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:

\_\_\_\_\_\_\_ Шляханов Д. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 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 queue.c:**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <pthread.h>

#include <unistd.h>

#include <string.h>

struct Strmsg

{

long mtype;

int mtext[4];

int islast;

};

void print\_nums(int \*nums, int n)

{

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

{

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

}

printf("\n");

}

int comparator(const void \*a, const void \*b)

{

return \*(int \*)a - \*(int \*)b;

}

void swap(int \*nums, int i, int j)

{

int a = nums[i];

nums[i] = nums[j];

nums[j] = a;

}

int next(int \*nums, int n)

{

int i = n - 2;

while (i != -1 && nums[i] >= nums[i + 1]) i--;

if (i == -1) return 0;

int j = n - 1;

while (nums[i] >= nums[j]) j--;

swap(nums, i, j);

int m = i + 1, k = n - 1;

while (m < k) swap(nums, m++, k--);

return 1;

}

void \*parentMainCode(int msgId)

{

struct Strmsg localmsg;

int nums[4], count = 0;

srand(time(0));

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

{

localmsg.mtext[i] = rand() % 10;

}

print\_nums(localmsg.mtext, 4);

printf("\n");

localmsg.mtype = 1;

localmsg.islast = 1;

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

while (localmsg.islast)

{

msgrcv(msgId, &localmsg, sizeof(localmsg), 2, 0);

if (!localmsg.islast) break;

print\_nums(localmsg.mtext, 4);

count++;

}

printf("Combinations count:%d\n", count);

waitpid(0, 0, 0);

msgctl(msgId, IPC\_RMID, NULL);

}

void \*childMainCode(int msgId)

{

struct Strmsg localmsg;

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

qsort(localmsg.mtext, 4, sizeof(int), comparator);

localmsg.mtype = 2;

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

while(localmsg.islast)

{

localmsg.islast = next(localmsg.mtext, 4);

localmsg.mtype = 2;

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

}

}

int main()

{

const size\_t semCount = 10;

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

pid\_t childId = fork();

if (childId > 0)

{

parentMainCode(msgId);

}

else

{

childMainCode(msgId);

}

}

**Результат, выведенный на экран:**

den@den-gremlin:~/7$ make

gcc queue.c -pthread -o queue

./queue

3 6 7 2

2 3 6 7

2 3 7 6

2 6 3 7

2 6 7 3

2 7 3 6

2 7 6 3

3 2 6 7

3 2 7 6

3 6 2 7

3 6 7 2

3 7 2 6

3 7 6 2

6 2 3 7

6 2 7 3

6 3 2 7

6 3 7 2

6 7 2 3

6 7 3 2

7 2 3 6

7 2 6 3

7 3 2 6

7 3 6 2

7 6 2 3

7 6 3 2

Combinations count:24

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