Факультатив по программированию на языке C

Занятие 4
Библиотеки. Основы ввода/вывода

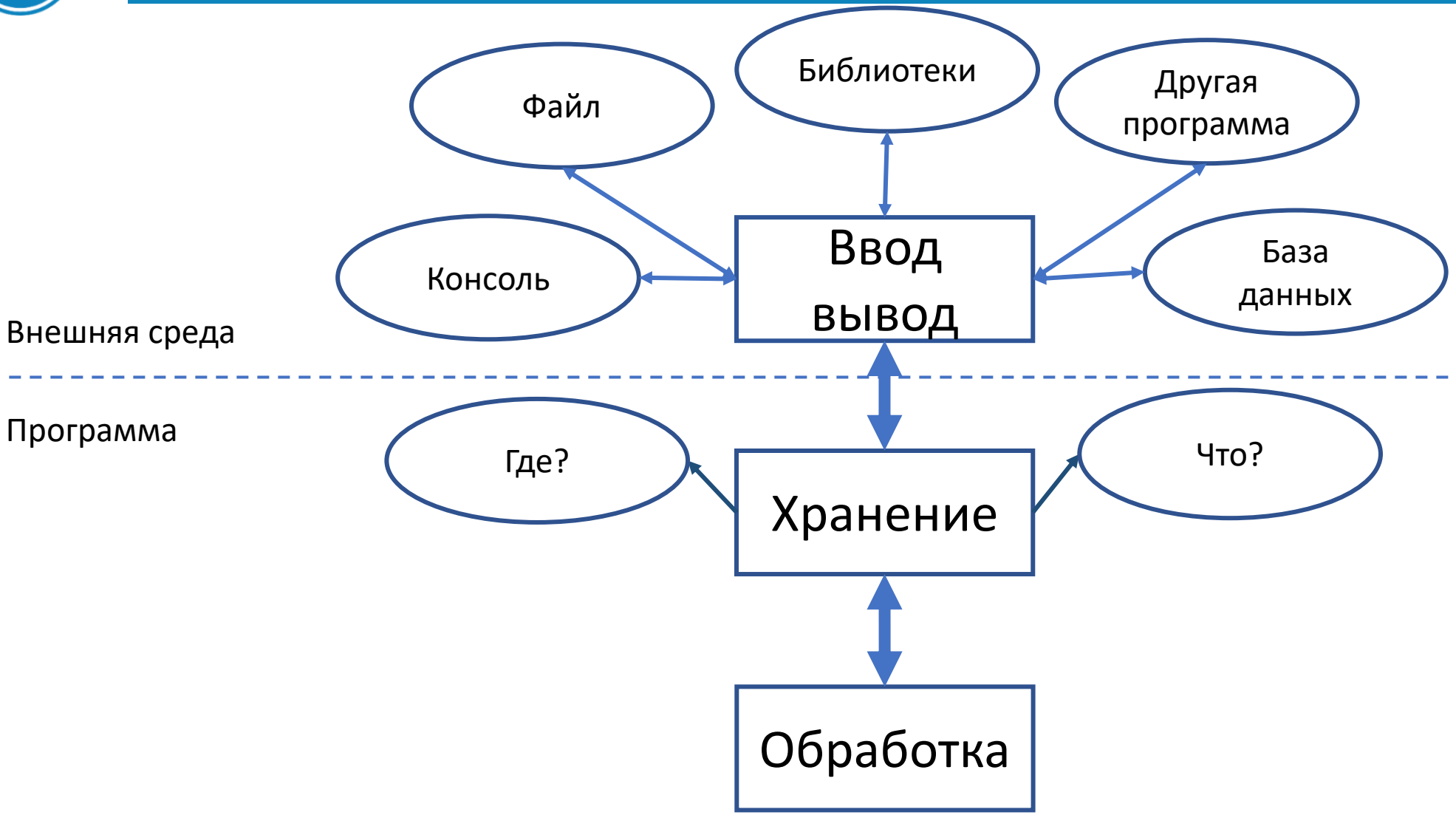
План занятий

№	Тема	Описание
1	Введение в курс	Языки программирования. Основы работы с Linux.
2	Основы языка С	Написание и компиляция простейших программ с использованием gcc. Правила написания кода.
3	Компиляция	Разбиение программы на отдельные файлы. Make файлы. Компиляция.
4	Ввод данных. Библиотеки	Работа со вводом/выводом. Статические и динамические библиотеки.
5	Язык ассемблера	Основы анализа программ на языке ассемблер.
6	Хранение данных. Память	Хранение процесса в памяти компьютера. Виртуальная память, сегментация. Секции программы.
7	Хранение данных.	Стек, куча. Типы данных. Преобразования типов. Gdb и отладка Хранение различных типов данных. Указатели. Передача аргументов в функцию по указателю.
8	Обработка данных	Безопасные функции. Битовые операции – сдвиги, логические операции. Битовые поля.
9	Программирование под встраиваемые ОС	Работа с микрокомпьютером Raspberry Pi

Что мы пройдем сегодня?

1)Создание библиотек
2)Работа с вводом/выводом

Дерево языка

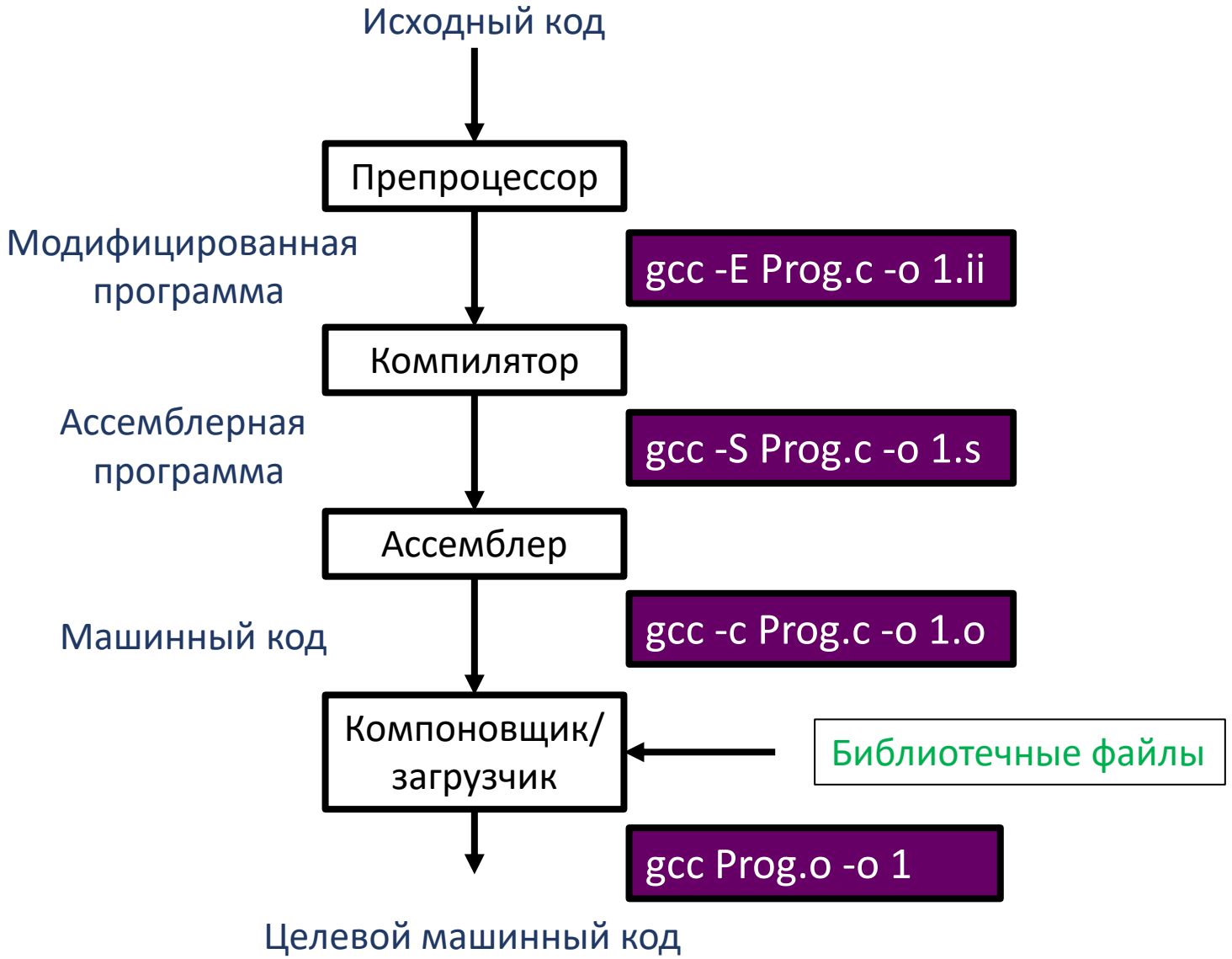


Исходный код программ

https://github.com/SergeyBalabaev

Elective-C-Programming-Language → *Lesson4*

Этапы компиляции



В качестве завершения…

```
int x, y;
int main()
{
        x = y + 1;
        y = 1;
}
```

Как будет скомпилирована программа?

В качестве завершения…

```
1   int x, y;
2   int main()
3   {
4       x = y + 1;
5       y = 1;
6   }
```

Порядок выполнения команд верен

```
1   x:
2       .zero   4
3   y:
4       .zero   4
5   main:
6       pushq   %rbp
7       movq    %rsp, %rbp
8       movl    y(%rip), %eax
9       addl    $1, %eax
10      movl    %eax, x(%rip)
11      movl    $1, y(%rip)
12      movl    $0, %eax
13      popq    %rbp
14      ret
```

https://gcc.godbolt.org/

В качестве завершения…

```
1    int x, y;
2    int main()
3    {
4        x = y + 1;
5        y = 1;
6    }
```

Порядок выполнения команд нарушен

```
1    main:
2        movl    y(%rip), %eax
3        movl    $1, y(%rip)
4        addl    $1, %eax
5        movl    %eax, x(%rip)
6        xorl    %eax, %eax
7        ret
8    y:
9        .zero   4
10   x:
11       .zero   4
```

-O2 -std=c++20

Разбиение программы на модули

lib.h

```
#pragma once
extern void summ(int, int, int*);
extern void mult(int, int, int*);
```

lib.c

```
void summ(int x, int y, int* sum)
{
	*sum = x + y;
}

void mult(int x, int y, int* mult)
{
	*mult = x * y;
}
```

main.c

```
#include <stdio.h>
#include "lib.h“

int main()
{
int res_sum, res_mult;
summ(10, 10, &res_sum);
mult(2, 2, &res_mult);
if(res_sum < res_mult)
	printf("Sum = %d\n", res_sum);
else
	printf("Mult = %d\n", res_mult);
return 0;
}
```

```
gcc main.c –c -Werror -Wall -g -o main.o
gcc lib.c -c  -Werror -Wall -g -o lib.o
gcc main.o lib.o -o main
./main
```

Make файлы

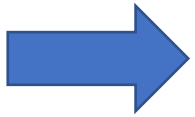
цель: зависимости
[tab] команда

makefile

```
all: clean
	gcc -c *.c
	gcc *.o -o main
clean:
	rm -f *.o
```

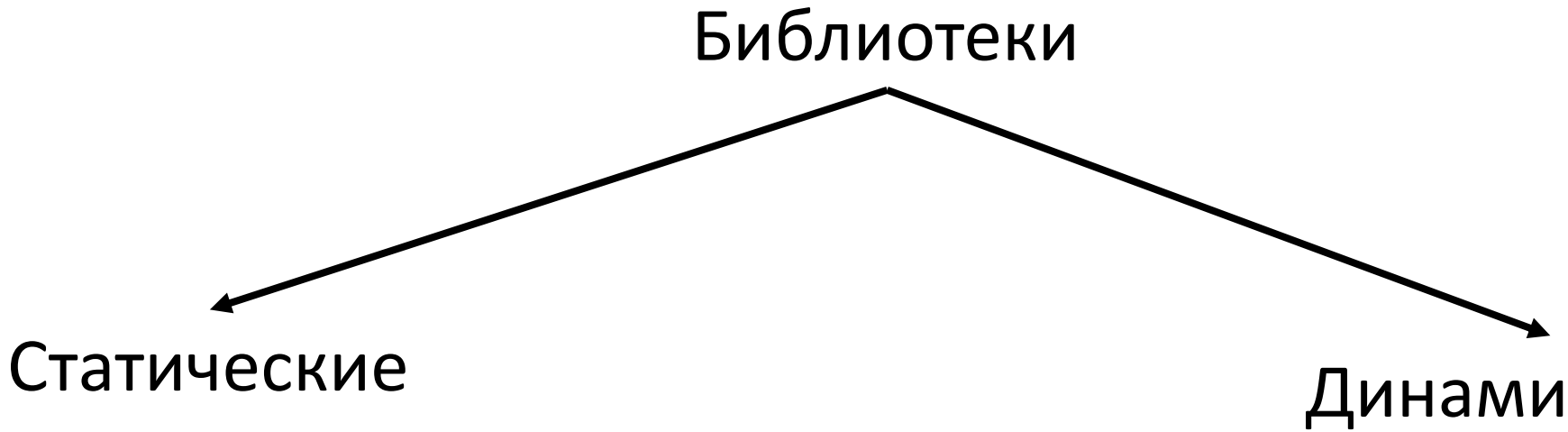
```
gcc main.c -c -o main.o
gcc lib.c -c -o lib.o
gcc main.o lib.o -o main
```

```
CFLAGS=-Wall -g –Werror
all: clean
	gcc -c $(CFLAGS) *.c
	gcc *.o $(CFLAGS) -o main
clean:
	rm -f *.o
```



```
make
```

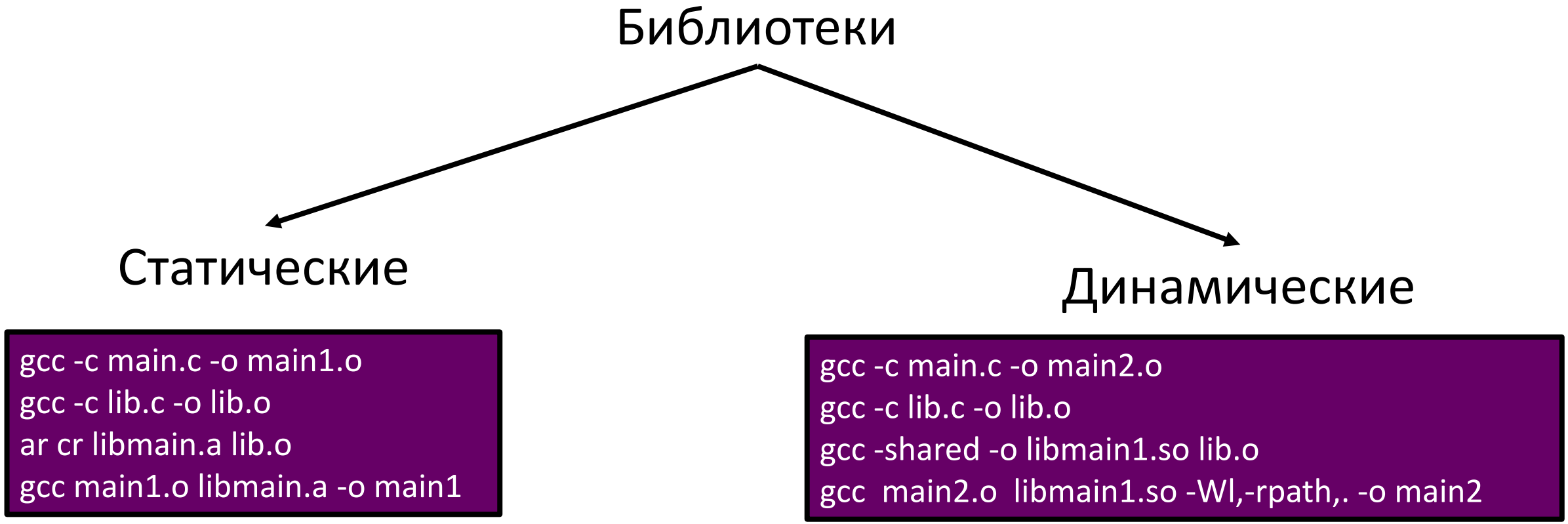
Библиотеки



Подключаются к программе во время **компоновки**.

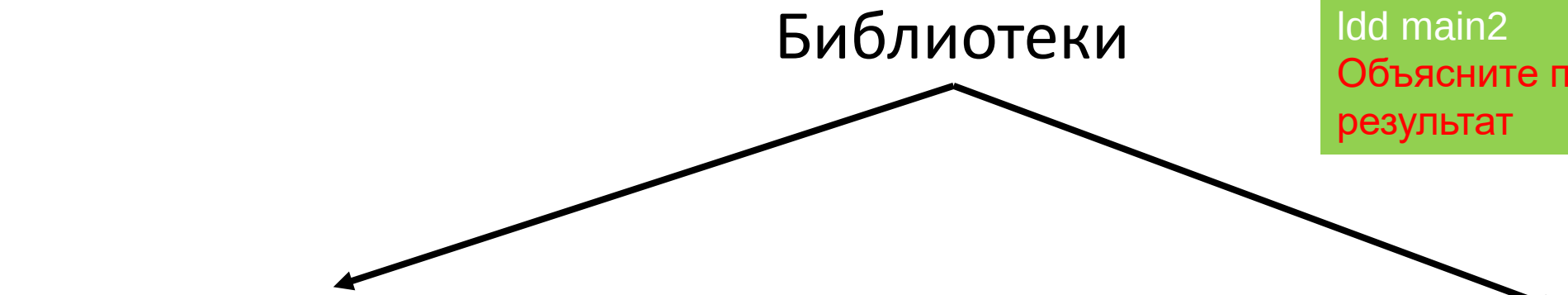
Подключаются к программе во время **выполнения программы**

Статические библиотеки



Статические библиотеки

Выполните команды
ldd main1
ldd main2
Объясните полученный результат




```
gcc -c main.c -o main1.o
gcc -c lib.c -o lib.o
ar cr libmain.a lib.o
gcc main1.o libmain.a -o main1
```

```
gcc -c main.c -o main2.o
gcc -c lib.c -o lib.o
gcc -shared -o libmain1.so lib.o
gcc  main2.o  libmain1.so -Wl,-rpath,. -o main2
```

Статические библиотеки

Выполните команды
objdump main1 -d
objdump main2 -d
Найдите отличия в файлах

Выполните команды
ldd main1
ldd main2
Объясните полученный результат

Библиотеки

Статические

```
gcc -c main.c -o main1.o
gcc -c lib.c -o lib.o
ar cr libmain.a lib.o
gcc main1.o libmain.a -o main1
```

Динамические

```
gcc -c main.c -o main2.o
gcc -c lib.c -o lib.o
gcc -shared -o libmain1.so lib.o
gcc  main2.o  libmain1.so -Wl,-rpath,. -o main2
```

Основные определения из курса ОС

Процесс – программа во время исполнения и все её элементы: адресное пространство, глобальные переменные , регистры, стек, счетчик команд, состояние, открытые файлы, дочерние процессы и т. д

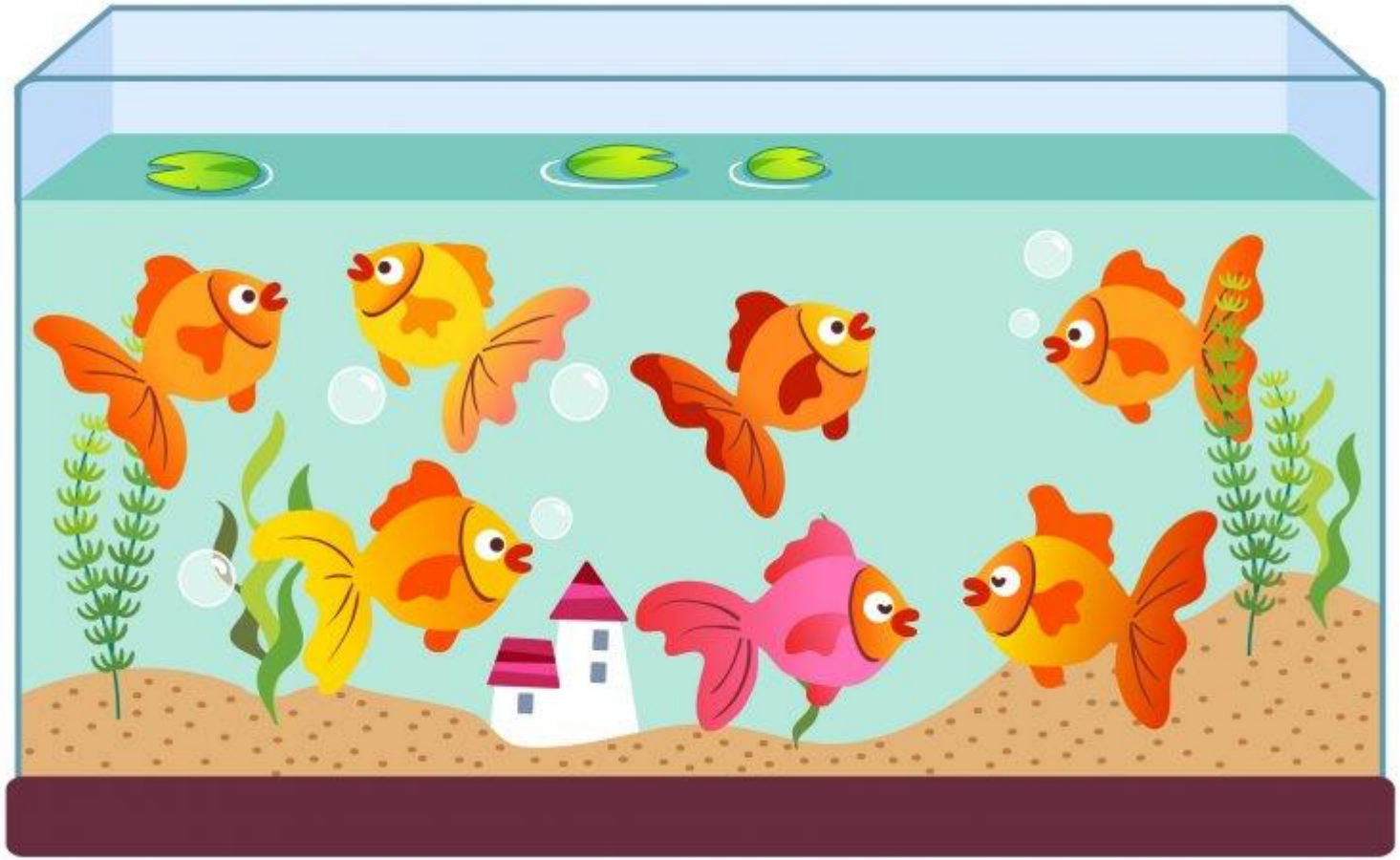
Поток - самостоятельная цепочка последовательно выполняемых операторов программы, соответствующих некоторой подзадаче

Прерывание – событие, при котором меняется последовательность команд, выполняемых процессором

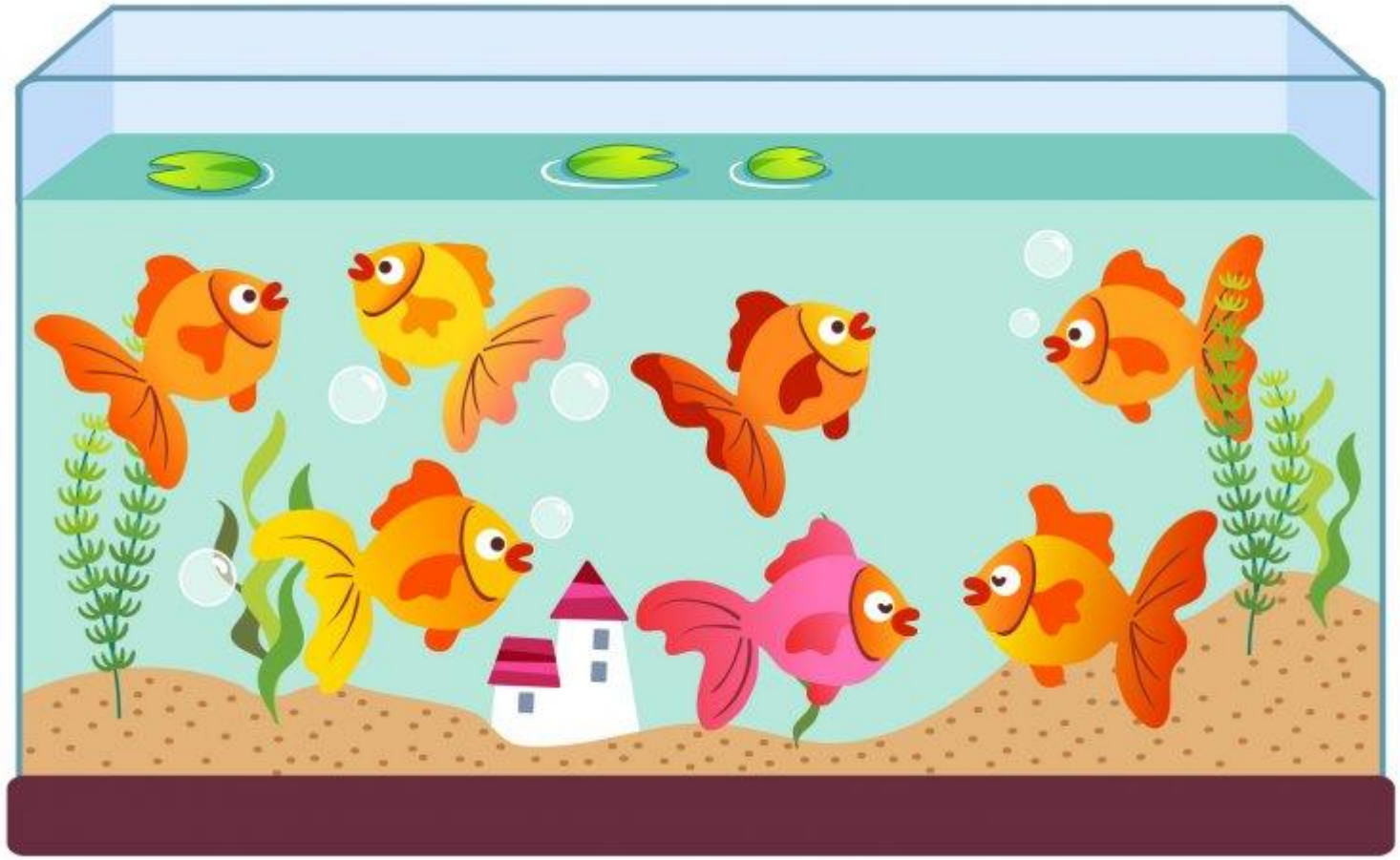
Системный вызов – это интерфейс для получения услуг операционной системы

Файл – поименованная совокупность данных

Основные определения из курса ОС



Основные определения из курса ОС



Аквариум - процесс

Рыбки - потоки

Корм - ресурсы

```c
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

Стандартный ввод и вывод

arch/x86/boot/printf.c

```
int printf(const char* fmt, ...)
{
   char printf_buf[1024];
   va_list args; int printed;
   va_start(args, fmt);
   printed = vsprintf(printf_buf, fmt, args);
   va_end(args);
   puts(printf_buf);
   return printed;
}
```

Стандартный ввод и вывод

arch/x86/boot/printf.c

```
int vsprintf(char* buf, const char* fmt, va_list args)
{
    /////
    case '%':
    *str++ = '%';
    continue;
    /* integer number formats - set up the flags and "break" */
    case 'o':
    case 'x':
    case 'X':
    case 'd':
    case 'i':
    case 'u':
    break;
    /////
    return str - buf;
}
```

Стандартный ввод и вывод

/arch/nios2/boot/compressed/console.c

```
static int puts(const char* s)
{
   while (*s)
      putchar(*s++);
   return 0;
}
```

Стандартный ввод и вывод

/arch/nios2/boot/compressed/console.c

```
static int putchar(int ch)
{
   uart_putc(ch);
   if (ch == '\n')
      uart_putc('\r');
   return ch;
}
```

Стандартный ввод и вывод

/arch/nios2/boot/compressed/console.c

```
static void uart_putc(int ch)
{
int i;

for (i = 0; (i < 0x10000); i++) {
if (readw(uartbase + ALTERA_UART_STATUS_REG) &
    ALTERA_UART_STATUS_TRDY_MSK)
    break;
}
        writeb(ch, uartbase + ALTERA_UART_TXDATA_REG);
}
```

Стандартный ввод и вывод

/arch/x86/boot/tty.c

```
void __section(".inittext") puts(const char* str)
{
   while (*str)
      putchar(*str++);
}
```

Стандартный ввод и вывод

/arch/x86/boot/tty.c

```
void __section(".inittext") putchar(int ch)
{
    if (ch == '\n')
        putchar('\r');/* \n -> \r\n */

    bios_putchar(ch);

    if (early_serial_base != 0)
        serial_putchar(ch);
}
```

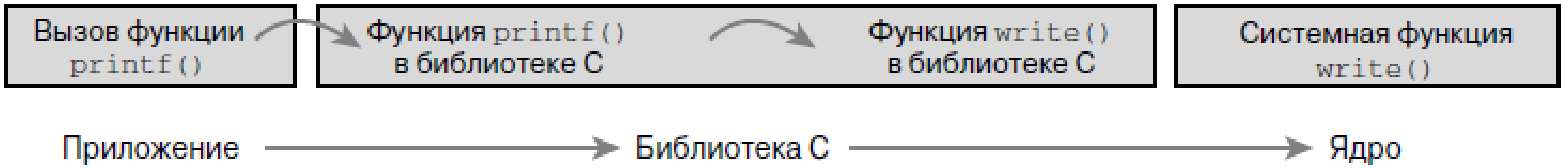
Стандартный ввод и вывод

/arch/x86/boot/tty.c

```
static void __section(".inittext") bios_putchar(int ch)
{
    struct biosregs ireg;

    initregs(&ireg);
    ireg.bx = 0x0007;
    ireg.cx = 0x0001;
    ireg.ah = 0x0e;
    ireg.al = ch;
    intcall(0x10, &ireg, NULL);
}
```

Стандартный ввод и вывод



ssize_t **write**(int fd, const void *buf, size_t nbytes);

write(fd1, buf, strlen(buf));

Hello world

```c
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

```c
#include <unistd.h>

int main(int argc, char* argv[])
{
    char str[] = "Hello, world!\n";
    write(1, str, sizeof(str) - 1);
    _exit(0);
}
```

Стандартный ввод и вывод

```
extern FILE *stdin; /* Standard input stream. */
extern FILE *stdout; /* Standard output stream. */
extern FILE *stderr; /* Standard error output stream. */
```

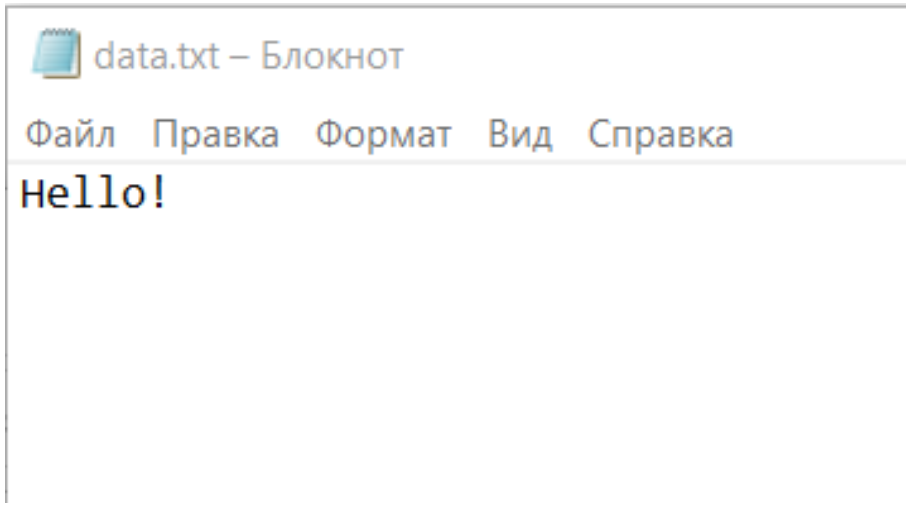
Основные функции работы с файлами

Описание	**Синтаксис**	**Комментарий**
Открытие файла	errno_t **fopen_s**(FILE** pFile, const char* filename, const char* mode);	Модификаторы: **"r"** Открывает для чтения **"w"** Открывает для перезаписи **"a"** Открывает для записи в конец файла
Запись в файл	int **fprintf**(FILE *stream, const char *format [, argument]...);	Если stream=***stdout***, то вывод будет производиться на экран
Чтение из файла	char ***fgets**(char *str, int numChars, FILE *stream);	Считывание возможно проводить в цикле до момента, когда функция вернет NULL *while ((fgets(c, 3, fp)) != NULL)* Считывает numChars – 1 символ
Закрытие файла	int **fclose**(FILE *stream);	

Работа с файлами

```c
#include <stdio.h>

int main(void)
{
    FILE* fp;
    fopen_s(&fp, "data.txt", "w");
    fprintf(fp, "%s", "Hello!");
    fclose(fp);
    return 0;
}
```



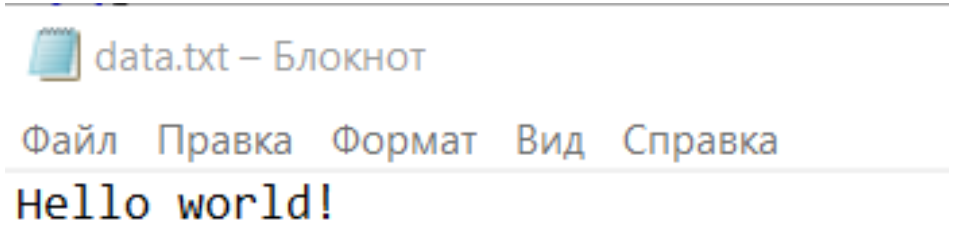
Работа с файлами

```c
#include <stdio.h>

int main(void)
{
    FILE* fp;
    char sym[10];
    fopen_s(&fp, "data.txt", "w");
    fprintf(fp, "%s", "Hello world!");
    fclose(fp);

    fopen_s(&fp, "data.txt", "r");
    while ((fgets(sym, 10, fp)) != NULL)
    {
        printf("%s\n", sym);
    }
    fclose(fp);

return 0;
}
```




```
Hello wor
ld!
```

Работа с файлами

```
if ((fopen_s(&fp, filename, "w")) != NULL)
    {
        perror("Error");
        return 1;
    }
```

При работе с файлами рекомендуется проводить проверку на корректное открытие

Полный код см. в заметках к слайду

Буфер stdout

```c
#include <stdio.h>

int main()
{
    fprintf(stdout, "Hello 1 ");
    fprintf(stderr, "This is error. ");
    fprintf(stdout, " Hello 2 \n");
    return 0;
}
```

Что будет выведено в результате работы программы?

Буфер stdout

```c
#include <stdio.h>

int main()
{
    fprintf(stdout, "Hello 1 ");
    fprintf(stderr, "This is error. ");
    fprintf(stdout, " Hello 2 \n");
    return 0;
}
```

Что будет выведено в результате работы программы?

```
sab@LAPTOP-B03PIUAN:.../Lesson4$ ./ffl
This is error. Hello 1  Hello 2
```

Перенаправление потоков ввода/вывода

write.c

```
#include <stdio.h>
int main(){
        printf("world!");
        return 0;
}
```

read.c

```
#include <stdio.h>

int main(){
        char address[100];
        printf("Hello ");
        scanf("%s", address);
        printf("%s\n", address);
        return 0;
}
```

```
./read > log.txt
./write > text.txt
./read  < text.txt
./read > log.txt < text.txt
./write | ./read
```

Просто интересный пример

```
int x = 0;
int y = 0;

int r1 = 0;
int r2 = 0;

std::thread t1([&]() {
    x = 1;
    r1 = y;
});

std::thread t2([&]() {
    y = 1;
    r2 = x;
});
```

Чему могут быть равны r1 и r2 в результате работы программы?

Спасибо за внимание!