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

«Средства Systen V IPC. Организация работы с разделяемой памятью. Понятие потоков в UNIX»

Выполнил студент группы 9091

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

Подпись ФИО

Принял преподаватель

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

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работы**

Цель работы: познакомиться с механизмом обмена данными между потоками процессов - разделяемой памяться. Познакомться с потоками thread в соответствии стандарта POSIX (pthread).

**Задание на разделяемую память**

**Исходный текст программ**

|  |
| --- |
| sender.c |
| #include <stdio.h>  #include <sys/shm.h>  #include <stdlib.h>  using namespace std;  int main()  {  const int ArraySize = 20;    const size\_t memSize = 1024; //alias of PAGE\_SIZE  int memId = shmget(IPC\_PRIVATE, memSize, 0600|IPC\_CREAT|IPC\_EXCL);  printf("shmid= %i\n", memId);  if(memId <= 0)  {  printf("Fatal error trying shmget()\n");  return -1;  };  int \*mem = (int \*)shmat(memId, 0, 0);  if(mem == NULL)  {  printf("Fatal error trying shmat()\n");  return -2;  };    for(int i = 0; i < ArraySize; i++)  {  \*mem = rand();  mem++;  }  char callbuf[1024];  sprintf(callbuf, "./receiver %i", memId);  system(callbuf);  return 0;  } |

|  |
| --- |
| receiver.c |
| #include <stdio.h>  #include <sys/shm.h>  #include <stdlib.h>  using namespace std;  int main(int argv, char \*argc[])  {  const int ArraySize = 20;  if(argv <= 1)  {  printf("Not enough input params");  return -1;  };  char \*paramStr = argc[1];  int memId = atoi(paramStr);  if (memId == 0)  {  printf("incorrect parameter string: %s\n", paramStr);  return -2;  }  printf("receiving the memory data: shmid = %i\n", memId);  int \*mem = (int \*)shmat(memId, 0, 0);  if (NULL == mem)  {  printf("error with shmat()\n");  return -3;  };  printf("RECEIVED ARRAY:\n");  for(int i = 0; i < ArraySize; i++)  {  printf("%i\n",mem[i]); //using compile conversion x[n] = \*(x+n)  };  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;  };  };  };    printf("SORTED ARRAY:\n");  for(int i = 0; i < ArraySize; i++)  {  printf("%i\n",mem[i]); //using compile conversion x[n] = \*(x+n)  };    char callbuf[1024];  sprintf(callbuf, "ipcrm -m %i", memId);  system(callbuf);    return 0;  } |

**Результат выполнения программ**

|  |
| --- |
| Вывод в терминале |
| shmid= 4  receiving the memory data: shmid = 4  RECEIVED ARRAY:  1804289383  846930886  1681692777  1714636915  1957747793  424238335  719885386  1649760492  596516649  1189641421  1025202362  1350490027  783368690  1102520059  2044897763  1967513926  1365180540  1540383426  304089172  1303455736  SORTED ARRAY:  304089172  424238335  596516649  719885386  783368690  846930886  1025202362  1102520059  1189641421  1303455736  1350490027  1365180540  1540383426  1649760492  1681692777  1714636915  1804289383  1957747793  1967513926  2044897763 |

**Задание на потоки**

**Исходный текст программ**

|  |
| --- |
| threader2.c |
| #include <pthread.h>  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  void \*firstThread(void \*dummy)  {  for(int i = 1; i <= 10; i++)  {  printf("Hello Threads (%i)\n", i);  sleep(1);  };  return NULL;  }  void \*secondThread(void \*dummy)  {  for(int i = 1; i <= 12; i++)  {  printf("This is iterration %i\n", i);  sleep(2);  };  return NULL;  }  int main()  {  pthread\_t firstThid, secondThid;    int result;    result = pthread\_create(&firstThid,NULL, firstThread, NULL);  if(result != 0){  printf ("Error on thread create, return value = %d\n", result);  exit(-1);  }  printf("1st Thread created, thid = %d\n", firstThid);  result = pthread\_create(&secondThid,NULL, secondThread, NULL);  if(result != 0){  printf ("Error on thread create, return value = %d\n", result);  exit(-1);  }  printf("2nd Thread created, thid = %d\n", secondThid);    pthread\_join(firstThid, (void \*\*)NULL);  pthread\_join(secondThid, (void \*\*)NULL);    return 0;  } |
|  |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| 1st Thread created, thid = -1129540032  2nd Thread created, thid = -1137932736  This is iterration 1  Hello Threads (1)  Hello Threads (2)  This is iterration 2  Hello Threads (3)  Hello Threads (4)  This is iterration 3  Hello Threads (5)  Hello Threads (6)  This is iterration 4  Hello Threads (7)  Hello Threads (8)  This is iterration 5  Hello Threads (9)  Hello Threads (10)  This is iterration 6  This is iterration 7  This is iterration 8  This is iterration 9  This is iterration 10  This is iterration 11  This is iterration 12 |

**Вывод**

Вывод: выполняя лабораторную работу, я познакомился с разделяемой памятью и с потоками в соответствии стандарта POSIX (pthread).