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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Малинин Никита Валерьевич/

Подпись ФИО

Принял преподаватель

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

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работы**

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

**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 mdata[4];  char isLast;  } msgStruct;  void swap(int \*a, int i, int j)  {  int s = a[i];  a[i] = a[j];  a[j] = s;  }  void printMsg(int \*a, int n)  {  static int num = 1;  std::cout.width(3);  std::cout << num++ << ": ";  for (int i = 0; i < n; i++)  {  std::cout << a[i] << " ";  }  std::cout << std::endl;  }  bool permutationsAlgorithm(int \*a, int n)  {  int j = n - 2;  while (j != -1 && a[j] >= a[j + 1]) j--;  if (j == -1)  return false;  int k = n - 1;  while (a[j] >= a[k]) k--;  swap(a, j, k);  int l = j + 1, r = n - 1;  while (l<r)  swap(a, l++, r--);  return true;  }  void\* pthreadFunction(void\* args)  {  msgStruct pthreadMsg;  int msgID = \*((int\*) args);  ssize\_t len = msgrcv(msgID, &pthreadMsg, sizeof(pthreadMsg), 0, 0);  do  {  if(!std::next\_permutation(pthreadMsg.mdata, pthreadMsg.mdata + 4))  {  pthreadMsg.isLast = 1;  msgsnd(msgID, &pthreadMsg, sizeof(pthreadMsg), 0);  break;  }  msgsnd(msgID, &pthreadMsg, sizeof(pthreadMsg), 0);  } while(1);    return 0;  }  int main(void)  {  int randomNumber[4];  srand(time(NULL));  for(int i = 0; i < 4; i++)  {  randomNumber[i] = rand() % 9;  }  int msgID;  if(msgID = msgget(IPC\_PRIVATE, 0666 | IPC\_CREAT) < 0)  {  perror("Error with msgget()!\n");  return -1;  }  pthread\_t thread;  int result = pthread\_create(&thread, NULL, pthreadFunction, &msgID);  msgStruct parentMsg;  parentMsg.mtype = 1;  parentMsg.isLast = 0;    for(int i = 0; i < 4; i++)  {  parentMsg.mdata[i] = randomNumber[i];  }  msgsnd(msgID, &parentMsg, sizeof(parentMsg), 0);  while(!parentMsg.isLast)  {  msgrcv(msgID, &parentMsg, sizeof(parentMsg), 0, 0);  printMsg(parentMsg.mdata, 4);  };  msgctl(msgID, IPC\_RMID, NULL);  return 0;  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| >> make  g++ main.cpp -lpthread -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**

Вывод: выполняя лабораторную работу, я познакомился и научился работать с механизмом общения между процессами, используя очереди сообщений в UNIX.