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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Юшин Григорий Станиславович/

Подпись ФИО

Принял преподаватель

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

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работы**

## Задание на лабораторную работу

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 поток 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.

**Исходный текст программ**

|  |
| --- |
| main.c |
| #include <stdlib.h>  #include <sys/types.h>  #include <sys/ipc.h>  #include <sys/msg.h>  #include <string.h>  #include <stdio.h>  #include <pthread.h>  #include <unistd.h>  #include <stdbool.h>  #define LAST\_MESSAGE 255  struct msgbuf {  long msg\_type;  int msg\_text;  };  const size\_t msg\_length = sizeof(int);  const size\_t inc\_msglen = sizeof(int[4]);  void swap (int \*a, int \*b) {  int t = \*a;  \*a = \*b;  \*b = t;  }  bool NextSet(int \*a, int n)  {  int j = n - 2;  while (j != -1 && a[j] >= a[j + 1]) j--;  if (j == -1)  return false; // больше перестановок нет  int k = n - 1;  while (a[j] >= a[k]) k--;  swap(&a[j],&a[k]);  int l = j + 1, r = n - 1; // сортируем оставшуюся часть последовательности  while (l<r)  {  swap(&a[l], &a[r]);  l++; r--;  }  return true;  }  void sortArraySM(int \*mem, int ArraySize)  {  int temp;  for (int i = 0; i < ArraySize - 1; i++)  {  for (int j = 0; j < ArraySize - i - 1; j++)  {  if (mem[j] > mem[j + 1])  {  temp = mem[j];  mem[j] = mem[j + 1];  mem[j + 1] = temp;  };  };  };  }  void \*secondThread(void \*arg)  {  int mid = (int) arg; //short-int-char size == size of ptr - ez cast  struct msgbuf income\_msg;  int cArray[4];  int arraySize = 4;  for(int i = 0; i < 4; i++)  {  if((msgrcv(mid, (struct msgbuf\*) &income\_msg, msg\_length, 0,0) < 0) && income\_msg.msg\_type != 1)  {  printf("Can't recieve anything or incorrect msg type\n");  exit(-1);  }  else  {  printf("Recieved %i msg: %i\n", i+1, income\_msg.msg\_text);  cArray[i] = income\_msg.msg\_text;  };  };  sortArraySM(cArray, arraySize);  for(int i = 0; i < arraySize-1; i++)  {  if(cArray[i] == cArray[i+1])  {  swap(&cArray[i],&cArray[arraySize-1]);  arraySize--;  }  }  sortArraySM(cArray,arraySize);  income\_msg.msg\_text = arraySize;  msgsnd(mid, (struct msgbuf \*) &income\_msg, msg\_length, 0);  if(arraySize > 1)  {  for(int i = 0; i < arraySize; i++)  {  income\_msg.msg\_text = cArray[i];  msgsnd(mid, (struct msgbuf \*) &income\_msg, msg\_length, 0);  }  while(NextSet(cArray, arraySize))  {  for(int i = 0; i < arraySize; i++)  {  income\_msg.msg\_text = cArray[i];  msgsnd(mid, (struct msgbuf \*) &income\_msg, msg\_length, 0);  }  }  } else  {  income\_msg.msg\_text = cArray[0];  msgsnd(mid, (struct msgbuf \*) &income\_msg, msg\_length, 0);  }  income\_msg.msg\_type = LAST\_MESSAGE;  msgsnd(mid, (struct msgbuf \*) &income\_msg, msg\_length, 0);  return NULL;  }  int main()  {  int msgQuery\_ID;  char pathname[] = "main.c";  key\_t key;  //Creating MSGQ  if((key =ftok(pathname, 0)) < 0)  {  printf("Cant generate key\n");  exit(-1);  }  if((msgQuery\_ID = msgget(key, 0666 | IPC\_CREAT)) < 0)  {  printf("Cant get msg query \n");  exit(-1);  }  //Creating 2-st thread  pthread\_t secondThID;  int result = pthread\_create(&secondThID, NULL, secondThread, (void\*) msgQuery\_ID);  if(result != 0){  printf ("Error on thread create, return value = %d\n", result);  exit(-1);  }  printf("2st Thread created, sending messges\n");  //Compiling and sending msgs  struct msgbuf msg;  msg.msg\_type = 1;  for(int i = 0; i < 4; i++)  {  msg.msg\_text = (rand() % (9 - 1 + 1)) + 1;  printf("%i msg: %i\n", i, msg.msg\_text);  if (msgsnd(msgQuery\_ID, (struct msgbuf \*) &msg, msg\_length, 0) < 0)  {  printf("Can\'t send message to queue\n");  msgctl(msgQuery\_ID, IPC\_RMID, (struct msqid\_ds \*) NULL);  exit(-1);  }  };  sleep(1);  int swap\_counter = 0;  msgrcv(msgQuery\_ID, (struct msgbuf\*) &msg, msg\_length, 0,0);  int printLen = msg.msg\_text;  int temp[printLen];  int ctr = 0;  while((msgrcv(msgQuery\_ID, (struct msgbuf\*) &msg, msg\_length, 0,0) > 0) && msg.msg\_type != LAST\_MESSAGE)  {  temp[ctr] = msg.msg\_text;  if(ctr == printLen-1)  {  for(int i = 0; i < printLen; i++)  {  printf("%i ", temp[i]);  }  ctr = 0;  swap\_counter++;  printf("\n");  }  else  {  ctr++;  swap\_counter++;  }  }  if(swap\_counter == 0)  {  printf("Why the commutator is silent...\n");  msgctl(msgQuery\_ID, IPC\_RMID,(struct msqid\_ds \*) NULL);  exit(-2);  }  int reference = 1;  for(int i = 1; i <= printLen; i++)  {  reference = reference\*i;  }  printf("Swaps done (reference %i): %i\n", reference,swap\_counter/3);  pthread\_join(secondThID, (void \*\*)NULL);  //pid\_t childId = fork();  msgctl(msgQuery\_ID, IPC\_RMID,(struct msqid\_ds \*) NULL);  return 0;  } |

1. Основной поток создаёт очереди сообщений.
2. Поток 1 создаёт поток 2, параметры потока 2 – ID MSG;
3. Поток 1 отправляет 4 числа.
4. Поток 2 принимает 4 числа, сортирует их, убирает одинаковые элементы и сообщает количество оставшихся потоку 1.
5. Поток 2 выполняет перестановки, отправляя по одному сообщению типа **int** потоку 1.
6. Поток 1 знает, сколько будет символов в строке, но считывает сообщения, ожидая сообщение завершающего типа.
7. Поток 2 передал завершающее.
8. Поток 1 вывел количество перестановок на экран.
9. ???
10. PROFIT!

**Результат выполнения программ**

|  |
| --- |
| 2st Thread created, sending messges  0 msg: 2  1 msg: 8  2 msg: 1  3 msg: 8  Recieved 1 msg: 2  Recieved 2 msg: 8  Recieved 3 msg: 1  Recieved 4 msg: 8  1 2 8  1 8 2  2 1 8  2 8 1  8 1 2  8 2 1  Swaps done (reference 6): 6 |
|  |

**Вывод**

Вывод: выполняя лабораторную работу, я научился работать с методами синхронизации процессов с помощью семафоров.