Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования "Новгородский государственный университет имени Ярослава Мудрого"

Кафедра «Информационных технологий и систем»

Дисциплина «Операционные системы»

Отчет по лабораторной работе

«Очереди сообщений в UNIX и работа с ними»

Выполнил студент группы 9091

\_\_\_\_\_\_\_\_\_\_\_\_\_/Алентьев Александр Григорьевич/

Подпись ФИО

Принял преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_/Ананьев Владислав Валерьевич/

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работ:**

Цель работы: познакомиться с механизмом общения между процессами, используя очереди сообщений в UNIX.

**Задание:**

1. В данной работе требуется использовать параллельные потоки (threads), а не процессы.
2. Основной поток должен сгенерировать и вывести на экран 4 случайных числа.
3. Далее эти числа должны быть переданы во второй процесс через очередь сообщений.
4. 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.
5. Первый поток должен получить все эти перестановки и вывести их на экран. В конце требуется вывести количество полученных перестановок.
6. Первый поток должен самостоятельно выполнить освобождение всех выделенных ресурсов в конце своей работы.

**Ход работы:**

**Исходный текст программы:**

|  |
| --- |
| main.cpp |
| #include "stdio.h"  #include "stdlib.h"  #include "unistd.h"  #include "time.h"  #include "pthread.h"  #include "sys/ipc.h"  #include "sys/msg.h"  #include <iostream>  typedef struct {  long mtype;  int data[4];  } msgStruct;  void swap(int \*a, int i, int j)  {  int s = a[i];  a[i] = a[j];  a[j] = s;  }  void Print(int \*a, int n) // permutation output  {  static int num = 1; // permutation number  std::cout.width(3); // width of the output field of the permutation number  std::cout << num++ << ": ";  for (int i = 0; i < n; i++)  std::cout << a[i] << " ";  std::cout << std::endl;  }  bool NextSet(int \*a, int n)  {  int j = n - 2;  while (j != -1 && a[j] >= a[j + 1]) j--;  if (j == -1)  return false; // no more permutations  int k = n - 1;  while (a[j] >= a[k]) k--;  swap(a, j, k);  int l = j + 1, r = n - 1; // sort the rest of the sequence  while (l<r)  swap(a, l++, r--);  return true;  }  void\* pthreadFunc(void\* args)  {  msgStruct pthreadMsg;  int msgId = \*((int \*) args);  ssize\_t len = msgrcv(msgId, &pthreadMsg, sizeof(pthreadMsg), 0, 0);    while (NextSet(pthreadMsg.data, 4))  msgsnd(msgId, &pthreadMsg, sizeof(pthreadMsg), 0);  return 0;  }  int main(void)  {  int randNum[4];  srand(time(NULL));  for (int i = 0; i < 4; i++)  randNum[i] = rand() % 9;  //-------------Create msg--------------  int msgId;  if(msgId = msgget(IPC\_PRIVATE, 0666 | IPC\_CREAT) < 0)  {  perror("Error with msgget()!\n");  return -1;  }  pthread\_t thread;  int res1 = pthread\_create(&thread, NULL, pthreadFunc, &msgId);  msgStruct parentMsg;  parentMsg.mtype = 1;  for (int i = 0; i < 4; i++)  parentMsg.data[i] = randNum[i];  msgsnd(msgId, &parentMsg, sizeof(parentMsg), 0);  int msgStart = 0;  int msgEnd = 24;  while (msgStart != msgEnd)  {  ssize\_t len = msgrcv(msgId, &parentMsg, sizeof(parentMsg), 0, 0);  Print(parentMsg.data, 4);  msgStart++;  }  msgctl(msgId, IPC\_RMID, NULL);  return 0;  } |

|  |
| --- |
| Output Terminal |
| >> make  gcc main.c -o main  ./main  1: 1 2 3 4  2: 1 2 4 3  3: 1 3 2 4  4: 1 3 4 2  5: 1 4 2 3  6: 1 4 3 2  7: 2 1 3 4  8: 2 1 4 3  9: 2 3 1 4  10: 2 3 4 1  11: 2 4 1 3  12: 2 4 3 1  13: 3 1 2 4  14: 3 1 4 2  15: 3 2 1 4  16: 3 2 4 1  17: 3 4 1 2  18: 3 4 2 1  19: 4 1 2 3  20: 4 1 3 2  21: 4 2 1 3  22: 4 2 3 1  23: 4 3 1 2  24: 4 3 2 1 |

**Вывод:**

Входе выполнения лабораторной работы, я познакомился с механизмом синхронизации процессов при помощи семафоров в операционной системе UNIX.