Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №7**по дисциплине «Операционные системы»  
на тему «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ В ОС LINUX»

Выполнил:  
Студент *Альков В.С.*  
Группа *И407Б*  
  
Преподаватель:  
*Шимкун В.В.*

Санкт-Петербург  
2022 г.

**Цель работы** - освоить использование в программах некоторых возможностей ОС Linux: параллельных процессов, перенаправления, семафоров, разделяемой памяти, сокетов, сигналов.

**Задание**: написать программу для вычисления суммы элементов матрицы. Матрица вводится из файла. Вычисление сумм элементов каждой строки производить в отдельных процессах. Для передачи данных в родительский процесс использовать механизм именованного канала (FIFO).

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

int\*\* readMatrix(char\* filename, int\* row, int\* column);

void printMatrix(int \*\*m, int r, int c);

int\*\* freeMatrix(int\*\* m, int\* r, int\* c);

int sum\_row(int\* row, int size);

void sum\_rows(int process\_count, int\*\* m, int count, int column\_count, int row\_offset, int fd\_fifo);

char\* readFifo(char\* filename, int size);

int main()

{

char fifo\_name[] = "/tmp/fifo\_0001.2";

unlink(fifo\_name);

int fd\_fifo;

if((fd\_fifo = mknod(fifo\_name, S\_IFIFO | S\_IRUSR | S\_IWUSR, 0)) == -1)

{

fprintf(stderr, "Невозможно создать fifo\n");

return 0;

}

if((fd\_fifo=open(fifo\_name, O\_RDWR)) < 0)

{

fprintf(stderr, "Невозможно открыть fifo\n");

return 0;

}

int row, column, process\_count=1, i=0;

int\*\* m=readMatrix("/home/user/alkov/lab7/matrix", &row, &column);

if(!m)

{

fprintf(stderr, "Не удалось прочитать матрицу\n");

return 0;

}

printMatrix(m, row, column);

for(i=0; i<row/process\_count; i+=process\_count)

sum\_rows(process\_count, m, 0, column, i, fd\_fifo);

sum\_rows(row%process\_count, m, 0, column, i, fd\_fifo);

int\* sum = (int\*)readFifo(fifo\_name, sizeof(int)\*row\*2);

for(i=0; i<row\*2; i+=2)

printf("%d) %d\n",sum[i]+1, sum[i+1]);

close(fd\_fifo);

return 0;

}

int\*\* readMatrix(char\* filename, int\* row, int\* column)

{

int newline, count\_r, count\_c, negative, c, segmentation\_error = 0;

\*row = \*column = count\_r = count\_c = 1;

FILE\* f = fopen(filename, "r");

int \*\*m= (int\*\*)calloc(sizeof(int\*), 1), \*\*m\_copy;

m[0] = (int\*)calloc(sizeof(int), 1);

if(!f)

{

printf("Не удалось открыть файл: %s\n", filename);

return 0;

}

while(1)

{

newline = negative = 0;

while((c = fgetc(f)) != EOF && (c < '0' || c > '9' || c == ' '))

{

if (c == '\n') newline = 1;

if (c == '-') negative = 1;

}

if(newline)

count\_r++, count\_c=1;

if(c == EOF)

break;

else

ungetc(c,f);

if(negative)

ungetc('-',f);

if(count\_r>\*row)

{

m\_copy = m;

if(!((m = realloc(m, sizeof(int\*)\*count\_r)) && (m[count\_r-1] = calloc(\*column, sizeof(int)))))

{

segmentation\_error = 1;

break;

}

m\_copy = m, \*row = count\_r;

}

if(count\_c > \*column)

{

for(int i = 0; i<\*row; i++)

{

if((m[i] = realloc(m[i], sizeof(int)\*count\_c)) == 0)

{

printf("2");

segmentation\_error = 1;

break;

};

for(int j = \*column; j<count\_c; j++)

m[i][j] = 0;

}

\*column = count\_c;

}

if((fscanf(f, "%d", &m[\*row-1][count\_c-1])) == 0)

break;

count\_c++;

}

fclose(f);

if(segmentation\_error)

{

printf("segmentation error");

m\_copy = m = freeMatrix(m\_copy, row, column);;

}

return m;

}

void printMatrix(int \*\*m, int r, int c)

{

for(int i=0; i<r; i++)

{

for(int j=0; j<c; j++)

printf("%d\t", m[i][j]);

printf("\n");

}

}

int\*\* freeMatrix(int\*\* m, int\* r, int\* c)

{

for(int i = 0; i<\*r; i++)

free(m[i]);

free(m);

\*r = \*c= 0;

return 0;

}

int sum\_row(int\* row, int size)

{

int sum = 0;

for(int i=0; i<size; i++)

sum+=row[i];

return sum;

}

void sum\_rows(int process\_count, int\*\* m, int count, int column\_count, int row\_offset, int fd\_fifo)

{

if(count++ >= process\_count)

return;

pid\_t pid=fork();

if(pid == -1)

{

perror("fork");

return;

}

if(!pid)

{

int res[2];

res[0] = row\_offset+count-1, res[1] = sum\_row(m[res[0]], column\_count);

write(fd\_fifo, res, 2\*sizeof(int));

sum\_rows(process\_count, m, count, column\_count, row\_offset, fd\_fifo);

exit(0);

}

if(pid)

{

wait(&pid);

if(count != 1)

exit(0);

}

}

char\* readFifo(char\* filename, int size)

{

int fd\_fifo;

char\* data = NULL;

if((fd\_fifo=open(filename, O\_RDWR)) == -1)

fprintf(stderr, "Невозможно открыть fifo\n");

else

if((data = (char\*)malloc(size)) == NULL)

fprintf(stderr, "Невозможно выделить память\n");

else

if((read(fd\_fifo, data, size)) == -1)

fprintf(stderr, "Невозможно прочесть из FIFO\n");

else

return data;

if(!data)

free(data);

return NULL;

}