Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**ОЧЕРЕДЬ СООБЩЕНИЙ В UNIX И РАБОТА С НИМИ**

Лабораторная работа №7по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_\_\_ Петров Ф. М.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** Научиться работать с очередям сообщений в UNIX-like системах.

**Задание:**

В данной работе требуется использовать параллельные потоки (threads), а не процессы.

Основной поток должен сгенерировать и вывести на экран 4 случайных числа.

Далее эти числа должны быть переданы во второй процесс через очередь сообщений.

Второй поток должен сгенерировать все возможные перестановки из этих чисел без повторений («1 2 3 4», «2 1 3 4», «4 3 1 2» и т.д.) и передать их обратно.

Первый поток должен получить все эти перестановки и вывести их на экран. В конце требуется вывести количество полученных перестановок.

Первый поток должен самостоятельно выполнить освобождение всех выделенных ресурсов в конце своей работы.

В отчете привести исходный код программы, а также результаты, выведенные на экран.

**Исходный код программы:**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <pthread.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct msgbuf

{

long mtype;

int digits[4];

};

struct args

{

int msgId;

};

int comp\_func(const void \*x, const void \*y)

{

return \*(int \*)x - \*(int \*)y;

}

void swap(int \*digits, int x, int y)

{

int tmp = digits[x];

digits[x] = digits[y];

digits[y] = tmp;

}

int gen(int \*digits, size\_t n)

{

int j = n - 2;

while (j != -1 && digits[j] >= digits[j + 1])

{

j--;

}

if (j == -1)

{

return 0;

}

int k = n - 1;

while (digits[j] >= digits[k])

{

k--;

}

swap(digits, j, k);

int l = j + 1, r = n - 1;

while (l < r)

{

swap(digits, l++, r--);

}

return 1;

}

void \*mainThd(void \*thd\_args)

{

srand(time(NULL));

int digits[4];

for (size\_t i = 0; i < 4; ++i)

{

digits[i] = rand() % 100;

printf("%d ", digits[i]);

}

printf("\n");

struct args \*args\_var = (struct args \*)thd\_args;

int msgId = args\_var->msgId;

struct msgbuf message;

memcpy(message.digits, digits, 4 \* sizeof(int));

message.mtype = 1;

msgsnd(msgId, &message, sizeof(message), 0);

int cop = 0;

do

{

size\_t size = msgrcv(msgId, &message, sizeof(message), 101, 0);

if (message.digits[0] != -1)

{

for (size\_t i = 0; i < 4; ++i)

{

printf("%d ", message.digits[i]);

}

cop++;

printf("\n");

}

} while (message.digits[0] != -1);

printf("Count of permutations: %d\n", cop);

msgctl(msgId, IPC\_RMID, NULL);

return NULL;

}

void \*childThd(void \*thd\_args)

{

struct args \*args = (struct args \*)thd\_args;

int msgId = args->msgId;

struct msgbuf message;

msgrcv(msgId, &message, sizeof(message), 1, 0);

qsort(message.digits, 4, sizeof(int), comp\_func);

message.mtype = 101;

msgsnd(msgId, &message, sizeof(message), 0);

while (gen(message.digits, 4))

{

message.mtype = 101;

msgsnd(msgId, &message, sizeof(message), 0);

}

message.digits[0] = -1;

message.mtype = 101;

msgsnd(msgId, &message, sizeof(message), 0);

return NULL;

}

int main()

{

int msgId = msgget(IPC\_PRIVATE, 0600 | IPC\_CREAT);

struct args \*args\_var;

args\_var->msgId = msgId;

pthread\_t mainThread, childThread;

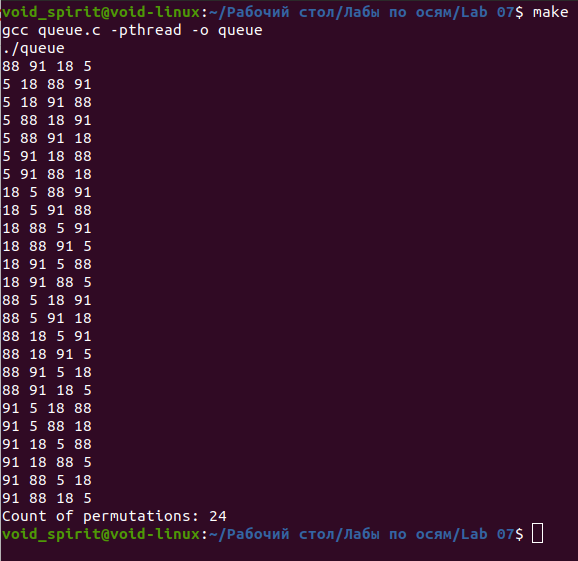
pthread\_create(&mainThread, NULL, mainThd, (void \*)args\_var);

pthread\_create(&childThread, NULL, childThd, (void \*)args\_var);

pthread\_join(mainThread, NULL);

}

**Скриншот результата выполнения программы:**

****

**Вывод:** В процессе выполнения лабораторной работы я научился работать с очередями сообщений.