Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**Процессы в ОС UNIX**

Лабораторная работа №4 по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_ Баранов А. С.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** познакомиться с механизмами разделённой памяти и многопоточности.

**Задание:**

Объединить программы первой части предыдущей лабораторной в одну программу.

Вначале программа должна сгенерировать 20 случайных чисел, разместить их в разделяемой памяти и вывести на экран.

Затем требуется породить новый процесс, который выполнит сортировку самостоятельно, без запуска других программ (в отличие от предыдущей лабораторной).

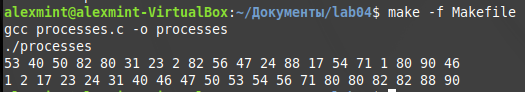
Второй процесс должен отсортировать числа и также вывести их на экран.

Первый процесс должен дождаться завершения работы второго, после чего самостоятельно выполнить освобождение выделенной разделяемой памяти и всех прочих выделенных ресурсов (при наличии таковых).

**Содержание файла processes.c:**

|  |
| --- |
| #include <stdio.h> |
| #include <sys/wait.h> |
| #include <sys/shm.h> |
| #include <stdlib.h> |
| #include <unistd.h> |
| #include <time.h> |
|  |
| void Sort(int data[], int lenD) |
| { |
| int j = 0; |
| int tmp = 0; |
| for(int i=0; i<lenD; i++){ |
| j = i; |
| for(int k = i; k<lenD; k++){ |
| if(data[j]>data[k]){ |
| j = k; |
| } |
| } |
| tmp = data[i]; |
| data[i] = data[j]; |
| data[j] = tmp; |
| } |
| } |
| const size\_t memSize = 20\*sizeof(int); |
|  |
| int compare\_ints(const void \*a, const void \*b) |
| { |
| return (\*((int \*)a) - \*((int \*)b)); |
| } |
|  |
| void sortAndPrint(int \*mem) |
| { |
| Sort(mem, 20); |
| for (int i = 0; i < 20; i++) |
| { |
| printf("%i ", mem[i]); |
| } |
| printf("\n"); |
| } |
|  |
| int main() |
| { |
| srand((unsigned)(time(0))); |
|  |
| int memId = shmget(IPC\_PRIVATE, memSize, 0600 | IPC\_CREAT | IPC\_EXCL); |
|  |
| int \*mem = (int \*)shmat(memId, 0, 0); |
|  |
| for (int i = 0; i < 20; i++) |
| { |
| mem[i] = rand() % 100; |
| } |
|  |
| for (int i = 0; i < 20; i++) |
| { |
| printf("%d ", mem[i]); |
| } |
|  |
| printf("\n"); |
|  |
| int child\_id = fork(); |
|  |
| if (child\_id == 0) |
| { |
| sortAndPrint(mem); |
| } |
| else |
| { |
| waitpid(child\_id, NULL, 0); |
| } |
|  |
| shmdt(mem); |
|  |
| return 0; |
| } |

**Результат, выведенный на экран:**



**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с механизмами запуска новых процессов в UNIX.