Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М80-201Б-22

Студент: Чапалда М.О.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 14.12.23

Москва, 2023

**Постановка задачи**

**Группа вариантов 2.**

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Родительский процесс выводит результат в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

**Вариант 9.**

В файле записаны команды вида:«число число число<endline>». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат “кладет” в map-file. Если происходит деление на 0, то дочерний (и родительский) процесс завершают свою работу с ошибкой. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип float. Количество чисел может быть произвольным.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int execl(const char \**path*, const char \**arg0*, ... /\*, (char \*)0 \*/); - заменяет текущий образ процесса новым образом
* pid\_t wait(int \*status); – приостаналивает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится.
* void\* mmap(void \*, size\_t, int, int, int, off\_t) - выделяет память или отображает файлы (или устройства) в памяти.
* int munmap(void\*, size\_t) - удаляет сопоставление с выделенной памятью.
* int ftruncate(int, off\_t) - приводит файл к заданному размеру.
* int shm\_open(const char \*, int, ...) - инициализирует область памяти
* int shm\_unlink(const char\* name) — разрывает связь между областью памяти и заданным ей именем

Сначала пользователь в качестве аргумента командной строки пишет имя файла, которое будет использоваться для открытия файла с таким же именем на чтение. Если строка введена корректно, и файл с таким именем существует, то создается дочерний процесс. После чего для дочернего процесса подменятся стандартный ввод, которым теперь является открытый файл. В дочернем процессе инициализируется область памяти именем «laba3» и туда записываются обработанные им данные. После завершения дочернего процесса родительский процесс читает данные из этой же области памяти и печатает их в терминал. В случае, если родительский процесс прочитает значение -1, на экран будет выведено сообщение «Attempt to divide by zero» и выполнение программы завершится ошибкой

**Код программы**

**parent.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include "stddef.h"

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#include <semaphore.h>

#define MEMORY\_NAME "laba3" *//определение имени разделяемой памяти*

#define DATA\_SIZE 256 *//определение размера данных*

#define MEMORY\_SIZE 8192 *//определение размера разделяемой памяти*

void check\_error(bool expression, char\* message) { *//объявление функции для проверки ошибок*

    if (expression) { *//если истинно*

        write(STDOUT\_FILENO, message, strlen(message) \* sizeof(char)); *//вывести сообщение об ошибке*

        write(STDOUT\_FILENO, "\\n", 1);

        exit(-1); *//завершить программу*

    }

}

typedef struct { *//объявление пользовательской структуры данных*

    size\_t size; *//размер данных*

    float data[DATA\_SIZE]; *//массив данных типа float*

} res;

int main (int argc, char\* argv[]) {

    pid\_t pid; *//объявление переменной для хранения идентификатора процесса*

    FILE \*fp = NULL; *//объявление указателя на файл*

    if (argc != 2) { *//проверка наличия имени файла*

        write(1, "Wrong arguments\\n", 17); *//вывод сообщения об ошибке*

        exit(EXIT\_FAILURE); *//завершение программы*

    }

    pid = fork(); *//создание дочернего процесса*

    if (pid == -1) { *//проверка на ошибку при создании дочернего процесса*

        perror("fork"); *//вывод сообщения об ошибке*

        return -1; *//возврат отрицательного статуса*

    }

    else if (pid == 0) { *//дочерний процесс*

        fp = freopen(argv[1], "r", stdin); *//открытие файла для чтения и перенаправление стандартного ввода на него*

        check\_error(fp == NULL, "Can't open file"); *//проверка на ошибки при открытии файла*

        execl("./child", "/.child", NULL); *//выполнить программу-ребенка*

        perror("execl"); *//вывод сообщения об ошибке, если execl завершился неудачно*

        return 1; *//возврат отрицательного статуса*

    }

    else { *//родительский процесс*

        wait(0); *// Ожидание завершения дочернего процесса*

        int fd = shm\_open(MEMORY\_NAME, O\_RDONLY, S\_IRUSR | S\_IWUSR); *//открытие разделяемой памяти для чтения*

        check\_error(fd == -1, "Can't oped shared memory file"); *//проверка на ошибки при открытии разделяемой памяти*

        res \*addr = mmap(NULL, MEMORY\_SIZE, PROT\_READ, MAP\_SHARED, fd, 0); *//отображение разделяемой памяти в адресное пространство процесса*

        check\_error(addr == (void\*) -1, "Mmap error"); *//проверка на ошибки при отображении памяти*

        for (int i = 0; i < addr->size; i++) { *//цикл по данным в разделяемой памяти*

            if (addr->data[i] == -1) { *//если обнаружена попытка деления на ноль*

                printf("Attempt to divide by zero\\n"); *//вывод сообщения об ошибке*

                break;

            }

            printf("%f\\n", addr->data[i]); *//вывод данных*

        }

        munmap(addr, MEMORY\_SIZE); *//удаление отображения разделяемой памяти из адресного пространства процесса*

        shm\_unlink(MEMORY\_NAME); *//удаление имени разделяемой памяти из системы после завершения всех отображений памяти на нее*

        close(fd); *//закрытие файлового дескриптора разделяемой памяти*

    }

    return 0;

}

**Child.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include "stddef.h"

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#include <semaphore.h>

#include <math.h>

#define MEMORY\_NAME "laba3"

#define DATA\_SIZE 256

#define MEMORY\_SIZE 8192

void check\_error(bool expression, char\* message) { *//функция на проверку ошибок*

    if (expression) { *//если истинно*

        write(STDOUT\_FILENO, message, strlen(message) \* sizeof(char)); *//вывести сообщение об ошибке*

        write(STDOUT\_FILENO, "\\n", 1);

        exit(-1);

    }

}

typedef struct { *//объявление пользовательской структуры данных*

    size\_t size; *//размер данных*

    float data[DATA\_SIZE]; *//массив данных типа float*

} res;

int main() {

    int fd = shm\_open(MEMORY\_NAME, O\_EXCL | O\_CREAT | O\_RDWR, S\_IRUSR | S\_IWUSR); *//создание разделяемой памяти для чтения и записи*

    check\_error(fd == -1, "Can't open shared memory file"); *//рроверка на ошибки при создании разделяемой памяти*

    if (ftruncate(fd, MEMORY\_SIZE) == -1) { *//установка размера разделяемой памяти*

        printf("File is too large"); *//файл слишком большой*

    }

    res \*addr = mmap(NULL, MEMORY\_SIZE, PROT\_WRITE, MAP\_SHARED, fd, 0); *//отображение разделяемой памяти в адресное пространство процесса для записи*

    check\_error(addr == (void\*)-1, "Mmap error"); *//проверка на ошибки при отображении памяти*

    addr->size = 0; *//установка начального размера данных*

    char c; *//объявление переменной для хранения символа*

    bool not\_end = true; *//объявление переменной для проверки окончания ввода данных*

    float nmbr = 0; *//объявление переменной для хранения числа*

    float result = 0; *//объявление переменной для хранения результата вычислений*

    float first = 0; *//объявление переменной для хранения первого числа в выражении*

    int k = 0; *//объявление переменной для хранения степени десятки при вводе дробных чисел*

    float dot = 0; *//объявление переменной для хранения дробной части числа*

    int count = 0; *//объявление переменной для подсчета количества чисел в выражении*

    float numbers[100]; *//объявление массива для хранения чисел*

    do { *//начало цикла ввода данных*

        if (not\_end) { *//если ввод данных не завершен*

            if (c <= '9' && c >= '0') { *//если символ является цифрой*

                if (nmbr != 0 && (floor(nmbr) != nmbr)) { *//если число не целое*

                    if (c == '0') { *//если следующая цифра равна нулю*

                        k += 1; *//увеличить степень десятки на 1*

                        nmbr += 0; *//добавить ноль к числу*

                    }

                    else {

                        dot = c - '0'; *//получить дробную часть числа из символа*

                        nmbr = (nmbr + 0.1) + (dot / pow(10, k + 1)); *//добавить дробную часть числа к числу с учетом степени десятки*

                    }

                }

                else {

                    nmbr = nmbr \* 10 + c - '0'; *//умножить число на 10 и добавить новую цифру к числу*

                }

            }

            if (c == '.') { *//если символ является точкой (разделителем целой и дробной частей числа)*

                nmbr = nmbr - 0.1; *//уменьшить число на 0.1 для корректного расчета дробной части числа*

            }

            if (c == ' ' || c == '\\n' || c == EOF) { *//если символ является пробелом, символом новой строки или концом файла*

                numbers[count] = nmbr; *//добавить число в массив чисел*

                nmbr = 0; *//сбросить значение переменной числа*

                count++; *//увеличить количество чисел в выражении*

                if (c == '\\n' || c == EOF) { *//если символ является символом новой строки или концом файла*

                    first = numbers[0]; *//присвоить первому числу значение первого числа в выражении*

                    for (int i = 1; i < count; i++) { *//цикл по оставшимся числам в выражении*

                        if (numbers[i] == 0) { *//если обнаружена попытка деления на ноль*

                            addr->data[addr->size++] = -1; *//добавить -1 в массив результатов и увеличить размер данных в разделяемой памяти*

                        }

                        result = first / numbers[i]; *//выполнить операцию деления и сохранить результат*

                        first = result;

                    }

                    first = 0;

                    count = 0;

                }

            }

        }

        if (c == '\\n' || c == EOF) { *//если символ является символом новой строки или концом файла*

            addr->data[addr->size++] = result; *//добавить результат операции в массив результатов и увеличить размер данных в разделяемой памяти*

            result = 0;

            k = 0;

            dot = 0;

            not\_end = true;

        }

    } while((scanf("%c", &c)) > 0); *// Продолжать цикл до тех пор, пока пользователь продолжает ввод данных*

    return 0;

}

**Протокол работы программы**

**Тестирование:**

900.000122

1.000000

14.273480

0.666667

90.000000

38.136364

4329.239258

2880.000000

Attempt to divide by zero

Strace:

mariyaach@LAPTOP-HSJSPL2E:~/оси/lab3$ strace -f ./parent file.txt

execve("./parent", ["./parent", "file.txt"], 0x7ffd3432d0b0 /\* 30 vars \*/) = 0

brk(NULL) = 0x5582ee46d000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffce83ae8b0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f0bf11bd000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=16299, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 16299, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f0bf11b9000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f0bf0f91000

mmap(0x7f0bf0fb9000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f0bf0fb9000

mmap(0x7f0bf114e000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f0bf114e000

mmap(0x7f0bf11a6000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f0bf11a6000

mmap(0x7f0bf11ac000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f0bf11ac000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f0bf0f8e000

arch\_prctl(ARCH\_SET\_FS, 0x7f0bf0f8e740) = 0

set\_tid\_address(0x7f0bf0f8ea10) = 13899

set\_robust\_list(0x7f0bf0f8ea20, 24) = 0

rseq(0x7f0bf0f8f0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f0bf11a6000, 16384, PROT\_READ) = 0

mprotect(0x5582ecbbf000, 4096, PROT\_READ) = 0

mprotect(0x7f0bf11f7000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f0bf11b9000, 16299) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 13900 attached

, child\_tidptr=0x7f0bf0f8ea10) = 13900

[pid 13900] set\_robust\_list(0x7f0bf0f8ea20, 24 <unfinished ...>

[pid 13899] wait4(-1, <unfinished ...>

[pid 13900] <... set\_robust\_list resumed>) = 0

[pid 13900] openat(AT\_FDCWD, "file.txt", O\_RDONLY) = 3

[pid 13900] dup3(3, 0, 0) = 0

[pid 13900] close(3) = 0

[pid 13900] execve("./child", ["/.child"], 0x7ffce83aea90 /\* 30 vars \*/) = 0

[pid 13900] brk(NULL) = 0x557e40f3f000

[pid 13900] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff2ab27300) = -1 EINVAL (Invalid argument)

[pid 13900] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f10d827e000

[pid 13900] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 13900] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 13900] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=16299, ...}, AT\_EMPTY\_PATH) = 0

[pid 13900] mmap(NULL, 16299, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f10d827a000

[pid 13900] close(3) = 0

[pid 13900] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 13900] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 13900] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

[pid 13900] mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f10d8193000

[pid 13900] mmap(0x7f10d81a1000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f10d81a1000

[pid 13900] mmap(0x7f10d821d000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f10d821d000

[pid 13900] mmap(0x7f10d8278000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f10d8278000

[pid 13900] close(3) = 0

[pid 13900] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 13900] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

[pid 13900] pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 13900] pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

[pid 13900] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

[pid 13900] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

[pid 13900] pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 13900] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f10d7f6b000

[pid 13900] mmap(0x7f10d7f93000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f10d7f93000

[pid 13900] mmap(0x7f10d8128000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f10d8128000

[pid 13900] mmap(0x7f10d8180000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f10d8180000

[pid 13900] mmap(0x7f10d8186000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f10d8186000

[pid 13900] close(3) = 0

[pid 13900] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f10d7f68000

[pid 13900] arch\_prctl(ARCH\_SET\_FS, 0x7f10d7f68740) = 0

[pid 13900] set\_tid\_address(0x7f10d7f68a10) = 13900

[pid 13900] set\_robust\_list(0x7f10d7f68a20, 24) = 0

[pid 13900] rseq(0x7f10d7f690e0, 0x20, 0, 0x53053053) = 0

[pid 13900] mprotect(0x7f10d8180000, 16384, PROT\_READ) = 0

[pid 13900] mprotect(0x7f10d8278000, 4096, PROT\_READ) = 0

[pid 13900] mprotect(0x557e405ac000, 4096, PROT\_READ) = 0

[pid 13900] mprotect(0x7f10d82b8000, 8192, PROT\_READ) = 0

[pid 13900] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 13900] munmap(0x7f10d827a000, 16299) = 0

[pid 13900] openat(AT\_FDCWD, "/dev/shm/laba3", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0600) = 3

[pid 13900] ftruncate(3, 8192) = 0

[pid 13900] mmap(NULL, 8192, PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f10d827c000

[pid 13900] newfstatat(0, "", {st\_mode=S\_IFREG|0644, st\_size=84, ...}, AT\_EMPTY\_PATH) = 0

[pid 13900] getrandom("\x49\x11\x1a\x6c\x22\x9d\xe5\xfd", 8, GRND\_NONBLOCK) = 8

[pid 13900] brk(NULL) = 0x557e40f3f000

[pid 13900] brk(0x557e40f60000) = 0x557e40f60000

[pid 13900] read(0, "9 00.01 1\n10 2.5 4\n1000 7.006 2 "..., 4096) = 84

[pid 13900] read(0, "", 4096) = 0

[pid 13900] exit\_group(0) = ?

[pid 13900] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 13900

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=13900, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=1} ---

openat(AT\_FDCWD, "/dev/shm/laba3", O\_RDONLY|O\_NOFOLLOW|O\_CLOEXEC) = 3

mmap(NULL, 8192, PROT\_READ, MAP\_SHARED, 3, 0) = 0x7f0bf11bb000

munmap(0x7f0bf11bb000, 8192) = 0

unlink("/dev/shm/laba3") = 0

close(3) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

После завершения этой лабораторной работы, я осознала, что существует эффективная альтернатива использованию pipe - это file mapping. Это очень интересная тема, но я понимаю, что для ее полного освоения требуется более детальное изучение, так как в этой области существует множество тонкостей и интересных аспектов, которые могут вызвать затруднения.