#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <unistd.h>

#define BUFFER\_SIZE 3

#define SHIPPMENT\_WORK 3

typedef struct salesPackage

{

int bufferArr[BUFFER\_SIZE]; //number of items in the buffer

size\_t len; //number of items in buffer

pthread\_mutex\_t mutex; // needed to add/remove data from the buffer

pthread\_cond\_t can\_produce\_caone; //signaled when are removed from my buffer

pthread\_cond\_t can\_produce\_paddle; //signaled when are removed from my buffer

pthread\_cond\_t can\_consume\_produce; //signaled when are added from my buffer

}salesPackage;

void \* makeCanoe(void \* arg);

void \* makePaddle(void \* arg);

void \* makeShipment(void \* arg);

int main(int argc, char \*argv[]){

salesPackage buffer = {

.len = 0,

.mutex = PTHREAD\_MUTEX\_INITIALIZER,

.can\_produce\_caone = PTHREAD\_COND\_INITIALIZER,

.can\_produce\_paddle = PTHREAD\_COND\_INITIALIZER,

.can\_consume\_produce = PTHREAD\_COND\_INITIALIZER

};

int n = atoi(argv[1]);

time\_t t = (time\_t) n;

srand((unsigned) time(&t));

//generating random numbers between 2 and 5 inclusive

int randomTimeForThread = (rand() %4) + 2;

//printf("%d\n", randomTimeForThread);

pthread\_t canoeThread;

pthread\_t paddleThread;

pthread\_t shipperThread;

int x;

for(x = n; x >= 0; x = x-randomTimeForThread){

pthread\_create (&paddleThread, NULL, &makePaddle, (void\*)&buffer);

//this will let the canoe thread wait for the canoe thread to be done

pthread\_join(paddleThread, NULL);

pthread\_create (&canoeThread, NULL, &makeCanoe, (void\*)&buffer);

//this will let the paddle thread wait for the canoe thread to be done

pthread\_join(canoeThread, NULL);

pthread\_create (&shipperThread, NULL, &makeShipment,(void\*)&buffer);

pthread\_join(shipperThread, NULL);

}

if(x == 0){

pthread\_detach(canoeThread);

pthread\_detach(paddleThread);

pthread\_detach(shipperThread);

}

pthread\_exit(NULL);

return 0;

}

void \* makePaddle(void \* arg){

salesPackage \*buffer = (salesPackage\*)arg;

while(1){

#ifdef UNDERFLOW

// used to show that if the producer is somewhat "slow" the consumer will not fail (i.e. it'll just wait for new items to consume)

sleep(rand() % 3);

#endif

pthread\_mutex\_lock(&buffer->mutex);

if(buffer->len == BUFFER\_SIZE){

//if full wait till all is consumed

pthread\_cond\_signal(&buffer->can\_consume\_produce);

makeShipment(buffer);

pthread\_cond\_wait(&buffer->can\_produce\_paddle, &buffer->mutex);

pthread\_mutex\_unlock(&buffer->mutex);

//if buffer is size 0 or 1 produce paddles

} else if(buffer->len == 2){

pthread\_cond\_signal(&buffer->can\_produce\_caone);

makeCanoe(buffer);

pthread\_cond\_wait(&buffer->can\_produce\_paddle, &buffer->mutex);

}

printf("We have a paddle\n");

buffer->bufferArr[buffer->len] = 1;

buffer->len++;

pthread\_mutex\_unlock(&buffer->mutex);

}

return 0;

}

void \* makeCanoe(void \* arg){

salesPackage \*buffer = (salesPackage\*)arg;

while(1){

#ifdef UNDERFLOW

// used to show that if the producer is somewhat "slow" the consumer will not fail (i.e. it'll just wait for new items to consume)

sleep(rand() % 3);

#endif

pthread\_mutex\_unlock(&buffer->mutex);

printf("We have a canoe\n");

buffer->bufferArr[buffer->len] = 1;

buffer->len++;

if(buffer->len == BUFFER\_SIZE){

//if full wait till all is consumed

pthread\_cond\_signal(&buffer->can\_consume\_produce);

makeShipment(buffer);

pthread\_cond\_wait(&buffer->can\_produce\_caone, &buffer->mutex);

pthread\_mutex\_unlock(&buffer->mutex);

//if buffer is size 0 or 1 produce paddles

} else if(buffer->len == 1 || buffer->len == 0){

pthread\_cond\_signal(&buffer->can\_produce\_paddle);

makePaddle(buffer);

pthread\_cond\_wait(&buffer->can\_produce\_caone, &buffer->mutex);

}

}

//pthread\_mutex\_unlock(&buffer->mutex);

return 0;

}

void \* makeShipment(void \* arg){

salesPackage \*buffer = (salesPackage\*)arg;

while(1){

#ifdef OVERFLOW

// showing that the buffer won't overflow if the consumer is slow (i.e. the producer will wait)

sleep(rand() % 3);

#endif

pthread\_mutex\_unlock(&buffer->mutex);

if(buffer->len == 0|| buffer->len == 1){

pthread\_cond\_signal(&buffer->can\_produce\_paddle);

makePaddle(buffer);

pthread\_cond\_wait(&buffer->can\_consume\_produce, &buffer->mutex);

} else if (buffer->len == 2){

pthread\_cond\_signal(&buffer->can\_produce\_caone);

makeCanoe(buffer);

pthread\_cond\_wait(&buffer->can\_consume\_produce, &buffer->mutex);

}else{

for (int i = 0; i<SHIPPMENT\_WORK; i++){

--buffer->len;

}

}

printf("We have a shipment\n");

//pthread\_mutex\_unlock(&buffer->mutex);

makePaddle(buffer);

pthread\_cond\_signal(&buffer->can\_produce\_paddle);

}

return 0;

}

Text

Description automatically generated