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 «Процессы в ОС UNIX»

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: Познакомится с методами работы с процессами в OS Linux;

**Исходный текст программ**

|  |
| --- |
| processmaker.c |
| #include <sys/shm.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  #include <stdio.h>  //Allocating 1024 (4096 on fact) bytes for our tasks and return pointer to begging of it + changing outer value of memId  int \*allocateSharedMemory(size\_t memSize, int &memId)  {  memId = shmget(IPC\_PRIVATE, memSize, 0600|IPC\_CREAT|IPC\_EXCL);    if(memId <= 0)  {  perror("Fatal error trying shmget()\n");  return NULL;  }  int \*memPtr = (int \*)shmat(memId, 0, 0);    if(memPtr == NULL)  {  perror("Fatal error trying shmat()\n");  }    return memPtr;  }  //Fills array with (n = ArraySize) pseudo-random numbers;  void fillArray(int \*MemPtr, int ArraySize)  {  for(int i = 0; i < ArraySize; i++)  {  \*MemPtr = rand();  printf("%i is %i\n", i+1, \*MemPtr);  MemPtr++;  }  }  void sortArray(int \*mem, int ArraySize)  {  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;  };  };  };  }  int main()  {  const int ArraySize = 20;  const size\_t memSize = 1024; //alias of PAGE\_SIZE  int memId;    int \*sharedMemPtr = allocateSharedMemory(memSize, memId);  printf("ID of shared memory = %d\n", memId);    printf("Unsorted Array:\n");  fillArray(sharedMemPtr, ArraySize);  pid\_t childId = fork();  if(childId < 0)  {  perror("fork error");  }  else if(childId > 0)  {  printf("Parent: This is main process. Child id = %d\n", childId);  printf("Parent: awaiting for child.\n");  waitpid(childId, NULL, 0);    printf("Parent: execution is continued.\n");  }  else  {  printf("Child: This is child process. My parrent id = %d\n",getppid());  sortArray(sharedMemPtr, ArraySize);    int \*MemPtr = sharedMemPtr;  printf("Sorted Array\n");  for(int i = 0; i < ArraySize; i++)  {  printf("%i is %i\n", i+1, \*MemPtr);  MemPtr++;  };  exit(0);  };    if(shmctl(memId,IPC\_RMID, 0) == 0)  {  printf("Shared memory cleared!\n");  }  else  {  perror("Shared memory control problem!");  };      return 0;  } |

**Результат выполнения программ**

|  |
| --- |
| Вывод в терминале (output.txt) |
| ID of shared memory = 26  Unsorted Array:  1 is 1804289383  2 is 846930886  3 is 1681692777  4 is 1714636915  5 is 1957747793  6 is 424238335  7 is 719885386  8 is 1649760492  9 is 596516649  10 is 1189641421  11 is 1025202362  12 is 1350490027  13 is 783368690  14 is 1102520059  15 is 2044897763  16 is 1967513926  17 is 1365180540  18 is 1540383426  19 is 304089172  20 is 1303455736  Parent: This is main process. Child id = 4771  Parent: awaiting for child.  Child: This is child process. My parrent id = 4770  Sorted Array  1 is 304089172  2 is 424238335  3 is 596516649  4 is 719885386  5 is 783368690  6 is 846930886  7 is 1025202362  8 is 1102520059  9 is 1189641421  10 is 1303455736  11 is 1350490027  12 is 1365180540  13 is 1540383426  14 is 1649760492  15 is 1681692777  16 is 1714636915  17 is 1804289383  18 is 1957747793  19 is 1967513926  20 is 2044897763  Parent: execution is continued.  Shared memory cleared! |
|  |

**Вывод**

Вывод: выполняя лабораторную работу, я научился работать с процессами и обеспечивать обмен информации между процессами посредствам разделяемой памяти.