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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Лопатин Даниил Михайлович/

Подпись ФИО

Принял преподаватель

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

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работ:**

Познакомиться с организацией работы с разделяемой памяти. Познакомиться с нитями исполнения (thread):

Задание по теме «разделяемая память»:

1. Написать две программы на С или С++.
2. Первая программа должна сформировать массив из 20 случайных чисел. Для массива должна быть выделена область в разделяемой памяти. Затем эта программа должна запустить вторую программу и передать ей ключ для доступа к общей области памяти в качестве параметра командной строки.
3. Вторая программа должна прочитать из общего массива все числа и вывести их на экран.
4. Затем она должна отсортировать их по возрастанию и вывести результат сортировки на экран.

Задание по теме «потоки»:

1. Написать многопоточную программу. Основной поток (который начинается в функции main) должен породить два новых потока, дождаться их завершения и закончить работу.
2. Первый из новых потоков должен вывести на экран "Hello Threads (1)", "Hello Threads (2)" и т.д., всего 10 раз.
3. Второй поток должен выводить в цикле строки "This is iteration 1", "This is iteration 2", ... "This is iteration 12".
4. Каждый из порожденных потоков после вывода каждой строки должен делать системный вызов sleep() с параметром 1 для первого потока и параметром 2 для второго потока (чтобы можно было успевать наблюдать переключение между потоками).

**Задание по теме «разделяемая память»**

**Исходный текст программ:**

|  |
| --- |
| sender.c |
| #include "stdio.h"  #include "sys/shm.h"  #include "stdlib.h"  #include "sys/ipc.h"  #include "time.h"  int main(void)  {  const size\_t memSize = 80;    int memId = shmget(IPC\_PRIVATE, memSize, 0600 | IPC\_CREAT | IPC\_EXCL);  printf("shmid = %i\n", memId);  if (memId <= 0)  {  printf("error with shmid()\n");  return -1;  }  int\* mem = (int\* )shmat(memId, 0, 0);  if (NULL == mem)  {  printf("error with shmat()\n");  return -2;  }  srand(time(NULL));  for (int i = 0; i < 20; i++)  \*(mem + i) = rand() % 100;  char callbuf[1024];  sprintf(callbuf, "./receiver %i", memId);  system(callbuf);  return 0;  } |

|  |
| --- |
| receiver.c |
| #include "stdio.h"  #include "sys/shm.h"  #include "stdlib.h"  void print\_array\_int(int\* arr)  {  for (int i = 0; i < 20; i++)  printf("%i ", \*(arr + i));  printf("\n");  }  int compare\_int\_value(const void\* a, const void\* b)  {  return \*((int\*) a) - \*((int\*) b);  }  int main(int argv, char\* argc[])  {  if (argv <= 1)  {  printf("not enough params\n");  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("receiving next array:\n");  print\_array\_int(mem);  qsort(mem, 20, 4, compare\_int\_value);  printf("sorted array:\n");  print\_array\_int(mem);  return 0;  } |

|  |
| --- |
| Output Terminal |
| >>make  gcc sender.c -o sender  gcc receiver.c -o receiver  ./sender  shmid = 2  receiving the memory data: shmid = 2  receiving next array:  78 35 51 88 78 82 50 35 47 4 22 31 53 5 37 47 48 83 1 87  sorted array:  1 4 5 22 31 35 35 37 47 47 48 50 51 53 78 78 82 83 87 88 |

**Задание на тему «потоки»**

**Исходный текст программы:**

|  |
| --- |
| threader.c |
| #include "stdio.h"  #include "stdlib.h"  #include "pthread.h"  #include <unistd.h>  void\* print\_message\_func1()  {  for (int i = 1; i < 11; i++)  printf("Hello Thread(%i)\n", i), sleep(1);  return 0;  }  void\* print\_message\_func2()  {  for (int i = 1; i < 13; i++)  printf("This is iteration %i\n", i), sleep(2);  return 0;  }  int main(void)  {  pthread\_t thrd1, thrd2;  int res1 = pthread\_create(&thrd1, 0, print\_message\_func1, 0);  int res2 = pthread\_create(&thrd2, 0, print\_message\_func2, 0);  int iret1, iret2;  pthread\_join(thrd1, (void\*\*) &iret1);  pthread\_join(thrd2, (void\*\*) &iret2);  return 0;  } |

|  |
| --- |
| Output Terminal |
| >>make  gcc threader.c -lpthread -o threader  ./threader  This is iteration 1  Hello Thread(1)  Hello Thread(2)  This is iteration 2  Hello Thread(3)  Hello Thread(4)  This is iteration 3  Hello Thread(5)  Hello Thread(6)  This is iteration 4  Hello Thread(7)  Hello Thread(8)  This is iteration 5  Hello Thread(9)  Hello Thread(10)  This is iteration 6  This is iteration 7  This is iteration 8  This is iteration 9  This is iteration 10  This is iteration 11  This is iteration 12 |

**Вывод:**

Выполняя данную лабораторную работу, я познакомился с разделяемой памятью и потоками в соответствии со стандартом POSIX (thread).