Xi Liu, xl3504, hw4

1.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

int i = 0;

/\* ADD SOME THINGS HERE \*/

pthread\_mutex\_t m = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t c = PTHREAD\_COND\_INITIALIZER;

void thread\_exit()

{

pthread\_mutex\_lock(&m);

i = 1;

pthread\_cond\_signal(&c);

pthread\_mutex\_unlock(&m);

}

void

\*foo(void \*arg)

{

printf("I am foo!!!\n");

/\* ADD SOME CODE HERE \*/

thread\_exit();

return NULL;

}

void

\*boo(void \*arg)

{

/\* ADD SOME CODE HERE \*/

pthread\_mutex\_lock(&m);

while(i == 0)

{

pthread\_cond\_wait(&c, &m);

}

pthread\_mutex\_unlock(&m);

printf("I am boo!!!\n");

}

int

main(int argc, char\*\* argv)

{

pthread\_t p1, p2;

pthread\_create(&p1, NULL, &foo, NULL);

pthread\_create(&p2, NULL, &boo, NULL);

// wait for threads to finish

// before exiting

pthread\_join(p1, NULL);

pthread\_join(p2, NULL);

printf("main: end\n");

exit(0);

}

2. 1

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Variable Type | Initial Value | Description |
| Paper | int | 0 | If the smoker has paper, then paper is 1;  otherwise, paper is 0. |
| Tobacco | int | 0 | If the smoker has tobacco, then tobacco is 1;  otherwise, tobacco is 0. |
| Matches | int | 0 | If the smoker has matches, then matches is 1;  otherwise, match is 0. |
| remaining\_ingredient | int | 0 | Before the procedure chooseIngredients() run, remaining\_ingredient is 0, indicates that nothing is selected;  After an execution of chooseIngredients() that randomly selects 2 of the 3 ingredients, if remaining\_ingredient  is 1,  indicates that paper is the remaining ingredient;  if remaining\_ingredient  is 2,  indicates that tobacco is the remaining ingredient;  if remaining\_ingredient  is 3,  indicates that matches is the remaining ingredient; |
| smoker\_i->ingredient | int | 0 | i = 1, 2, 3  smoker->ingredient is the ingredient that the smoker has.  If smoker->ingredient  is 0,  indicates that the smoker process has not been initialized  If smoker->ingredient  is 1,  indicates that the smoker process has paper  If smoker->ingredient  is 2,  indicates that the smoker process has tobacco  If smoker->ingredient  is 3,  indicates that the smoker process has matches |
| smoker\_complete | int | 0 | smoker\_complete is initialized as 1, since no smoker started smoking, they are set to be “complete”.  if a smoker completed smoking, smoker\_complete is immediately set to 1;  before a call to chooseIngredients, smoker\_complete is set to 0, which indicates that the smoker would start to smoke. |
| mutex | pthread\_mutex\_t |  | Mutual exclusion, ensure that only one thread can modifies its state at a given time. |
| c | pthread\_cond\_t |  | Conditional variable,  Used like a queue to communicate between threads. |

2.2

//3 smokers can smoke at the same time

int paper = 0;

int tobacco = 0;

int matches = 0;

int remaining\_ingredient = 0;

int smoker\_complete = 1;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t c = PTHREAD\_COND\_INITIALIZER;

Agent()

{//the agent places 2 of the ingredients on the table

acquire(&mutex);

while(smoker\_complete = = 1)

{

smoker\_complete = 0;

chooseIngredients(&paper, &tobacco, &matches);

if(paper & tobacco)

{

remaining\_ingredient = 3; //matches

}

else if(paper & matches)

{

remaining\_ingredient = 2; //tobacco

}

else if(tobacco & matches)

{

remaining\_ingredient = 1; //paper

}

}

release(&mutex);

}

matchSmoker()

{//this smoker has a lot of matches

acquire(&mutex)

while(remaining\_ingredient != 3)

{

pthread\_cond\_wait(&c, &m);

}

//the smoker has the remaining ingredient which is matches

smoke();

paper = 0;

tobacco = 0;

smoker\_complete = 1;

pthread\_cond\_signal(&c); //signal the agent

release(&mutex);

}

3.

Class WaitingRoom

{

int n\_used = 0; //chair used

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t c = PTHREAD\_COND\_INITIALIZER;

Queue<Integer> q = new Queue<Integer>();

int myturn = 0;

int call = 0;

int enter()

{

acquire(&mutex);

if(n\_used = = NCHAIRS)

{

release(&mutex);

return WR\_FULL;

}

else

{

n\_used++;

call++;

q.enqueue();

while(myturn != call)

{

pthread\_cond\_wait(&c, &mutex);

}

n\_used--;

q.dequeue();

release(&mutex);

return MY\_TURN;

}

}

callNextCustomer()

{

acquire(&mutex);

if(n\_used = = 0)

{

release(&mutex);

return WR\_EMPTY;

}

else

{

pthread\_cond\_signal(&c);

release(&mutex);

return WR\_BUSY;

}

}

}

Class BarberChair

{

int state = EMPTY;

int customer = 0;

pthread\_cond\_t barber\_up = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t customer\_done = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t sit\_in\_chair = PTHREAD\_COND\_INITIALIZER;

void napInChair()

{

acquire(&mutex);

if(costumer = = 0)

{

state = BARBER\_IN\_CHAIR;

//when no customer present, barber sits in chair

//and falls asleep

}

while(customer = = 0)

{

pthread\_cond\_wait(&barber\_up, &mutex);

//barber remains sleeping if no customer

}

release(&mutex);

}

void wakeBarber()

{

acquire(&mutex);

customer = 1;

pthread\_cond\_signal(&barber\_up);

release(&mutex);

}

void sitInChair()

{

acquire(&mutex);

while(state != EMPTY)

{

pthread\_cond\_wait(&sit\_in\_chair, &mutex);

}

if(LONG\_HAIR\_CUSTOMER\_IN\_CHAIR != NULL)

{

state = LONG\_HAIR\_CUSTOMER\_IN\_CHAIR;

}

while(state != SHORT\_HAIR\_CUSTOMER\_IN\_CHAIR)

{

pthread\_cond\_wait(&customer\_done, &mutex);

}

state = EMPTY;

pthread\_cond\_signal(&customer\_done, &mutex);

release(&mutex);

}

void cutHair()

{

acquire(&mutex);

while(state != LONG\_HAIR\_CUSTOMER\_IN\_CHAIR)

{

pthread\_cond\_wait(&customer\_done, &mutex);

}

if(SHORT\_HAIR\_CUSTOMER\_IN\_CHAIR != NULL)

{

state = SHORT\_HAIR\_CUSTOMER\_IN\_CHAIR;

}

release(&mutex);

}

void tellcustomerDone()

{

acquire(&mutex);

while(state != EMPTY)

{

pthread\_cond\_wait(&customer\_done, &mutex);

}

release(&mutex);

}

}