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

\_\_\_\_\_\_\_\_\_\_\_\_\_/Малинин Никита Валерьевич/

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>  #include <time.h>  int main()  {  const size\_t memSize = 1024;    int memId = shmget(IPC\_PRIVATE, memSize, 0600|IPC\_CREAT|IPC\_EXCL);    printf("shmid = %i\n", memId);  if (memId <= 0)  {  printf("error with shmget()\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>  int compareValue(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;  }    for(int i = 0; i < 20; i ++)  {  printf("%i ", \*(mem + i));  }  printf("\n");  qsort(mem, 20, 4, compareValue);  for(int i = 0; i < 20; i ++)  {  printf("%i ", \*(mem + i));  }  printf("\n");  return 0;  } |

**Результат выполнения программ**

|  |
| --- |
| Вывод в терминале |
| >> make  gcc receiver.c -o receiver  gcc sender.c -o sender  >> ./sender  shmid = 32792  receiving the memory data: shmid = 32792  9 55 75 59 11 31 97 66 43 61 6 18 28 40 11 40 16 48 88 59  6 9 11 11 16 18 28 31 40 40 43 48 55 59 59 61 66 75 88 97 |

**Задание на потоки**

**Исходный текст программ**

|  |
| --- |
| thread.c |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #include <unistd.h>  void\* printMessageThreads ()  {  for( int i=0; i < 10; i++)  {  printf("Hello Threads %i \n", i),sleep(1);  }  }  void\* printmMessageIteration ()  {  for( int i=0; i < 12; i++)  {  printf("This is iteration %i \n", i),sleep(2);  }  }  int main()  {  pthread\_t thread1, thread2;  int res1 = pthread\_create(&thread1, NULL, printMessageThreads, NULL);  int res2 = pthread\_create(&thread2, NULL, printmMessageIteration, NULL);  int iret1, iret2;  pthread\_join(thread1, (void \*\*)&iret1);  pthread\_join(thread2, (void \*\*)&iret2);    return 0;  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| >> make  gcc thread.c -lpthread -o thread  >>./thread  Hello Threads 0  This is iteration 0  Hello Threads 1  This is iteration 1  Hello Threads 2  Hello Threads 3  This is iteration 2  Hello Threads 4  Hello Threads 5  This is iteration 3  Hello Threads 6  Hello Threads 7  This is iteration 4  Hello Threads 8  Hello Threads 9  This is iteration 5  This is iteration 6  This is iteration 7  This is iteration 8  This is iteration 9  This is iteration 10  This is iteration 11 |

**Вывод**

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