hold(1.0);
use(f, 1.0);
record(1.0, st);
}
}

void low_task()
{
int i;

create("low");
set_priority(1l);
for(i = 0; i < 5; i++) {
use(f, 1.0);
record(1.0, st);
hold(1.0);
}
}
```

CSIM Simulation Report (Version 19.0 for SPARC Solaris)

Mon Feb 15 17:51:13 2010

Ending simulation time: 21.000
Elapsed simulation time: 21.000
CPU time used (seconds): 0.000

FACILITY SUMMARY

facility name	service disc	service time	util.	through-put	queue length	response time	compl count
f	rnd_pri	0.57143	0.952	1.66667	2.42857	1.45714	35

TABLE 1: serv tm

minimum	1.000000	mean	1.000000
maximum	1.000000	variance	0.000000
range	0.000000	standard deviation	0.000000
observations	20	coefficient of var	0.000000

```
/* client-server: test out mailbox and messages */

#include "csim.h"
#include <stdio.h>

#define SIMTIME 10.0
#define NS 2l
#define NC 4l

#define SERV_REQ 1l
#define ACK 2l
#define QUIT 3l
#define QUIT_ACK 4l

typedef struct msg *msg_t;

struct msg {
MBOX reply;
long type;
};

MBOX serv_mb[NS];

char str[24];
long msg_ct, msg_cur;

void server();
void client();

msg_t new_reply();
msg_t new_send();
void stop_servers();
void my_error();
void del_msg();
void msg_rep();

void sim()
{
long i;

create("sim");
for(i = 0; i < NS; i++)
serv_mb[i] = mailbox("serv");
msg_ct = msg_cur = 0;
for(i = 0; i < NS; i++)
server(i);
for(i = 0; i < NC; i++)
client(i);
wait(event_list_empty);
stop_servers();
msg_rep();
}

void server(n)
```

```
long n;
{
msg_t r;
msg_t s;
long type;

create("server");
do {
receive(serv_mb[n], &r);
hold(1.0);
   printf("server.%ld sends back to mb %s\n",
n, mailbox_name(r->reply));
switch(r->type) {
case SERV_REQ:
type = ACK;
break;
case QUIT:
type = QUIT_ACK;
break;
default:
my_error("server %ld: unexpected msg type", n);
}
s = new_reply(n, type);
send(r->reply, s);
del_msg(r);
} while(type != QUIT);
}

void client(n)
long n;
{
MBOX recv;
msg_t s;
msg_t r;
long i;

create("client");
sprintf(str, "cli.%ld", n);
recv = mailbox(str);
while(clock < SIMTIME) {
s = new_send(n, SERV_REQ, recv);
i = random(0l, NS-1);
printf("client.%ld sends to server.%ld\n",
n, i);
send(serv_mb[i], s);
receive(recv, &r);
del_msg(r);
}
delete_mailbox(recv);
}

void stop_servers()
{
long i;
```

```
msg_t s;
msg_t r;
MBOX recv;

recv = mailbox("stop");
for(i = 0; i < NS; i++) {
s = new_send(i, QUIT, recv);
send(serv_mb[i], s);
}
for(i = 0; i < NS; i++) {
receive(recv, &r);
del_msg(r);
}
delete_mailbox(recv);
}

void my_error(f, n)
char *f; long n;
{
printf(f,n);
printf("\n");
exit(1);
}

msg_t new_send(n, t, m)
long n, t; MBOX m;
{
msg_t msg;

msg = (msg_t) do_malloc(sizeof(struct msg));
msg->reply = m;
msg->type = t;
msg_ct++;
msg_cur++;
return(msg);
}

msg_t new_reply(n, t) long n, t;
{
msg_t msg;

msg = (msg_t) do_malloc(sizeof(struct msg));
msg->type = t;
msg_ct++;
msg_cur++;
return(msg);
}

void del_msg(msg)
msg_t msg;
{
msg_cur--;
free((char*) msg);
```

```
}

void msg_rep()
{
printf("messages created %4d\n", msg_ct);
printf("current messages %4d\n", msg_cur);
}
```

```
client.0 sends to server.1
client.1 sends to server.0
client.2 sends to server.0
client.3 sends to server.1
server.1 sends back to mb cli.0
server.0 sends back to mb cli.1
client.0 sends to server.1
client.1 sends to server.0
server.1 sends back to mb cli.3
server.0 sends back to mb cli.2
client.3 sends to server.1
client.2 sends to server.0
server.1 sends back to mb cli.0
server.0 sends back to mb cli.1
client.0 sends to server.0
client.1 sends to server.0
server.1 sends back to mb cli.3
server.0 sends back to mb cli.2
client.3 sends to server.0
client.2 sends to server.0

server.0 sends back to mb cli.0
client.0 sends to server.0
server.0 sends back to mb cli.1
client.1 sends to server.0
server.0 sends back to mb cli.3
client.3 sends to server.1
server.0 sends back to mb cli.2
server.1 sends back to mb cli.3
client.2 sends to server.1
client.3 sends to server.1
server.0 sends back to mb cli.0
server.1 sends back to mb cli.2
client.0 sends to server.1
client.2 sends to server.1
server.0 sends back to mb cli.1
server.1 sends back to mb cli.3
server.1 sends back to mb cli.0
server.1 sends back to mb cli.2
server.0 sends back to mb stop
server.1 sends back to mb stop
messages created   42
current messages    0
```

```
/* simulate two CSIM processes passing control to one another
   One is "sim" and the other is "cust"
*/

#include "csim.h"
#include <stdio.h>

EVENT done;

float busy = 10;

void cust();


void sim() /*1st process - named sim */
{

set_model_name("Two Processes");
create("sim"); /*required create statement*/

        done = event("done");

        printf("\n Thread Sim is born at time %lf\n",simtime());

        printf("\n What happens when we call Cust? Let's find out \n");

        printf("\n Thread Sim is yielding control to Thread Cust at time %lf \n",simtime());

        cust();

        printf("\n Thread Sim regains control from Cust at time %lf \n",simtime());

        printf("\n Thread Sim is going to hold for %lf units at time %lf\n",busy,simtime());

        hold (busy);

        printf("\n Thread Sim is back again at time %lf\n",simtime());

        printf("\n Thread Sim waits for Thread Cust to set a flag\n");

        wait(done);

        printf("\n Thread Sim is now leaving the system\n");
}


void cust() /*process customer*/
{
create("cust"); /*required create statement*/

        printf("\n Hi There! I am Thread Cust!\n");

        printf("\n I was born at time %lf\n",simtime());
```

```
        printf("\n I am going to hold for %lf units at time %lf\n",busy,simtime());
        hold(busy);
        printf("\n I'm Thread Cust and I'm back at time %lf! \n",simtime());
        printf("\n Set a flag so Thread Sim can leave\n");
        set(done);
        printf("\n I have nothing left to do and so I am leaving\n");
}
```

```
Thread Sim is born at time 0.000000

What happens when we call Cust? Let's find out

Thread Sim is yielding control to Thread Cust at time 0.000000

Thread Sim regains control from Cust at time 0.000000

Thread Sim is going to hold for 10.000000 units at time 0.000000

Hi There! I am Thread Cust!

I was born at time 0.000000

I am going to hold for 10.000000 units at time 0.000000

Thread Sim is back again at time 10.000000

Thread Sim waits for Thread Cust to set a flag

I'm Thread Cust and I'm back at time 10.000000!

Set a flag so Thread Sim can leave

I have nothing left to do and so I am leaving

Thread Sim is now leaving the system
```