**Московский Авиационный Институт**

**(Национальный Исследовательский Университет)**

**Факультет информационных технологий и прикладной математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работ №4 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. Обеспечить общение процессов через mmap.

**2.Листинг программы**

|  |
| --- |
| #include <stdlib.h> #include <stdio.h> #include <stdint.h> #include <fcntl.h> #include <sys/stat.h> #include <sys/mman.h> #include <unistd.h> #include <sys/types.h>  #include <sys/wait.h> #include <stdbool.h> #include <string.h> #include "itoa.h"  //////////////////Sum func char\* sum(char\* map){  int a = atoi(map);  if(a!=1){  pid\_t pid;  int res, buff\_a = a - 1;  char \*buff\_map = map;  int len = itoa(buff\_a, buff\_map);   pid = fork();  if(pid > 0){ //Родитель  wait(0);  res = atoi(map) + a;  itoa(res, map);  return map;  }  else if(pid < 0){  write(STDERR\_FILENO, "ERROR: fork\n", sizeof "ERROR: fork\n" - 1);  exit(EXIT\_FAILURE);  }  else{ //Ребенок  map = sum(map);  exit(0);  }  }  else  return map; }  int main() {  char \*map;  map = mmap(0, 17, PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, -1, 0);  int n = 0;  printf("Enter a sequence number: ");  scanf("%d", &n);  if(n<0 || n == 0)  printf("You should give the number, which is more than zero and zero\n");  else{  printf("there\n");  int len = itoa(n, map);  printf("there\n");  printf("%d\n", atoi(sum(map)));  }  return 0; } |

**3.** **Вывод**

*Цитата вывода ЛР2:  
 “*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().”  
 Но, на удивление есть еще один метод общения процессов и составляющих программы: Маппинг. Маппинг - это выделение виртуальной памяти, с помощью которого можно читать файл как строку, или загружать и выгружать обычные строки (не из файла).