

## Операционные системы и сети

### Лабораторная работа №2 Джугели Дмитрий ПИН-31Д

**Задание 1.** Запишите назначение основных опций компилятора **gcc** (**-c**, **-S**, **-E**, **-o**). Воспользуйтесь командой **\$man gcc** или **\$info gcc**

Опции компилятора GCC:

- c: компилирует исходный код в объектный файл, но не выполняет линковку
- S: генерирует ассемблерный код, но не выполняет компиляцию
- E: выполняет только препроцессинг и выводит результат на стандартный вывод
- o позволяет указать имя выходного файла

#### Задание 2.

**2.1.** Напишите и выполните программу **Hello**, выводящую строку **"Hello, world"**.

Код Hello.c

```
#include <stdio.h>

int main() {
    printf("Hello, world\n");
    return 0;
}
```

```
ant_daddy@Dmitriy ~ % gcc Hello.c -o Hello
ant_daddy@Dmitriy ~ % ./Hello
Hello, world
ant_daddy@Dmitriy ~ %
```

**2.2.** Определите главное отличие **stdout** от **stderr**. Для этого выполните программу, добавив

**... while (1) {fprintf(stdout, "a"); sleep(1);} ...**

Отличие между `stdout` и `stderr` заключается в том, что `stdout` предназначен для стандартного вывода данных (обычно на экран), а `stderr` предназначен для вывода сообщений об ошибках.


Затем выполните программу, заменив **`stdout`** на **`stderr`**. Запишите Ваши выводы в отчет.

```
ant_daddy@Dmitriy ~ % gcc program.c -o program
./program
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa^C
ant_daddy@Dmitriy ~ % gcc program.c -o program
./program
```

В случае использования вывода `stdout` в консоль выводится «aaa..»

В случае использования вывода `stderr` не отображается на экране

**2.3.** Напишите и выполните программу **`obr`**, состоящую из двух модулей: **`obr.c`** содержит функцию **`double obr (int i)`**, возвращающую число, обратное числу `i`; **`main.c`** содержит функцию **`main()`**, которая запрашивает у пользователя целое число `i` и выводит значение **`obr(i)`**.



```
#include <stdio.h>

double obr(int i) {
    return 1.0 / i;
}
```

```
main.c
#include <stdio.h>

double obr(int i);

int main() {
    int i;
    printf("Введите целое число: ");
    scanf("%d", &i);

    printf("Обратное число: %f\n", obr(i));

    return 0;
}
```

```
ant_daddy@Dmitriy ~ % gcc -c obr.c
gcc -c main.c
[gcc -o obr main.o obr.o
ant_daddy@Dmitriy ~ % ./obr
Введите целое число: 100
Обратное число: 0.010000
ant_daddy@Dmitriy ~ % ./obr
Введите целое число: 10
Обратное число: 0.100000
ant_daddy@Dmitriy ~ %
```

### Задание 3.

**3.1.** Создайте файл **Makefile** для программы 2.3, предварительно удалив файлы **\*.o** и **obr** из текущего каталога. Затем выполните команду **\$make**. Запишите в отчет текст файла **Makefile** с пояснениями и команды, выполненные утилитой **make**.

```
all: or
```

```
obr: main.o obr.o //компиляция исп файл obr из main.o obr.o
```

```
    gcc -o obr main.o obr.o // компиляция и линковка main.o и obr.o в
исполняемый файл "obr"
```

```
main.o: main.c // компиляция объектного файла main.o из исх main.c
```

gcc -c main.c // компиляция исх файла main.c в объектн main.o

obr.o: obr.c. // компиляция объектного файла obr.o из исходного obr.c

gcc -c obr.c // компиляция исходного файла obr.c в объектный файл obr.o

clean:

rm -f \*.o obr. // удаление всех файлов с расширением .o и obr

```
makefile.4: *** missing separator. Stop.
ant_daddy@Dmitriy 123 % make
gcc -c main.c
gcc -c obr.c
gcc -o obr main.o obr.o
ant_daddy@Dmitriy 123 %
```



main.c



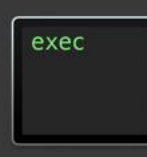
Makefile



obr.c



main.o



obr



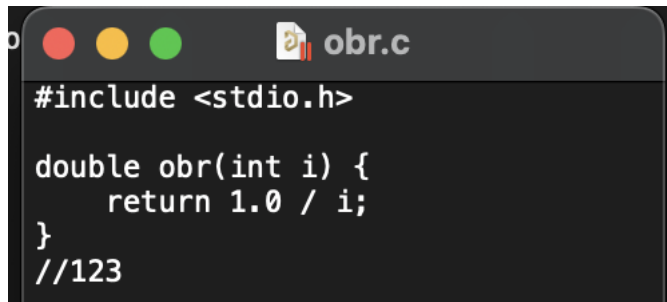
ant\_daddy — obr — 80x24

```
Last login: Fri Apr 26 17:30:01 on ttys001
ant_daddy@Dmitriy ~ % /Users/ant_daddy/123/obr ; exit;
Введите целое число: 100
Обратное число: 0.010000

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Процесс завершен]
```

**3.2.** Незначительно модифицируйте файл **obr.c**, выполните команду **\$make** и запишите команды, выполненные утилитой **make**.

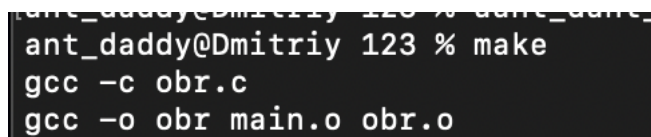
A screenshot of a code editor window titled 'obr.c'. The code inside is: 

```
#include <stdio.h>

double obr(int i) {
    return 1.0 / i;
}

//123
```

Незначительное изменение файла obr.c

A screenshot of a terminal window. The prompt is 'ant\_daddy@Dmitriy 123 %'. The user has entered 'make'. The output shows the compilation process: 

```
gcc -c obr.c
gcc -o obr main.o obr.o
```

Выполнение команды make

**Задание 4.** Запишите в отчет текст примеров и результаты их тестирования с параметрами.

**4.1.**

```
/* Ex4_1.c Печать параметров командной строки*/
#include <stdio.h>
main ( int argc, char *argv[] )
{   if   ( argc < 2 )
    {   printf( "Usage : %s parameter\n", argv[0] ) ;
        exit ( 1 ) ;
    }

    printf("Starting program %s \n", argv[0] ) ;
    printf("with %d parameter(s)\n", argc-1 ) ;
    printf("First parameter is %s\n", argv[1] ) ;
    exit ( 0 ) ;
}
```

```
ant_daddy@Dmitriy 123 % gcc -o ex4_1 Ex4_1.c

Ex4_1.c:13:37: warning: data argument not used by format string [-Wformat-extra-args]
    printf("with 1 parameter(s)\n", argc-1);
                                   ^
1 warning generated.
ant_daddy@Dmitriy 123 %
```

```
ant_daddy@Dmitriy 123 % gcc -o ex4_1 Ex4_1.c

ant_daddy@Dmitriy 123 % ./ex4_1 12
Starting program ./ex4_1
with 1 parameter(s)
First parameter is 12
ant_daddy@Dmitriy 123 %
```

## 4.2.

/\* Ex4\_3.c Печать произвольного числа параметров \*/

```
#include <stdio.h>

main(int argc, char *argv[])
{
    for ( ; *argv ; ++argv )
        printf("%s\n", *argv ) ;
}
```

```
ant_daddy@Dmitriy 123 % gcc -o ex4_3 Ex4_3.c
ant_daddy@Dmitriy 123 % ./Ex4_3 Hello
./Ex4_3
Hello
ant_daddy@Dmitriy 123 % ./Ex4_3 Hello world!
./Ex4_3
Hello
world!
ant_daddy@Dmitriy 123 % ./Ex4_3 Hello world! 152
./Ex4_3
Hello
world!
152
ant_daddy@Dmitriy 123 %
```

**4.3.** Измените предыдущий пример, задав параметры целого типа. Для преобразования строки в целое число используйте функцию **atoi**. Получить информацию о стандартных функциях C можно по команде **\$info libc**.

```
#include <stdio.h>
```

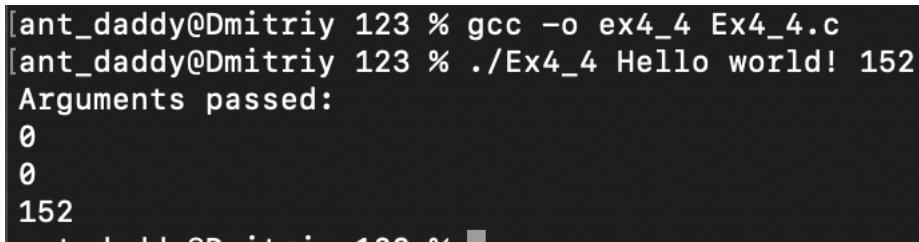
```
#include <stdlib.h>
```

```

int main(int argc, char *argv[]) {
    if (argc == 1) {
        printf("No arguments.\n");
    } else {
        printf("Arguments passed:\n");
        for (int i = 1; i < argc; i++) {
            int num = atoi(argv[i]);
            printf("%d\n", num);
        }
    }

    return 0;
}

```



```

[ant_daddy@Dmitriy 123 % gcc -o ex4_4 Ex4_4.c
[ant_daddy@Dmitriy 123 % ./Ex4_4 Hello world! 152
Arguments passed:
0
0
152
[ant_daddy@Dmitriy 123 %

```

**Задание 5.** Выполните пример, запишите в отчет его текст и результаты тестирования.

```

/* Ex5_1.c Печать переменных окружения */
#include <stdio.h>
extern char** environ;
int main ()
{
    char** var;
    for (var = environ; *var != NULL; ++var)
        printf ("%s\n", *var);
}

```

```
return 0;
```

```
}
```

```
ant_daddy@Dmitriy ant_daddy % cd 123
ant_daddy@Dmitriy 123 % gcc -o ex5_1 Ex5_1.c
ant_daddy@Dmitriy 123 % ./Ex5_1
__CFBundleIdentifier=com.apple.Terminal
TMPDIR=/var/folders/0x/cd5dmx1d3q79dh_kjnht9wj00000gn/T/
XPC_FLAGS=0x0
LaunchInstanceID=49807682-0284-42F2-8BE7-CC2498285149
TERM=xterm-256color
DISPLAY=/private/tmp/com.apple.launchd.QmizE2memQ/org.xquartz:0
SECURITYSESSIONID=186a3
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.sMHQ6H6A60/Listeners
XPC_SERVICE_NAME=0
TERM_PROGRAM=Apple_Terminal
TERM_PROGRAM_VERSION=453
TERM_SESSION_ID=A3B13597-99E4-4A64-A7EA-3E0D4B4A2B73
SHELL=/bin/zsh
HOME=/Users/ant_daddy
LOGNAME=ant_daddy
USER=ant_daddy
PATH=/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Version
s/3.10/bin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:
/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:
/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com
.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/X11/b
in:/Library/Apple/usr/bin:/usr/local/share/dotnet:~/dotnet/tools:/Library/Frame
works/Mono.framework/Versions/Current/Commands
SHLVL=1
PWD=/users/ant_daddy/123
OLDPWD=/users/ant_daddy
LANG=ru_RU.UTF-8
_=/users/ant_daddy/123/./Ex5_1
```

## Задание 6.

**6.1.** Модифицируйте приведенную ниже программу **copyfile.c** копирования файла, добавив в нее комментарии, проверку количества аргументов командной строки (при неверном количестве параметров установите код возврата=1) и обработку ошибок после системных вызовов **open** (коды возврата 2 и 3), **write** (код возврата 4). Протестируйте программу для разных ситуаций.

ПРИМЕЧАНИЕ. Для моделирования ситуации переполнения диска в качестве имени выходного файла задайте **/dev/full**.

```
#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

#define BUF_SIZE 256
```



```
int main (int argc, char *argv [])
{
int input_fd, output_fd;
int bytes_in, bytes_out;
char rec [BUF_SIZE];
input_fd = open (argv [1], O_RDONLY); /* код возврата=2 */
output_fd = open (argv [2], O_WRONLY | O_CREAT, 0666); /*Код
возврата=3 */
while ((bytes_in = read (input_fd, rec, BUF_SIZE)) > 0) {
    bytes_out = write (output_fd, rec, bytes_in);
/* Здесь код возврата=4, если число прочитанных байтов не равно
числу записанных*/
close (input_fd);
close (output_fd);
return 0;
}
```

```

ex6_1.c
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h> // Добавлен заголовочный файл для функций read и write
#define BUF_SIZE 256

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <input_file> <output_file>\n", argv[0]);
        return 1; // Код возврата=1 при неверном количестве аргументов
    }

    int input_fd, output_fd;
    int bytes_in, bytes_out;
    char rec[BUF_SIZE];

    input_fd = open(argv[1], O_RDONLY);
    if (input_fd == -1) {
        perror("Error opening input file");
        return 2; // Код возврата=2 при ошибке открытия входного файла
    }

    output_fd = open(argv[2], O_WRONLY | O_CREAT, 0666);
    if (output_fd == -1) {
        perror("Error opening output file");
        close(input_fd);
        return 3; // Код возврата=3 при ошибке открытия выходного файла
    }

    while ((bytes_in = read(input_fd, rec, BUF_SIZE)) > 0) {
        bytes_out = write(output_fd, rec, bytes_in);
        if (bytes_out != bytes_in) {
            perror("Error writing to output file");
            close(input_fd);
            close(output_fd);
            return 4; // Код возврата=4 при ошибке записи в файл
        }
    }

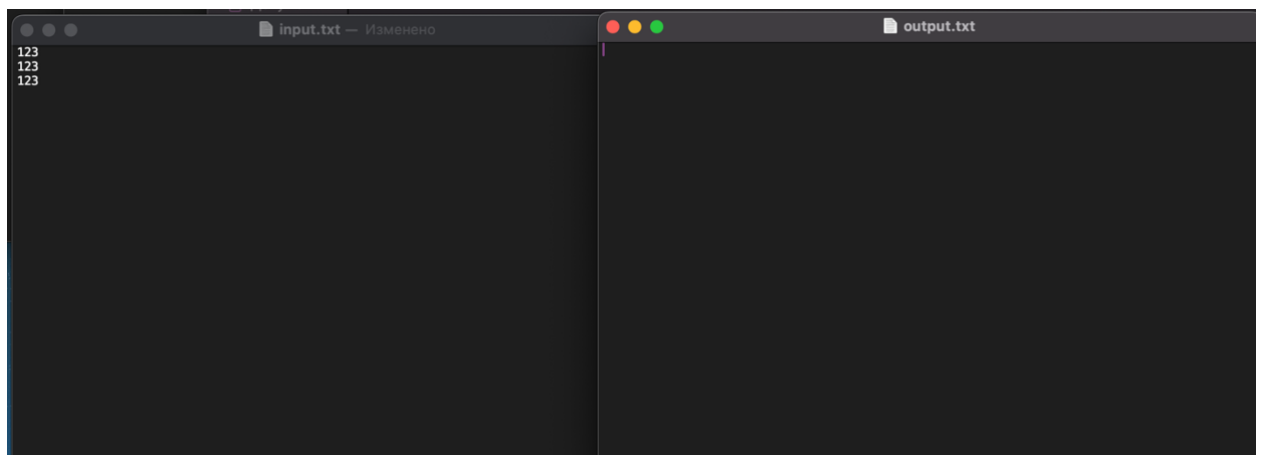
    close(input_fd);
    close(output_fd);

    return 0;
}

```

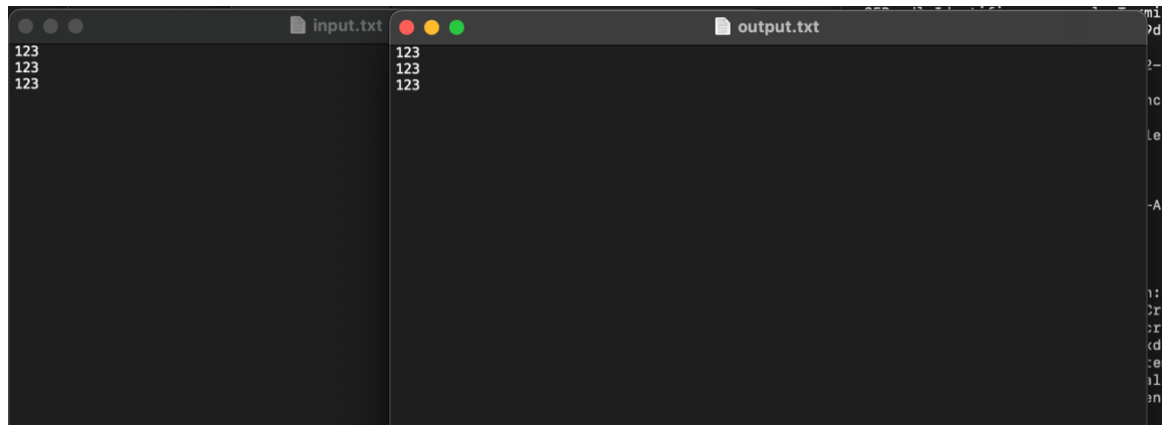
Создадим файлы input.txt(поместим в него текст)

И пустой output.txt



```
ant_daddy@Dmitriy 123 % gcc -o ex6_1 Ex6_1.c
ant_daddy@Dmitriy 123 % ./Ex6_1
Usage: ./Ex6_1 <input_file> <output_file>
ant_daddy@Dmitriy 123 % ./Ex6_1 input.txt output.txt
ant_daddy@Dmitriy 123 %
```

Получим:



**6.2.** Напишите и выполните программу **date.c**, которая: а) при помощи функции **open** создает файл **file1** с разрешением чтения/записи для всех пользователей; б) записывает в конец файла **file1** (задайте необходимый флаг при открытии файла!) текущую дату и время. Выполните программу несколько раз и выпишите в отчет содержимое **file1**.

ПРИМЕЧАНИЕ. Для получения даты в виде символьной строки, последовательно примените функции **time**, **localtime** и **asctime** (см. **man**). Перед записью в файл определите длину получившейся строки при помощи функции **strlen** (см. **info**).

```
date.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>

int main() {
    char filename[] = "file1";
    int fd = open(filename, O_RDWR | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);

    if (fd == -1) {
        perror("open");
        exit(1);
    }

    time_t rawtime;
    struct tm * timeinfo;
    char buffer[80];

    time(&rawtime);
    timeinfo = localtime(&rawtime);
    strftime(buffer, 80, "%c", timeinfo);

    int len = strlen(buffer);

    if (write(fd, buffer, len) != len) {
        perror("write");
        close(fd);
        exit(1);
    }

    close(fd);

    return 0;
}
```

```
[ant_daddy@Dmitriy 123 % gcc -o date date.c
[ant_daddy@Dmitriy 123 % gcc -o date Date.c
[ant_daddy@Dmitriy 123 % ./date
[ant_daddy@Dmitriy 123 % ./date
ant_daddy@Dmitriy 123 %
```

```
file1
Wed May 29 03:48:03 2024
```

```
file1
Wed May 29 03:48:03 2024Wed May 29 03:49:08 2024Wed May 29 03:50:28 2024
```