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»

**Очереди сообщений в UNIX и работа с ними**

Лабораторная работа №7 по 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:

\_\_\_\_\_\_\_ Шаклеин В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** познакомиться с механизмами работы сообщений в UNIX.

**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.

**Содержание файла msg\_queue.c:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

struct thread\_args

{

size\_t msgId1;

size\_t msgId2;

};

struct my\_msg

{

long mtype;

char msg[4];

};

void print\_nums(char \*nums, int c)

{

for (int i = 0; i < c; i++)

{

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

}

printf("\n");

}

void \*mainTask(void \*thread\_args)

{

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

size\_t msgId1 = args->msgId1;

size\_t msgId2 = args->msgId2;

struct my\_msg send;

struct my\_msg recive;

char nums[4];

srand((unsigned)(time(0)));

for (int i = 0; i < 4; i++)

{

nums[i] = rand() % 10;

}

print\_nums(nums, 4);

send.mtype = 1;

for (int i = 0; i < 4; i++)

{

send.msg[i] = nums[i];

}

msgsnd(msgId1, &send, sizeof(send), 0);

msgrcv(msgId2, &recive, sizeof(recive), 0, 0);

int p\_count = recive.msg[0];

printf("Permutations:\n");

for (int i = 0; i < p\_count; i++)

{

msgrcv(msgId2, &recive, sizeof(recive), 0, 0);

print\_nums(recive.msg, 4);

}

printf("Permutations count:%d\n", p\_count);

msgctl(msgId1, IPC\_RMID, NULL);

msgctl(msgId2, IPC\_RMID, NULL);

return NULL;

}

void \*childTask(void \*thread\_args)

{

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

size\_t msgId1 = args->msgId1;

size\_t msgId2 = args->msgId2;

struct my\_msg send;

struct my\_msg recive;

char nums[4];

msgrcv(msgId1, &recive, sizeof(recive), 0, 0);

for (int i = 0; i < 4; i++)

{

nums[i] = recive.msg[i];

}

char p[24][4];

int p\_count = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

if (j == i)

continue;

for (int k = 0; k < 4; k++)

{

if (k == i || k == j)

continue;

int l = 6 - i - j - k;

char curr\_p[4];

curr\_p[0] = nums[i];

curr\_p[1] = nums[j];

curr\_p[2] = nums[k];

curr\_p[3] = nums[l];

char unique = 1;

for (int m = 0; m < p\_count; m++)

{

if (memcmp(p[m], curr\_p, 4) == 0)

{

unique = 0;

break;

}

}

if (unique)

{

memcpy(p[p\_count], curr\_p, 4);

p\_count++;

}

}

}

}

send.mtype = 2;

send.msg[0] = p\_count;

msgsnd(msgId2, &send, sizeof(send), 0);

send.mtype = 1;

for (int i = 0; i < p\_count; i++)

{

memcpy(send.msg, p[i], 4);

msgsnd(msgId2, &send, sizeof(send), 0);

}

return NULL;

}

int main()

{

int msgId1 = msgget(IPC\_PRIVATE, 0600 | IPC\_CREAT);

int msgId2 = msgget(IPC\_PRIVATE, 0600 | IPC\_CREAT);

struct thread\_args \*args;

args->msgId1 = msgId1;

args->msgId2 = msgId2;

pthread\_t mainThread, childThread;

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

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

pthread\_join(mainThread, NULL);

}

**Результат, выведенный на экран:**

sevskii@MBP-sevskii 7 % make

gcc msg\_queue.c -pthread -o msg\_queue

./msg\_queue

6 9 9 7

Permutations:

6 9 9 7

6 9 7 9

6 7 9 9

9 6 9 7

9 6 7 9

9 9 6 7

9 9 7 6

9 7 6 9

9 7 9 6

7 6 9 9

7 9 6 9

7 9 9 6

Permutations count:12

**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с работой сообщений в UNIX.