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

«Организация взаимодействия процессов через pipe и FIFO 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 механизмами организации взаимодействия процессов через pipe и FIFO в UNIX.

**Исходный текст программы**

|  |
| --- |
| main.c |
| #include "stdio.h"  #include "stdlib.h"  #include "unistd.h"  #include "time.h"  #include "sys/wait.h"  #include "sys/stat.h"  #include "fcntl.h"  #include <sys/cdefs.h>  void printArray(int\* arr, int size)  {  for (int i = 0; i < size; i++)  printf("%d ", arr[i]);  printf("\n");  }  int compareValue(const void\* a, const void\* b)  {  return \*((int\*) a) - \*((int\*) b);  }  int main(int argv, char\* argc[])  {  if(argv <= 1)  {  printf("Error! Not enough params!\n");  return -1;  }  int arraySize = atoi(argc[1]);  int\* array = malloc(sizeof(int) \* arraySize);  srand(time(NULL));  for(int i = 0; i < arraySize; i++)  {  array[i] = rand() % 100;  printf("%d ", array[i]);  }  printf("\n");  int fdPipe[2], fdFifo;  size\_t size;  if(pipe(fdPipe) < 0)  {  printf("Error! Pipe was not created!\n");  return -1;  }  const char\* fileName = "fileFIFO";  (void)umask(0); // обнуление маски создания файлов текущего процесса для соответсвия прав доступа FIFO параметру вызова mknod  if(mknod(fileName, S\_IFIFO | 0666, 0) < 0)  {  printf("Error! FIFO was not created!\n");  return -1;  }  pid\_t childProcess = fork();  if(childProcess == -1)  {  printf("Error! Child cannot be forked!\n");  return -1;  }  else if(childProcess > 0)  {  if((fdFifo = open(fileName, O\_WRONLY)) < 0)  {  printf("Cannot open FIFO for writing\n");  return -1;  }  size = write(fdFifo, array, sizeof(int) \* arraySize);  if(size != sizeof(int) \* arraySize)  {  printf("Cannot write all string to FIFO\n");  return -1;  }  close(fdFifo);  close(fdPipe[1]);  size = read(fdPipe[0], array, sizeof(int) \* arraySize);  if(size < 0)  {  printf("Cannot read string \n");  return -1;  }  printf("Child sent to parent this array: ");  printArray(array, arraySize);  close(fdPipe[0]);    }  else  {  if((fdFifo = open(fileName, O\_RDONLY)) < 0)  {  printf("Cannot open FIFO for reading!\n");  return -1;  }  int\* sortArray = malloc(sizeof(int) \* arraySize);  size = read(fdFifo, sortArray, sizeof(int) \* arraySize);  if(size < 0)  {  printf("Cannot read string\n");  return -1;  }  qsort(sortArray, arraySize, sizeof(int), compareValue);  close(fdFifo);  close(fdPipe[0]);    size = write(fdPipe[1], sortArray, sizeof(int) \* arraySize);  if(size != sizeof(int) \* arraySize)  {  printf("Cannot write all string\n");  return -1;  }  close(fdPipe[1]);  free(sortArray);  printf("Child process has ended!\n");  return 0;  }  char deleteFIFOfile[124];  sprintf(deleteFIFOfile, "rm %s", fileName);  system(deleteFIFOfile);  free(array);  return 0;  } |

**Результат выполнения программы**

|  |
| --- |
| Вывод в терминале |
| >> make  gcc main.c -o main  ./main 10  50 22 42 51 51 96 24 45 56 17  Child process has ended!  Child sent to parent this array: 17 22 24 42 45 50 51 51 56 96 |

**Вывод**

Вывод: выполняя лабораторную работу, я познакомился с механизмами организации взаимодействия процессов через pipe и FIFO в UNIX.