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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Федотов Дмитрий Борисович/

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 <algorithm>  typedef struct {  long mtype;  int msg\_data[4];  } msg\_struct;  void swap\_values(int\* first, int\* second)  {  int temp = \*first;  \*first = \*second;  \*second = \*first;  }  int get\_factorial(int i)  {  if (i == 0)  return 1;  else  return i \* get\_factorial(i - 1);  }  void\* pthread\_work(void\* args)  {  msg\_struct pthread\_msg;  int msg\_id = \*((int \*) args);  ssize\_t msg\_len = msgrcv(msg\_id, &pthread\_msg, sizeof(pthread\_msg), 0, 0);  printf("Thread received random numbers. Ready to work!\n");    do  {  msgsnd(msg\_id, &pthread\_msg, sizeof(pthread\_msg), 0);  } while(std::next\_permutation(pthread\_msg.msg\_data, pthread\_msg.msg\_data + 4));  return 0;  }  void print\_received\_msg(msg\_struct\* msg)  {  for (int i = 0; i < 4; i++)  printf("%d ", msg->msg\_data[i]);  printf("\n");  }  int compare\_int\_value(const void\* a, const void\* b)  {  return \*((int\*) a) - \*((int\*) b);  }  int main(void)  {  pthread\_t thread;  int random\_numbers[4];    srand(time(NULL));  for (int i = 0; i < 4; i++)  random\_numbers[i] = rand() % 1000;  qsort(random\_numbers, 4, sizeof(int), compare\_int\_value);  printf("Random numbers is: ");  for (int i = 0; i < 4; i++)  printf("%i ", random\_numbers[i]);  printf("\n");  const size\_t sem\_count = 10;  int msg\_id = msgget(IPC\_PRIVATE, 0600 | IPC\_CREAT);  if (msg\_id < 0)  {  perror("Error with msgget()!\n");  return -1;  }  printf("Message id = %i\n", msg\_id);  int res1 = pthread\_create(&thread, NULL, pthread\_work, &msg\_id);  // prepare to send values to thread  msg\_struct parent\_msg;  parent\_msg.mtype = 7;  for (int i = 0; i < 4; i++)  parent\_msg.msg\_data[i] = random\_numbers[i];  msgsnd(msg\_id, &parent\_msg, sizeof(parent\_msg), 0);  int count\_msg\_received = 0;  int count\_msg\_to\_receive = get\_factorial(4);  printf("Needs: %d\n", count\_msg\_to\_receive);  while (count\_msg\_received != count\_msg\_to\_receive)  {  ssize\_t msg\_len = msgrcv(msg\_id, &parent\_msg, sizeof(parent\_msg), 0, 0);  printf("Received %li bytes: ", msg\_len);  print\_received\_msg(&parent\_msg);  count\_msg\_received++;  }  msgctl(msg\_id, IPC\_RMID, NULL);  return 0;  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| >> make  g++ main.cpp -lpthread -o main  ./main  Random numbers is: 149 273 456 555  Message id = 35  Needs: 24  Thread received random numbers. Ready to work!  Received 24 bytes: 149 273 456 555  Received 24 bytes: 149 273 555 456  Received 24 bytes: 149 456 273 555  Received 24 bytes: 149 456 555 273  Received 24 bytes: 149 555 273 456  Received 24 bytes: 149 555 456 273  Received 24 bytes: 273 149 456 555  Received 24 bytes: 273 149 555 456  Received 24 bytes: 273 456 149 555  Received 24 bytes: 273 456 555 149  Received 24 bytes: 273 555 149 456  Received 24 bytes: 273 555 456 149  Received 24 bytes: 456 149 273 555  Received 24 bytes: 456 149 555 273  Received 24 bytes: 456 273 149 555  Received 24 bytes: 456 273 555 149  Received 24 bytes: 456 555 149 273  Received 24 bytes: 456 555 273 149  Received 24 bytes: 555 149 273 456  Received 24 bytes: 555 149 456 273  Received 24 bytes: 555 273 149 456  Received 24 bytes: 555 273 456 149  Received 24 bytes: 555 456 149 273  Received 24 bytes: 555 456 273 149 |

**Вывод**

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