CS 246-LAB12

By Julia Sheng

2018/4/22

**Code**

\\Customer.h

typedef struct{

char \*id;

int arrivalTime;

int serviceTime;

}Customer;

Customer \*newCustomer(int arrTime);

char \*getCustId(const Customer \*c);

int getArrivalTime(const Customer \*c);

int getServiceTime(const Customer \*c);

char \*custToString(const Customer \*c);

\\Customer.c

#include "Customer.h"

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

static int count=0;

char \*genCustId(){

char \*id=malloc(5\*sizeof(char));

count++;

sprintf(id,"C%d",count);

return id;

}

Customer \*newCustomer(int at){

Customer \*c=malloc(sizeof(Customer));

c->arrivalTime=at;

c->serviceTime=5+rand()%4;

c->id=genCustId();

return c;

}

char \*getCustId(const Customer \*c){

return c->id;

}

int getArrivalTime(const Customer \*c){

return c->arrivalTime;

}

int getServiceTime(const Customer \*c){

return c->serviceTime;

}

char \*custToString(const Customer \*c){

char \*result=malloc(20\*sizeof(char));

sprintf(result,"<%s: %d, %d>",c->id,c->arrivalTime,c->serviceTime);

return result;

}

\\testCustomer.c

#include<stdio.h>

#include "Customer.h"

int main(void){

Customer \*cs[5];

int t=23;

for(int i=0;i<5;i++){

cs[i]=newCustomer(t);

t++;

}

for(int i=0;i<5;i++){

printf("<%s: %d, %d>\n",cs[i]->id,cs[i]->arrivalTime,cs[i]->serviceTime);

printf("Customer id: %s, Arr: %d, Ser: %d\n",getCustId(cs[i]),getArrivalTime(cs[i]),getServiceTime(cs[i]));

printf("%s\n",custToString(cs[i]));

printf("----------------------\n");

}

return 0;

}

\\Kiosk.h

#include "Customer.h"

typedef struct{

char \*id;

Customer \*nowServing;

int timeRemaining;

}Kiosk;

Kiosk \*newKiosk();

char \*getKioskId(const Kiosk \*k);

Customer \*getCustomer(const Kiosk \*k);

void assignCustomer(Kiosk \*k, Customer \*c);

void tick(Kiosk \*k);

int busy(const Kiosk \*k);

int getNServed();

char \*kioskToString(const Kiosk \*k);

\\Kiosk.c

#include "kiosk.h"

#include <stdio.h>

#include <stdlib.h>

static int count=0;

static int nServed=0;

char \*genkId(){

char \*id=malloc(5\*sizeof(char));

count++;

sprintf(id,"K%d",count);

return id;

}

Kiosk \*newKiosk(){

Kiosk \*k=malloc(sizeof(Kiosk));

k->id=genkId();

k->nowServing=NULL;

k->timeRemaining=0;

return k;

}

char \*getKioskId(const Kiosk \*k){

return k->id;

}

Customer \*getCustomer(const Kiosk \*k){

return k->nowServing;

}

void assignCustomer(Kiosk \*k, Customer \*c){

k->nowServing=c;

k->timeRemaining=c->serviceTime;

}

void tick(Kiosk \*k){

k->timeRemaining--;

if(k->timeRemaining==0){

nServed++;

k->nowServing=NULL;

}

}

int busy(const Kiosk \*k){

return (k->timeRemaining!=0);

}

int getNServed(){

return nServed;

}

char \*kioskToString(const Kiosk \*k){

char \*result=malloc(20\*sizeof(char));

sprintf(result,"%s: %s [%d]",k->id,(k->nowServing?custToString(k->nowServing):"NULL"),k->timeRemaining);

return result;

}

\\testKiosk.c

#include <stdio.h>

#include "kiosk.h"

int main(void){

Kiosk \*ks[5];

Customer \*cs[5];

int t=23;

for(int i=0;i<5;i++){

ks[i]=newKiosk();

cs[i]=newCustomer(t);

t++;

}

for(int i=0;i<5;i++){

printf("<%s: %d\n",ks[i]->id,ks[i]->timeRemaining);

printf ("%s\n",kioskToString(ks[i]));

}

for(int i=0;i<5;i++){

assignCustomer(ks[i], cs[i]);

printf("<%s: %d\n",ks[i]->id,ks[i]->timeRemaining);

printf ("%s\n",kioskToString(ks[i]));

}

while(busy(ks[0])){

tick(ks[0]);

}

printf("Get %d served.\n",getNServed());

return 0;

}

\\queue.h

#include "List.h"

typedef struct{

List \*L;

}Queue;

Queue \*newQueue();

int sizeQ(const Queue \*q);

int emptyQ(const Queue \*q);

void insertQ(Queue \*q,void\* item);

void \*removeQ(Queue \*q);

\\queue.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "queue.h"

Queue \*newQueue(){

Queue \*Q = malloc(sizeof(Queue));

List \*l= malloc(sizeof(List));

\*l=newList();

Q->L=l;

return Q;

}

int sizeQ(const Queue \*q){

return size(q->L);

}

int emptyQ(const Queue \*q){

return empty(q->L);

}

void insertQ(Queue \*q,void\* item){

struct node \*n=newNode(item);

if(q->L->size==0)

q->L->head=q->L->tail=n;

else{

q->L->tail->next=n;

q->L->tail=n;

}

q->L->size++;

}

void \*removeQ(Queue \*q){

if (q->L->head==NULL){

printf("Unable to remove.\n");

exit(EXIT\_FAILURE);

}

else{

void\* result=q->L->head->data;

struct node \*nxt=q->L->head->next;

free(q->L->head);

q->L->head=nxt;

q->L->size--;

return result;

}

}

\\testq.c

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include "queue.h"

int main(void){

Queue \*Q;

Q=newQueue();

printf("Q Size=%d\n",sizeQ(Q));

if(emptyQ(Q))

printf("The queue is empty.\n");

else

printf("The queue is not empty.\n");

int \*n;

printf("Adding: ");

for(int i=0;i<10;i++){

n=malloc(sizeof(int));

\*n=rand()%100;

printf("%d ",\*n);

insertQ(Q,n);

}

printf("\n");

printf("Q Size=%d\n",sizeQ(Q));

if(emptyQ(Q))

printf("The queue is empty.\n");

else

printf("The queue is not empty.\n");

for(int i=0;i<8;i++){

n=removeQ(Q);

printf("Removed: %d\n",\*n);

}

printf("Q Size=%d\n",sizeQ(Q));

if(emptyQ(Q))

printf("The queue is empty.\n");

else

printf("The queue is not empty.\n");

return 0;

}

\\Makefile

testc: testCustomer.o Customer.o

gcc -o testc testCustomer.o Customer.o

testk: testKiosk.o Customer.o kiosk.o

gcc -o testk testKiosk.o Customer.o kiosk.o

testq: testQ.o queue.o List.o node.o

gcc -o testq testQ.o List.o queue.o node.o

testCustomer.o: testCustomer.c Customer.h

gcc -c testCustomer.c

testKiosk.o: testKiosk.c kiosk.h

gcc -c testKiosk.c

testQ.o: testQ.c queue.h

gcc -c testQ.c

Customer.o: Customer.c Customer.h

gcc -c Customer.c

kiosk.o: kiosk.c kiosk.h

gcc -c kiosk.c

queue.o: queue.c queue.h

gcc -c queue.c

List.o: List.h List.c

gcc -c List.c

node.o: node.h node.c

gcc -c node.c

\\codes for List and Node are not changed, thus I did not print them.

**Output**

[jsheng@powerpuff lab12]$ make testc

gcc -c testCustomer.c

gcc -c Customer.c

gcc -o testc testCustomer.o Customer.o

[jsheng@powerpuff lab12]$ ./testc

<C1: 23, 8>

Customer id: C1, Arr: 23, Ser: 8

<C1: 23, 8>

----------------------

<C2: 24, 7>

Customer id: C2, Arr: 24, Ser: 7

<C2: 24, 7>

----------------------

<C3: 25, 6>

Customer id: C3, Arr: 25, Ser: 6

<C3: 25, 6>

----------------------

<C4: 26, 8>

Customer id: C4, Arr: 26, Ser: 8

<C4: 26, 8>

----------------------

<C5: 27, 6>

Customer id: C5, Arr: 27, Ser: 6

<C5: 27, 6>

----------------------

[jsheng@powerpuff lab12]$ make testk

gcc -c testKiosk.c

gcc -o testk testKiosk.o Customer.o kiosk.o

[jsheng@powerpuff lab12]$ ./testk

<K1: 0

K1: NULL [0]

<K2: 0

K2: NULL [0]

<K3: 0

K3: NULL [0]

<K4: 0

K4: NULL [0]

<K5: 0

K5: NULL [0]

<K1: 8

K1: <C1: 23, 8> [8]

<K2: 7

K2: <C2: 24, 7> [7]

<K3: 6

K3: <C3: 25, 6> [6]

<K4: 8

K4: <C4: 26, 8> [8]

<K5: 6

K5: <C5: 27, 6> [6]

Get 1 served.

[jsheng@powerpuff lab12]$ make testq

gcc -c queue.c

gcc -o testq testQ.o List.o queue.o node.o

[jsheng@powerpuff lab12]$ ./testq

Q Size=0

The queue is empty.

Adding: 83 86 77 15 93 35 86 92 49 21

Q Size=10

The queue is not empty.

Removed: 83

Removed: 86

Removed: 77

Removed: 15

Removed: 93

Removed: 35

Removed: 86

Removed: 92

Q Size=2

The queue is not empty.

**Reflection**

Initially, I implemented the queue without using List. But when I was revising, I suddenly found it is required to use List. Then I changed the component of queue from head, tail, size to List. And I changed every line of code containing things like “q->size” into “q->L->size” . However, after I done so, I got segmentation fault, even if I only run the first two line. Then I knew there must be something run with my newQueue() function. Then I went back to that function, and checked the class notes about segmentation fault, I found it is because I did not initialize the List in queue. After I created the new list and stored it in the heap segment, my program worked again! Another thing need to be mentioned is that I add some codes in testKiosk.c so that every functionality of module kiosk is now tested.