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 «Организация взаимодействия процессов через pipe и FIFO в 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 обмена информации между процессами в OS Linux;

**Исходный текст программ**

|  |
| --- |
| FIFO-pipe.c |
| #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <unistd.h>  #include <string.h>  #include <fcntl.h>  #include <wait.h>  void fillArray(int \*Array, int ArraySize)  {  for(int i = 0; i < ArraySize; i++)  {  Array[i] = rand();  };  printf("Filled array with %i pseudo-rand int\n", ArraySize);  }  void printArray(int \*Array, int ArraySize)  {  printf("Current array:\n", ArraySize);  for(int i = 0; i < ArraySize; i++)  {  printf("%i : %i\n", i+1, Array[i]);  };  }  void sortArray(int \*ArrayPtr, int ArraySize)  {  int temp;  for (int i = 0; i < ArraySize - 1; i++)  {  for (int j = 0; j < ArraySize - i - 1; j++)  {  if (ArrayPtr[j] < ArrayPtr[j + 1])  {  temp = ArrayPtr[j];  ArrayPtr[j] = ArrayPtr[j + 1];  ArrayPtr[j + 1] = temp;  };  };  };  }  int main (int argc, char \*argv[])  {  const int ArraySize = atoi(argv[1]);  size\_t sizeWrited, sizeFormed;  int ChildPID;    if(ArraySize < 0)  {  printf("Wrong array size!\n");  return 1;  }    const size\_t rawArraySize = sizeof(int) \* ArraySize;  int p[2]; pipe(p);    int Array[ArraySize];  printf("Created array of %i int\n", ArraySize);    fillArray(Array, ArraySize);  printArray(Array, ArraySize);    int fd;  char fifoName[] = "FIFO.fifo";  (void)umask(0);  unlink(fifoName);  if(mknod(fifoName, S\_IFIFO | 0666, 0) < 0)  {  printf("Can't create FIFO");  exit (-2);  }    if((ChildPID = fork()) < 0)  {  printf("Can\'t fork child\n");  exit(-3);  }  else  if(ChildPID > 0)  {  if((fd = open(fifoName, O\_WRONLY)) < 0)  {  printf("Can\'t open FIFO for writing\n");  exit(-1);  }  sizeFormed = rawArraySize;  sizeWrited = write(fd, Array, rawArraySize);  if(sizeWrited != sizeFormed )  {  printf("Can\'t write all string to FIFO\n");  exit(-1);  }  close(fd);  waitpid(ChildPID, NULL, 0);  printf("Fork ended\n");  }  else  {    if((fd = open(fifoName, O\_RDONLY)) < 0)  {  printf("Can\'t open FIFO for reading\n");  exit(-1);  }    size\_t size;  int ChildArray[ArraySize];    size = read(fd, ChildArray, rawArraySize);  if(size < 0){  printf("Can\'t read string\n");  exit(-1);  }  close(fd);    sortArray(ChildArray,ArraySize);  size = write(p[1],ChildArray,rawArraySize);  if(size != rawArraySize)  {  printf("PIPE sending FAILED\n");  exit(-1);  }  close(p[1]);  exit(0);  }    read(p[0],Array,rawArraySize);  close(p[0]);    unlink(fifoName);    printArray(Array,ArraySize);    return 0;  } |

**Результат выполнения программ**

|  |
| --- |
| make  gcc FIFO-pipe.c -o fifo-pipe  ./fifo-pipe 8  Created array of 8 int  Filled array with 8 pseudo-rand int  Current array:  1 : 1804289383  2 : 846930886  3 : 1681692777  4 : 1714636915  5 : 1957747793  6 : 424238335  7 : 719885386  8 : 1649760492  Fork ended  Current array:  1 : 1957747793  2 : 1804289383  3 : 1714636915  4 : 1681692777  5 : 1649760492  6 : 846930886  7 : 719885386  8 : 424238335 |
|  |

**Вывод**

Вывод: выполняя лабораторную работу, я научился работать с методами передачи данных через именованные и неименованные pipe;