Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа**

**по курсу «ОC»**

**Тема:**

**Межпроцессорное взаимодействие.**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Соколов А.А. |
| Вариант: | 17 |
| Оценка: |  |
| Дата: |  |

Москва

2019

1.**Код на C:**

d\_queue.h:

#include <stdbool.h>

#ifndef D\_QUEUE\_NEW

#define D\_QUEUE\_NEW

typedef struct {

double \*body;

char name[5];

int size;

int cap;

int front;

}queue;

bool q\_grow(queue \*s);

void q\_srink(queue \*s);

queue \*double\_q\_create( char name[5]);

void double\_q\_destroy(queue \*q);

bool q\_is\_empty(queue \*q);

bool double\_q\_push\_back(queue \*q, double val);

double double\_q\_pop\_front(queue \*q);

#endif

d\_queue.c:

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include "d\_queue.h"

#define MIN\_CAP 4

queue \*double\_q\_create(char name[5]){

queue \*q=(queue\*)malloc(sizeof(queue));

q->body=(double \*)malloc(5\*sizeof(double));

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

q->name[i]=name[i];

}

q->size=0;

q->cap=5;

q->front=0;

return q;

}

bool q\_is\_empty(queue \*q){

if(q->size==0){

return true;

} else {

return false;

}

}

bool q\_grow(queue \*q){

int new\_cap=2\*q->cap;

double \*new\_body=(double \*)realloc(q->body,new\_cap\* sizeof(double));

if(new\_body==NULL){

return false;

}

for(int i = q->size-1 ;i >= q->front; i=i-1){

new\_body[i+(new\_cap-q->cap)]=q->body[i];

}

q->body=new\_body;

if(q->front==0) {

q->front = q->front;

}else{

q->front = q->front+(q->cap-new\_cap);

}

q->cap=new\_cap;

return true;

}

void q\_shrink(queue \*q){

if(q->size > q->cap/4){

return;

}

int new\_cap=q->cap/2;

if(new\_cap < MIN\_CAP){

new\_cap=MIN\_CAP;

}

if(q->front+q->size >= q->cap){

for (int i = q->front; i < q->cap; ++i) {

q->body[i-(q->cap-new\_cap)]=q->body[i];

}

q->front=q->front-(q->cap-new\_cap);

} else{

for (int i = q->front; i < q->size+q->front; ++i) {

q->body[i-(q->cap-new\_cap)]=q->body[i];

}

}

q->body=(double \*)realloc(q->body, sizeof(double)\*new\_cap);

q->cap=new\_cap;

return;

}

double double\_q\_pop\_front(queue \*q){

double val=q->body[q->front];

if (q->front==q->cap-1){

q->front=0;

} else {

q->front++;

}

q->size--;

return val;

}

void double\_q\_destroy(queue \*q){

free(q->body);

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

q->name[i]='\0';

}

q->size=0;

q->cap=0;

q->front=0;

}

bool double\_q\_push\_back(queue \*q, double val){

if(q->size==q->cap){

if(!q\_grow(q)){

return false;

}

}

q->body[(q->size+q->front)%q->cap]=val;

q->size++;

return true;

}

vector.h:

#include <stdio.h>

#include <stdbool.h>

#include "d\_queue.h"

#ifndef \_VECTOR\_

#define \_VECTOR\_

typedef struct {

queue \*body;

int size;

} vector;

vector \*v\_create();

void v\_destroy(vector \*v);

void v\_set(vector \*v, int i, queue \*val);

queue\* v\_get(vector \*v,int i);

bool v\_set\_size(vector \*v,int size);

int v\_get\_size(vector \*v);

bool v\_push(vector \*v,queue \*val);

#endif

vector.c:

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include "vector.h"

vector \*v\_create() {

vector \*v = (vector \*) malloc(sizeof(vector));

v->size = 0;

v->body = NULL;

return v;

}

void v\_destroy(vector \*v) {

free(v->body);

free(v);

}

int v\_get\_size(vector \*v) {

return v->size;

}

queue\* v\_get(vector \*v, int i) {

return &v->body[i];

}

bool v\_set\_size(vector \*v, int size) {

queue \*bodyre = (queue \*) realloc(v->body, size \* sizeof(queue));

if (bodyre == NULL && size != 0) {

return false;

}

if(bodyre == NULL){

return true;

}

for (int i = v->size; i<size; i++) {

char str[5]={'0','0','0','0','0'};

queue\* q=double\_q\_create(str);

bodyre[i]=\*q;

}

v->body=bodyre;

v->size=size;

return true;

}

void v\_set(vector \*v, int i, queue\* val) {

v->body[i] = \*val;

}

bool v\_push(vector \*v, queue\* val) {

int new\_size=v->size + 1;

if(!v\_set\_size(v,new\_size)){

return false;

}

v\_set(v,new\_size - 1,val);

return true;

}

main.c:

[#include](https://vk.com/im?sel=157419337&st=%23include) <stdbool.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <stdlib.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <string.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <sys/types.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <sys/wait.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) <unistd.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include)<pthread.h>  
[#include](https://vk.com/im?sel=157419337&st=%23include) "d\_queue.h"  
[#include](https://vk.com/im?sel=157419337&st=%23include) "vector.c"  
[#include](https://vk.com/im?sel=157419337&st=%23include) "d\_queue.c"  
[#include](https://vk.com/im?sel=157419337&st=%23include) "vector.h"  
[#include](https://vk.com/im?sel=157419337&st=%23include) <math.h>  
  
int search(int size, vector \*v, char key[5]) {  
int j = -1;  
for (int i = 0; i < size; ++i) {  
if (strcmp(v->body[i].name, key) == 0) {  
j = i;  
break;  
}  
}  
return j;  
}  
  
bool is\_space(char a) {  
if ((a == ' ') || (a == '\n') || (a == '\t') || (a == '\0')) {  
return true;  
} else {  
return false;  
}  
}  
  
double char\_to\_double(char a1[100]) {  
double a = 0;  
for (int k = 0; k < strlen(a1); ++k) {  
a = k \* 10 \* a1[k] + a;  
}  
return a;  
}  
  
int main(int argc, char \*argv[]) {  
char str[5];  
char str1[7];  
char help[1];  
char a1[100];  
char chara[200];  
double a;  
ssize\_t y;  
pid\_t child;  
vector \*v = v\_create();  
write(1, "Введите имя очереди команды и аргументы через пробел\n", 98);  
int pipefd[2];  
int pipefd2[2];  
if (pipe(pipefd) == -1) {  
perror("pipe");  
exit(EXIT\_FAILURE);  
}  
if (pipe(pipefd2) == -1) {  
perror("pipe2");  
exit(EXIT\_FAILURE);  
}  
child = fork();  
if (child != 0) {  
if (child == -1) {  
exit(EXIT\_FAILURE);  
} else {  
while (read(0, help, 1) > 0) {  
close(pipefd[0]);  
close(pipefd2[1]);  
bool flag = false;  
for (int k = 0; k < 6; ++k) {  
str[k] = 0;  
}  
for (int l = 0; l < 8; ++l) {  
str1[l] = 0;  
}  
for (int m = 0; m < 200; ++m) {  
chara[m] = 0;  
}  
while (is\_space(help[0])) {  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
}  
int i = 0;  
while ((!is\_space(help[0])) && (i < 5)) {  
str[i] = help[0];  
help[0] = 0;  
y = read(0, help, 1);  
++i;  
}  
while (is\_space(help[0])) {  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
}  
int j = 0;  
while (!is\_space(help[0])) {  
str1[j] = help[0];  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
++j;  
}  
if (y <= 0) {  
break;  
}  
if ((strcmp(str1, "create") == 0) || (strcmp(str1, "destroy") == 0) || (strcmp(str1, "push") == 0) ||  
(strcmp(str1, "pop") == 0)) {  
if (strcmp(str1, "push") == 0) {  
while (is\_space(help[0])) {  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
}  
int z = 0;  
while (!is\_space(help[0])) {  
chara[z] = help[0];  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
++z;  
}  
if (y <= 0) {  
break;  
}  
for (int k = 1; k < strlen(chara); ++k) {  
if ((chara[k] < '0') || (chara[k] > '9')) {  
flag = true;  
write(1, "Неверно введены аргументы\n", 50);  
}  
}  
if (chara[0] != '-') {  
if ((chara[0] < '0') || (chara[0] > '9')) {  
flag = true;  
write(1, "Неверно введены аргументы\n", 50);  
}  
}  
}  
} else {  
write(1, "Неверно введена команда\n", 46);  
flag = true;  
}  
if (flag == false) {  
if (y <= 0) {  
break;  
}  
write(pipefd[1], str, 5);  
write(pipefd[1], str1, 7);  
if (strcmp(str1, "push") == 0) {  
write(pipefd[1], chara, sizeof(chara));  
} else {  
write(pipefd[1], "\t", sizeof("\t"));  
}  
if (read(pipefd2[0], a1, 100) > 0) {  
if (((strcmp(str1, "pop") == 0)) && (((a1[0] >= 0) && (a1[0] <= 9)) || (a1[0] == '-'))) {  
if (a1[0] != '-') {  
for (int k = 0; k < strlen(a1); ++k) {  
a1[k] = a1[k] + 48;  
}  
write(1, a1, strlen(a1));  
write(1, "\n", 1);  
} else {  
for (int k = 1; k < strlen(a1); ++k) {  
a1[k] = a1[k] + 48;  
}  
write(1, a1, strlen(a1));  
write(1, "\n", 1);  
}  
} else {  
write(1, a1, strlen(a1));  
write(1, "\n", 1);  
}  
  
}  
}  
while (help[0] != '\n') {  
help[0] = 0;  
y = read(0, help, 1);  
if (y <= 0) {  
break;  
}  
}  
}  
}  
} else {  
while ((read(pipefd[0], str, 5) > 0) && (read(pipefd[0], str1, 7) > 0) && (read(pipefd[0], chara, 200) > 0)) {  
close(pipefd[1]);  
close(pipefd2[0]);  
int c = 0;  
while (((str[c] >= 'a') && (str[c] <= 'z')) || ((str[c] >= '0') && (str[c] <= '9'))) {  
++c;  
}  
str[c] = 0;  
int k = 0;  
while ((str1[k] >= 'a') && (str1[k] <= 'z')) {  
++k;  
}  
str1[k] = 0;  
if (strcmp(str1, "create") == 0) {  
int i;  
i = search(v->size, v, str);  
if (i != -1) {  
write(pipefd2[1], "Очередь с таким именем уже существует\n", 76);  
} else {  
queue \*q = double\_q\_create(str);  
v\_push(v, q);  
write(pipefd2[1], "\0", sizeof("\0"));  
}  
}

else {  
if (strcmp(str1, "destroy") == 0) {  
int i;  
i = search(v->size, v, str);  
if (i == -1) {  
write(pipefd2[1], "Очереди с таким именем не существует\n", 74);  
} else {  
double\_q\_destroy(&v->body[i]);  
for (int j = i; j < v->size; ++j) {  
v->body[i] = v->body[i + 1];  
}  
—v->size;  
write(pipefd2[1], "\0", sizeof("\0"));  
}  
} else {  
if (strcmp(str1, "push") == 0) {  
int i;  
i = search(v->size, v, str);  
if (i == -1) {  
write(pipefd2[1], "Очереди с таким именем не существует\n", 74);  
} else {  
char \*\*endstr = 0;  
if (chara[0] == '-') {  
a = strtod(chara + 1 \* sizeof(char), endstr);  
if (chara[0] == '-') {  
a = -a;  
}  
double\_q\_push\_back(&v->body[i], a);  
write(pipefd2[1], "\0", sizeof("\0"));  
} else {  
a = strtod(chara, endstr);  
double\_q\_push\_back(&v->body[i], a);  
write(pipefd2[1], "\0", sizeof("\0"));  
}  
}  
} else {  
if (strcmp(str1, "pop") == 0) {  
int i;  
i = search(v->size, v, str);  
if (i == -1) {  
write(pipefd2[1], "Очереди с таким именем не существует\n", 74);  
} else {  
if (q\_is\_empty(&v->body[i])) {  
write(pipefd2[1], "Очередь пуста\n", 28);  
} else {  
double d;  
d = double\_q\_pop\_front(&v->body[i]);  
if (d >= 0) {  
for (int j = 0; j < 200; ++j) {  
chara[j] = 0;  
}  
double s = 1;  
while ((d / s) > 9) {  
s = s \* 10;  
}  
int z = 0;  
while (s >= 1) {  
chara[z] = (char) (d / s);  
d = fmod(d, s);  
s = s / 10;  
++z;  
}  
write(pipefd2[1], chara, 80);  
write(pipefd2[1], "\n\0", sizeof("\n\0"));  
} else {  
d = -d;  
for (int j = 0; j < 200; ++j) {  
chara[j] = 0;  
}  
double s = 1;  
while ((d / s) > 9) {  
s = s \* 10;  
}  
int z = 1;  
chara[0] = '-';  
while (s >= 1) {  
chara[z] = (char) (d / s);  
d = fmod(d, s);  
s = s / 10;  
++z;  
}  
write(pipefd2[1], chara, 80);  
write(pipefd2[1], "\n\0", sizeof("\n\0"));  
}  
}  
}  
}  
}  
}  
}  
}  
}  
return 0;  
}

2.**Набор тестов:**

test1:

q1 create

q1 push 6

q1 pop

6

q1 destroy

test2:

q1 pop

Очереди с таким именем не существует

q1 crete

Неверно введена команда

test3:

q1 create

q1 pop

Очередь пуста

test4:

q1 create

q1 push 4

q1 push 3

q1 push 2

q1 pop

4

q1 pop

3

q1 pop

2

3.**Результаты работы программы на тестах:**

test1:

q1 create

q1 push 6

q1 pop

6

q1 destroy

test2:

q1 pop

Очереди с таким именем не существует

q1 crete

Неверно введена команда

test3:

q1 create

q1 pop

Очередь пуста

test4:

q1 create

q1 push 4

q1 push 3

q1 push 2

q1 pop

4

q1 pop

3

q1 pop

2

4**. Объяснение работы программы:**

Пользователь вводит имя очереди и команду, которая как-то действует на данную очередь, и её аргументы, если они есть. Программа в родительском процессе только проверяет верность команд и их аргументов, после чего обработка очереди, в соответствии с введенной командой, идет на дочерний процесс. После обработки очереди дочерний процесс возвращает родительскому результат, который родительский процесс выводит пользователю.

5.**Вывод:**

В данной программе показывается каким образом, с помощью возможностей языка С, можно распределить выполнение одной задачи на несколько процессов, используя, в основном ,такие системные вызовы как fork-создание нового дочернего процесса и pipe-создание канала передачи данных между процессами.