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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Прокофьев Максим Юрьевич/

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**

|  |
| --- |
| Lab07.c |
| #include <stdio.h>  #include <sys/shm.h>  #include <stdlib.h>  #include <pthread.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <string.h>  #include <fcntl.h>  #include <sys/stat.h>  #include <sys/sem.h>  #include <sys/msg.h>  #include <sys/ipc.h>  struct StrmsgBuf  {  long mtype;  int numbers[4];  };  struct Arguments  {  int msgId;  };  int comparator(const void\* i, const void\* j)  {  return \*(int \*)i - \*(int \*)j;  }  void swap(int\* numbers, int i, int j)  {  int s = numbers[i];  numbers[i] = numbers[j];  numbers[j] = s;  }  int nSet(int\* a, int n)  {  int j = n - 2;  while (j != -1 && a[j] >= a[j + 1])  {  j--;  }  if (j == -1)  return 0;  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 1;  }  void\* child\_thread(void\* Arg)  {  struct Arguments\* Arguments = (struct Arguments\*)Arg;  int msgId = Arguments->msgId;  struct StrmsgBuf Message;  msgrcv(msgId, &Message, sizeof(Message), 1, 0);  qsort(Message.numbers, 4, sizeof(int), comparator);  Message.mtype = 101;  msgsnd(msgId, &Message, sizeof(Message), 0);  while (nSet(Message.numbers, 4))  {  Message.mtype = 101;  msgsnd(msgId, &Message, sizeof(Message), 0);  }  Message.numbers[0] = -1;  Message.mtype = 101;  msgsnd(msgId, &Message, sizeof(Message), 0);  return NULL;  }  void\* parent\_thread(void\* Arg)  {  int numbers[4];  printf("Initial data: ");  srand((unsigned)(time(0)));  for (size\_t i = 0; i < 4; ++i)  {  numbers[i] = rand() % 10;  printf("%d ", numbers[i]);  }  printf("\n");  struct Arguments\* args = (struct Arguments\*)Arg;  int msgId = args->msgId;  struct StrmsgBuf Message;  memcpy(Message.numbers, numbers, 4 \* sizeof(int));  Message.mtype = 1;  msgsnd(msgId, &Message, sizeof(Message), 0);  int k = 1;  do  {  size\_t size = msgrcv(msgId, &Message, sizeof(Message), 101, 0);  if (Message.numbers[0] != -1)  {  printf("Combination %i: ", k);  for (size\_t i = 0; i < 4; ++i)  {  printf("%d ", Message.numbers[i]);  }  printf("\n");  k++;  }  } while (Message.numbers[0] != -1);  msgctl(msgId, IPC\_RMID, NULL);  return NULL;  }  int main()  {  int msgId = msgget(IPC\_PRIVATE, 0600 | IPC\_CREAT);  if (msgId < 0)  {  perror("error with msgget()");  return -1;  }    struct Arguments \*args;  args->msgId = msgId;  pthread\_t ParentThread, ChildThread;  pthread\_create(&ParentThread, NULL, parent\_thread, (void \*)args);  pthread\_create(&ChildThread, NULL, child\_thread, (void\*)args);  pthread\_join(ParentThread, NULL);  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| …/program  Initial data: 7 5 2 0  Combination 1: 0 2 5 7  Combination 2: 0 2 7 5  Combination 3: 0 5 2 7  Combination 4: 0 5 7 2  Combination 5: 0 7 2 5  Combination 6: 0 7 5 2  Combination 7: 2 0 5 7  Combination 8: 2 0 7 5  Combination 9: 2 5 0 7  Combination 10: 2 5 7 0  Combination 11: 2 7 0 5  Combination 12: 2 7 5 0  Combination 13: 5 0 2 7  Combination 14: 5 0 7 2  Combination 15: 5 2 0 7  Combination 16: 5 2 7 0  Combination 17: 5 7 0 2  Combination 18: 5 7 2 0  Combination 19: 7 0 2 5  Combination 20: 7 0 5 2  Combination 21: 7 2 0 5  Combination 22: 7 2 5 0  Combination 23: 7 5 0 2  Combination 24: 7 5 2 0 |

**Вывод**

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