**Московский Авиационный Институт**

**(Национальный Исследовательский Университет)**

**Факультет информационных технологий и прикладной математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работ №2 по курсу**

**«Операционные системы»**

**УПРАВЛЕНИЕ ПРОЦЕССАМИ В ОС UNIX**

**Студент: Баранников Степан Алексеевич**

**Группа: М80–201Б-18**

**Вариант: 03**

**Преподаватель: Соколов Андрей Алексеевич**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_**

**Дата: \_\_\_\_\_\_\_\_\_\_\_**

**Подпись: \_\_\_\_\_\_\_\_\_\_\_**

**Москва, 2019**

**1.** **Постановка задачи**

На вход программе подается число, для которого нужно рекурсивно посчитать сумму всех чисел до этого числа включительно.

**3.Листинг программы**

|  |
| --- |
| #include <stdlib.h> #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h> #include <stdbool.h> #include "itoa.h"  int sum(int a){  if(a!=1){  pid\_t pid;  int pipe\_1[2], pipe\_2[2], res=0, buff\_a = a-1;  if((pipe(pipe\_1) == -1) || (pipe(pipe\_2) == -1)){  write(STDERR\_FILENO, "ERROR: pipe\n", sizeof "ERROR: pipe\n" - 1);  exit(EXIT\_FAILURE);  }  pid = fork();  if(pid > 0){ //Родитель  close(pipe\_1[0]);  write(pipe\_1[1], &buff\_a, sizeof(buff\_a));  close(pipe\_1[1]);  int status;  wait(&status);  printf(">>%i\n", WEXITSTATUS(status));  close(pipe\_2[1]);  read(pipe\_2[0], &res, sizeof(res));  close(pipe\_2[0]);  return res+a;  }  else if(pid < 0){  write(STDERR\_FILENO, "ERROR: fork\n", sizeof "ERROR: fork\n" - 1);  exit(EXIT\_FAILURE);  }  else{ //Ребенок  int get = 0;  close(pipe\_1[1]);  read(pipe\_1[0], &get, sizeof(get));  close(pipe\_1[0]);  get = sum(get);  close(pipe\_2[0]);  write(pipe\_2[1], &get, sizeof(get));  close(pipe\_2[1]);  exit(42);  }  }  else  return a; }  int main() {  bool flag = true;  int input;  char save[100];  int save\_len;  int sol = 0;  char sol\_str[100];  int sol\_str\_len;  write(STDOUT\_FILENO, "1. Чтобы вычислить сумму от 1 до n введите число\n", sizeof "1. Чтобы вычислить сумму от 1 до n введите число\n" - 1);  write(STDOUT\_FILENO, "2. Чтобы завершить программу, введите '0'\n", sizeof "2. Чтобы завершить программу, введите '0'\n" - 1);  while(flag){  write(STDOUT\_FILENO, "> ", sizeof "> " - 1);  save\_len = read(STDIN\_FILENO, save, 100); //считывание команды  save[save\_len-1] = '\0';  input = atoi(save);  if(input == 0){  flag = false;  write(STDOUT\_FILENO, "Выход из программы...\n", sizeof "Выход из программы...\n" - 1);  }  else{  if(input < 0)  write(STDOUT\_FILENO, "Ошибка, введено некорректное число!\n", sizeof "Ошибка, введено некорректное число!\n" - 1);  else if(input > 0){  sol = sum(input);  sol\_str\_len = itoa(sol, sol\_str);  write(STDOUT\_FILENO, "Итоговая сумма = ", sizeof "Итоговая сумма = " - 1);  write(STDOUT\_FILENO, sol\_str, sol\_str\_len - 1);  write(STDOUT\_FILENO, "\n", sizeof "\n" - 1);  }  }  } } |

**4.** **Вывод**

С помощью fork() можно разделить задачи для более эффективного решения. Обмен данными родительского и дочернего процессов можно обеспечить с помощью pipe или же с помощью wait() --- exit(), что говорит нам о достаточном количестве возможностей fork().