МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №3

Выполнил студент группы Б17-501: Борзенков А.В.

Проверил: Ктитров С.В.

Москва, 2020

Задача:

Вариант С–3

Тема «Очереди сообщений SYSTEM V»

Разработать программу, позволяющую интерактивно работать с очередью: помещать сообщение, получать его (указывая тип), просматривать состояние очереди, создавать и удалять ее. Программа должна собираться из нескольких файлов с использованием make.

Выполнение:

После выполнения команды make в консоль выдаётся окно с номерами команд, которые реализованы в программном коде, далее в зависимости от запрашиваемых данных требуется вводить те или иные данные в консоль.

Код программы:

main.c

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. #define MSG_SIZE 1024
5.
6. extern int SendMessage(int,char*,int);
7. extern char* GetMessage(int,int);
8. extern char* GetStateOfQueue(int);
9. extern long long GetKey(const char*, int);
10.    extern int CreateQueue(long long);
11.    extern int RemoveQueue(int);
12.
13.    void NoQueue() {
14.        printf("First, You have to set the queue with it's id\n");
15.    }
16.
17.    void WriteResult(const char* s, int result) {
18.        printf("%s %d\n", s, result);
19.    }
20.
21.    int main(int argc, char* argv[]) {
22.        for (int i = 0; i < (1 << 15); i++) { RemoveQueue(i); }
23.
24.        int cmd = 0;
25.        int queue_id = -1;
26.
27.        /*
28.         * 1 - send message to the queue
29.         * 2 - get message with setted type
30.         * 3 - print state of the queue
31.         * 4 - create queue
32.         * 5 - delete queue
33.         * 6 - get info
34.         * 0 - exit
35.         */
```

```
36.       const char* info_string = "1 - send message to the queue\n2 - get
   message with a set type\n3 - print state of the queue\n4 - create a new
   queue\n5 - delete the queue\n6 - get info\n0 - exit";
37.       printf("%s\n", info_string);
38.       printf("Write command\n");
39.       scanf("%d", &cmd);
40.       while (cmd != 0) {
41.           printf("Command is %d\n", cmd);
42.           if (cmd == 1) {
43.               if (queue_id == -1) {
44.                   NoQueue();
45.               } else {
46.                   printf("Write your message:\n");
47.                   char* s = malloc(sizeof(char) * MSG_SIZE);
48.                   scanf("%s", s);
49.                   int type;
50.                   printf("Write type of message:\n");
51.                   scanf("%d", &type);
52.                   int result = SendMessage(queue_id, s, type);
53.                   WriteResult("Result of sending message", result);
54.               }
55.           } else if (cmd == 2) {
56.               if (queue_id == -1) {
57.                   NoQueue();
58.               } else {
59.                   int type;
60.                   printf("Write type of message: ");
61.                   scanf("%d", &type);
62.                   printf("%s\n", GetMessage(queue_id, type));
63.               }
64.           } else if (cmd == 3) {
65.               if (queue_id == -1) {
66.                   NoQueue();
67.               } else {
68.                   char* result = GetStateOfQueue(queue_id);
69.                   printf("%s\n", result);
70.                   fflush(stdout);
71.               }
72.           } else if (cmd == 4) {
73.               if (queue_id == -1) {
74.                   int id;
75.                   printf("Write id for key: ");
76.                   scanf("%d", &id);
77.                   queue_id = CreateQueue(GetKey(argv[0], id));
78.                   WriteResult("Queue`s id is", queue_id);
79.               } else {
80.                   printf("Queue already exists\n");
81.               }
82.           } else if (cmd == 5) {
83.               if (queue_id == -1) {
```

```
                    NoQueue();
                } else {
                    int result = RemoveQueue(queue_id);
                    WriteResult("Result of removing queue", result);
                    queue_id = -1;
                }
            } else if (cmd == 6) {
                printf("%s\n", info_string);
            } else {
                printf("Error command\n");
            }
            printf("\n");


            printf("Write next command:\n");
            scanf("%d", &cmd);
        }

        if (queue_id != -1) {
            RemoveQueue(queue_id);
        }

    }
```

message_queue.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct mymsg {
     long mtype;
        char mtext[1024];
    };

    #define MSG_SIZE 1024

    key_t GetKey(char* path, int id) {
        return ftok(path, id);
    }

    int CreateQueue(key_t key) {
        const int FLAG = IPC_CREAT | IPC_EXCL | 0666;
        return msgget(key, FLAG);
    }

    int SendMessage(int queue_id, char* message, int type) {
```

```
25.       struct mymsg sending_message;
26.       sending_message.mtype = type;
27.       sprintf(sending_message.mtext, "%s", message);
28.       return msgsnd(queue_id, &sending_message, MSG_SIZE, MSG_NOERROR);
29.   }
30.
31.   char* GetMessage(int queue_id, int type) {
32.       struct mymsg received_message;
33.       int receive_status =  msgrcv(queue_id, &received_message, MSG_SIZE,
   0, MSG_NOERROR);
34.       if (receive_status == -1) {
35.           return "no message";
36.       } else {
37.           char* result = malloc(MSG_SIZE);
38.           sprintf(result, received_message.mtext);
39.           return result;
40.       }
41.   }
42.
43.
44.   struct msqid_ds GetCurrentState(int queue_id) {
45.       struct msqid_ds result;
46.       msgctl(queue_id, IPC_SET, &result);
47.       return result;
48.   }
49.
50.   int AddToLineString(char** d, int pos, char* s) {
51.       int n = strlen(s);
52.       for (int i = 0; i < n; i++) {
53.           (*d)[i + pos] = s[i];
54.       }
55.       return n;
56.   }
57.
58.   int AddToLineInt(char** d, int pos, int value) {
59.       char* s = malloc(128 * sizeof(char));
60.       sprintf(s, "%lld", value);
61.       int n = strlen(s);
62.       for (int i = 0; i < n; i++) {
63.           (*d)[i + pos] = s[i];
64.       }
65.       return n;
66.   }
67.
68.   char* GetStateOfQueue(int queue_id) {
69.       struct msqid_ds state = GetCurrentState(queue_id);
70.       struct ipc_perm perm = state.msg_perm;
71.
72.       char* result = (char*) malloc(sizeof(char) * MSG_SIZE);
73.
```

```
74.       int pos = 0;
75.       pos += AddToLineString(&result, pos, "ipc_perm:\n");
76.       pos += AddToLineString(&result, pos, " key = ");
77.       pos += AddToLineInt(&result, pos, perm.__key);
78.       pos += AddToLineString(&result, pos, "\n");
79.
80.       pos += AddToLineString(&result, pos, " uid = ");
81.       pos += AddToLineInt(&result, pos, perm.uid);
82.       pos += AddToLineString(&result, pos, "\n");
83.
84.       pos += AddToLineString(&result, pos, " gid = ");
85.       pos += AddToLineInt(&result, pos, perm.gid);
86.       pos += AddToLineString(&result, pos, "\n");
87.
88.       pos += AddToLineString(&result, pos, " cuid = ");
89.       pos += AddToLineInt(&result, pos, perm.cuid);
90.       pos += AddToLineString(&result, pos, "\n");
91.
92.       pos += AddToLineString(&result, pos, " cgid = ");
93.       pos += AddToLineInt(&result, pos, perm.cgid);
94.       pos += AddToLineString(&result, pos, "\n");
95.
96.       pos += AddToLineString(&result, pos, " mode = ");
97.       pos += AddToLineInt(&result, pos, perm.mode);
98.       pos += AddToLineString(&result, pos, "\n");
99.
100.      pos += AddToLineString(&result, pos, " seq = ");
101.      pos += AddToLineInt(&result, pos, perm.__seq);
102.      pos += AddToLineString(&result, pos, "\n");
103.
104.      pos += AddToLineString(&result, pos, "\n");
105.
106.      pos += AddToLineString(&result, pos, "msg_qnum = ");
107.      pos += AddToLineInt(&result, pos, state.msg_qnum);
108.      pos += AddToLineString(&result, pos, "\n");
109.
110.      pos += AddToLineString(&result, pos, "msg_qbytes = ");
111.      pos += AddToLineInt(&result, pos, state.msg_qbytes);
112.      pos += AddToLineString(&result, pos, "\n");
113.
114.      pos += AddToLineString(&result, pos, "msg_lspid = ");
115.      pos += AddToLineInt(&result, pos, state.msg_lspid);
116.      pos += AddToLineString(&result, pos, "\n");
117.
118.      pos += AddToLineString(&result, pos, "msg_lrpid = ");
119.      pos += AddToLineInt(&result, pos, state.msg_lrpid);
120.      pos += AddToLineString(&result, pos, "\n");
121.
122.      pos += AddToLineString(&result, pos, "msg_msg_stime = ");
123.      pos += AddToLineInt(&result, pos, state.msg_stime);
```

```
124.      pos += AddToLineString(&result, pos, "\n");
125.
126.      pos += AddToLineString(&result, pos, "msg_rtime = ");
127.      pos += AddToLineInt(&result, pos, state.msg_rtime);
128.      pos += AddToLineString(&result, pos, "\n");
129.
130.      pos += AddToLineString(&result, pos, "msg_ctime = ");
131.      pos += AddToLineInt(&result, pos, state.msg_ctime);
132.      pos += AddToLineString(&result, pos, "\n");
133.
134.
135.      pos += AddToLineString(&result, pos, "\0");
136.
137.      return result;
138.  }
139.
140.  int RemoveQueue(int queue_id) {
141.      return msgctl(queue_id, IPC_RMID, NULL);
142.  }
```

Makefile

```
1. SHELL = /bin/bash
2. run:
3.          gcc -c message_queue.c main.c
4.          gcc -o main message_queue.o main.o
5.          ./main
```

При любом изменении будут транслироваться оба файла

Пример работы:

```
1. kali@kali:~/home_work/lab3$ make
2. gcc -c message_queue.c main.c
3. gcc -o main message_queue.o main.o
4. ./main
5. 1 - send message to the queue
6. 2 - get message with a set type
7. 3 - print state of the queue
8. 4 - create a new queue
9. 5 - delete the queue
10.   6 - get info
11.   0 - exit
12.   Write command
13.   4
14.   Command is 4
15.   Write id for key: 17
16.   Queue`s id is 32770
17.
18.   Write next command:
19.   1
```

```
Command is 1
Write your message:
NewMessageToTheQueue
Write type of message:
5
Result of sending message 0

Write next command:
3
Command is 3
ipc_perm:
 key = 0
 uid = 0
 gid = 0
 cuid = 0
 cgid = 0
 mode = 0
 seq = 255

msg_qnum = 448405776
msg_qbytes = 1232500992
msg_lspid = 0
msg_lrpid = 0
msg_msg_stime = 16711680
msg_rtime = 1232500992
msg_ctime = 0


Write next command:
2
Command is 2
Write type of message: 5
NewMessageToTheQueue

Write next command:
5
Command is 5
Result of removing queue 0

Write next command:
6
Command is 6
1 - send message to the queue
2 - get message with a set type
3 - print state of the queue
4 - create a new queue
5 - delete the queue
6 - get info
0 - exit

```

70. Write next command:
71. 0