

Разработка кроссплатформенных
программных систем.
**Лекция №2. Разработка переносимых
приложений на языке C**

П.Н. Советов

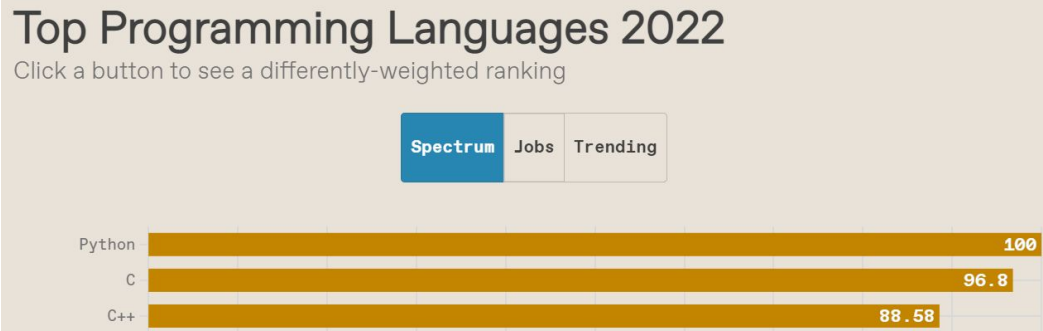
РТУ МИРЭА, 2022

Актуальность языка C

C (1972) до сих пор популярен в области системного программирования и встраиваемых систем.

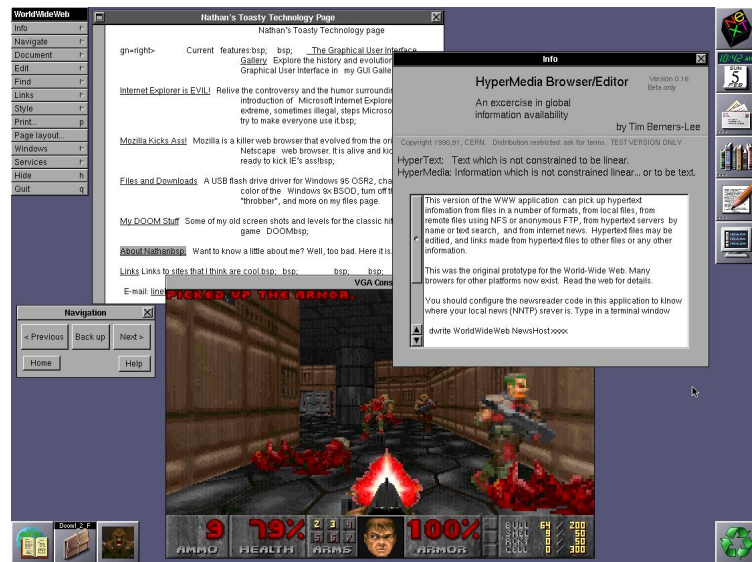
На C написаны:

- UNIX-подобные ОС, в частности Linux,
- Git,
- интерпретаторы Python и Lua,
- Веб-сервер Nginx,
- СУБД SQLite и Redis.
- Классические игры Doom и Quake.



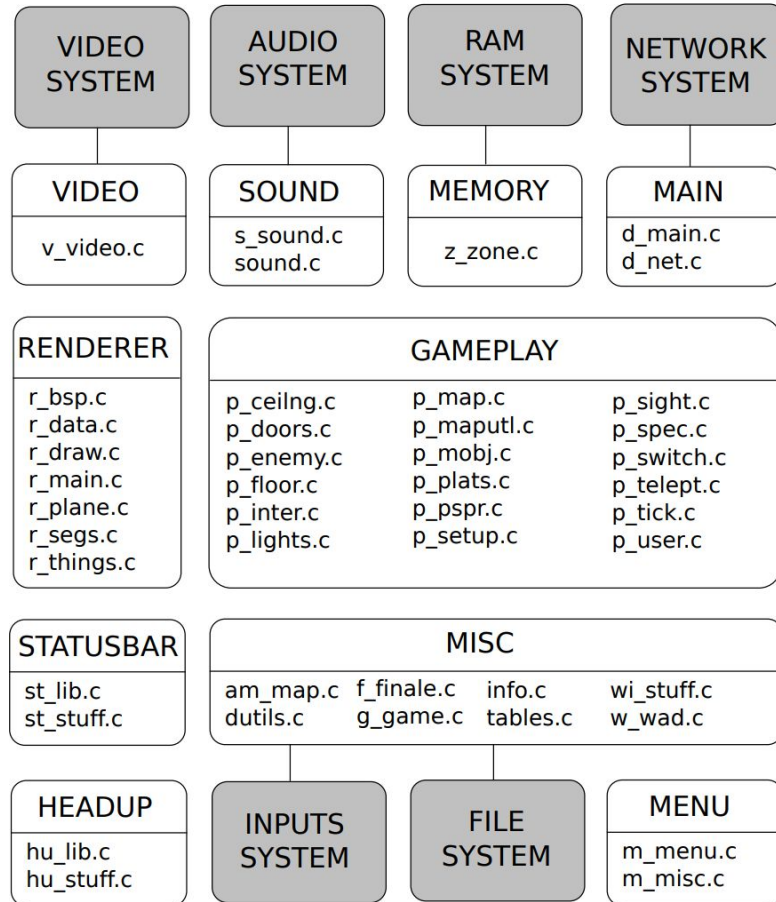
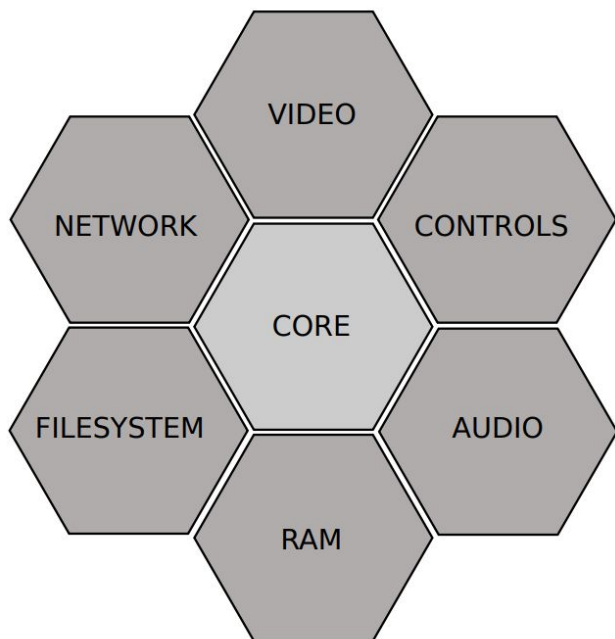
- Рабочая станция NeXT Computer под управлением варианта ОС UNIX NeXTSTEP с компилятором **gcc**.
- PC с ОС MS-DOS и компилятором **Watcom**.

В дальнейшем Doom был портирован на множество самых различных платформ.



Переносимая архитектура Doom

В Doom выделено переносимое ядро. Платформо-зависимые компоненты (серый цвет) подключались к проекту в виде *.c файлов.



Исходный код современной версии Doom: <https://github.com/ozkl/doomgeneric>

Для портирования doomgeneric на новую платформу в коде достаточно переопределить всего 5 функций:

DG_Init	Инициализация платформо-зависимых функций
DG_DrawFrame	Вывод видеобuffers на экран.
DG_SleepMs	Задержка в миллисекундах.
DG_GetTicksMs	Время в миллисекундах.
DG_GetKey	Опрос клавиатуры.

- Хотя C и C++ схожи, C не является подмножеством C++.
- C представляет собой значительно более простой язык, чем C++.
- В C отсутствуют развитые средства абстрагирования из C++: классы и шаблоны.
- В этой связи использование C для написания больших прикладных программ, скорее всего, не является оправданным.
- Компиляторы C существуют для большего числа платформ, чем в случае C++.
- Низкоуровневые интерфейсы для ОС и библиотек часто представлены только в варианте для C.

Основные кроссплатформенные компиляторы C:

- gcc,
- clang.

Для Windows предлагается использовать сборку gcc под названием MinGW-W64. Архив с префиксом x86_64_..._release-win32-seh-...:

<https://github.com/nixman/mingw-builds-binaries/releases>

Онлайн-компиляторы C: <https://godbolt.org/>

Стандарт	Описание
K&R C	Оригинальный вариант 1978 года.
C89, ANSI C, C90	Первый стандарт. Считается устаревшим, но поддерживается большинством компиляторов.
C99	Наиболее популярный сегодня стандарт, который будем использовать и мы.
C11	
C17, C19	Исправления для C11.
C2x	Грядущий стандарт.


```
gcc -std=c99 -Wall -Wpedantic program.c
```

- Ключ `-std=c99` включает поддержку стандарта C99.
- Ключ `-Wall` включает оповещение обо всех “подозрительных” моментах в коде.
- Ключ `-Wpedantic` включает более тщательную проверку на соответствие стандарту.

В стандарте C порядок вычислений в выражениях и аргументах не определен, поэтому такой код является непереносимым:

```
buf1[i] = buf2[++i];
```

В выражениях от операторов автоинкремента/декремента лучше вообще отказаться.

- В C размер таких типов как `int` может иметь различные значения на различных платформах.
- Лучше задавать разрядность типов явно, например так: `int32_t`, `uint32_t`. Для этого надо подключить заголовочный файл `stdint.h`.
- Размеры и индексы можно хранить с помощью типа `size_t` (определен в `stddef.h`).
- Булев тип `bool` имеет значения `true` и `false`. Определен в `stdbool.h`.

Избегайте неинициализированных данных.

Используйте следующую конструкцию для инициализации массивов и структур:

```
uint32_t data[128] = {0};
```

Вместо malloc используйте calloc, поскольку calloc обнуляет выделенную память.

Интерфейс модуля `my` описан в заголовочном файле `my.h`. Функции имеют дело со структурой `my_state`, представляющей собой объект – состояние модуля:

```
#ifndef MY_H
#define MY_H

int func1(struct my_state *my, ...);
int func2(struct my_state *my, ...);
...

#endif
```

Классические учебники по C89:

- Керниган Б. У., Ритчи Д. М. Язык программирования C/The C Programming Language. – Вильямс, 2015.
- Керниган Б. В. Пайк Роб. Практика программирования //СПб.: Невский диалект. – 2001.

Учебники по современному C:

- Gustedt J. Modern C. – Simon and Schuster, 2019. URL:
<https://hal.inria.fr/hal-02383654/file/ModernC.pdf>
- Beej's Guide to C Programming. URL:
https://beej.us/guide/bgc/pdf/bgc_usl_c_1.pdf

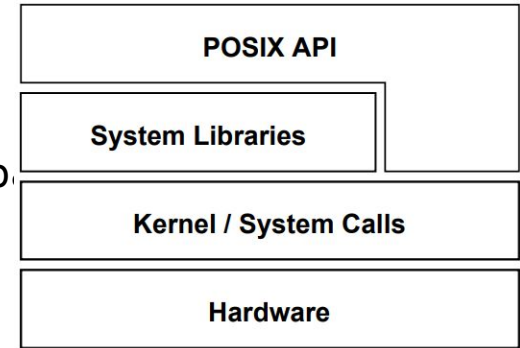
В 80-е годы количество несовместимых UNIX-систем стало слишком большим. Возник проект набора стандартов на API для программ, выполняющихся в UNIX-совместимых ОС.

POSIX (portable operating system interface) – переносимый интерфейс операционной системы. Включает в себя стандартную библиотеку C.

Текущая версия POSIX.1-2017 содержит описание C API и набор стандартных команд и утилит ОС. В POSIX не определена работа с графикой и базами данных.

Linux и macOS являются POSIX-совместимыми ОС.

В Windows POSIX поддерживается на уровне WSL (Windows subsystem for Linux). Подмножество POSIX поддерживается в MinGW-W64.



Заголовочные файлы POSIX

16

Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
< aio.h>	•	•	•	•	asynchronous I/O
< cpio.h>	•	•	•	•	cpio archive values
< dirent.h>	•	•	•	•	directory entries (Section 4.22)
< dlfcn.h>	•	•	•	•	dynamic linking
< fcntl.h>	•	•	•	•	file control (Section 3.14)
< fnmatch.h>	•	•	•	•	filename-matching types
< glob.h>	•	•	•	•	pathname pattern-matching and generation
< grp.h>	•	•	•	•	group file (Section 6.4)
< iconv.h>	•	•	•	•	codeset conversion utility
< langinfo.h>	•	•	•	•	language information constants
< monetary.h>	•	•	•	•	monetary types and functions
< netdb.h>	•	•	•	•	network database operations
< nl_types.h>	•	•	•	•	message catalogs
< poll.h>	•	•	•	•	poll function (Section 14.4.2)
< pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
< pwd.h>	•	•	•	•	password file (Section 6.2)
< regex.h>	•	•	•	•	regular expressions
< sched.h>	•	•	•	•	execution scheduling
< semaphore.h>	•	•	•	•	semaphores
< strings.h>	•	•	•	•	string operations
< tar.h>	•	•	•	•	tar archive values
< termios.h>	•	•	•	•	terminal I/O (Chapter 18)
< unistd.h>	•	•	•	•	symbolic constants
< wordexp.h>	•	•	•	•	word-expansion definitions
< arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
< net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
< sys/mman.h>	•	•	•	•	memory management declarations
< sys/select.h>	•	•	•	•	select function (Section 14.4.1)
< sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
< sys/stat.h>	•	•	•	•	file status (Chapter 4)
< sys/statvfs.h>	•	•	•	•	file system information
< sys/times.h>	•	•	•	•	process times (Section 8.17)
< sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
< sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
< sys/utsname.h>	•	•	•	•	system name (Section 6.9)
< sys/wait.h>	•	•	•	•	process control (Section 8.6)

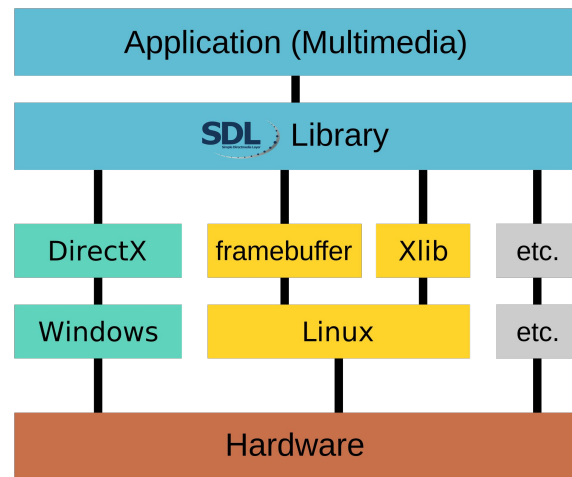
- Стивенс У. Р., Раго С. А. UNIX. Профессиональное программирование.-3-е изд //СПб.: Питер. – 2018.
- Курс В. Галатенко “Программирование в стандарте POSIX” на русском языке: <https://intuit.ru/studies/courses/47/47/info>
- Онлайн-версия стандартов POSIX:
<https://pubs.opengroup.org/onlinepubs/9699919799.2018edition/>
- MAN-страницы с указанием POSIX-совместимости API:
<https://man7.org/linux/man-pages/>

SDL2 (simple DirectMedia layer) – кроссплатформенная библиотека на C для написания переносимых игр и других мультимедийных приложений.

Поддерживаются следующие периферийные устройства:

- звуковая карта,
- клавиатура,
- мышь,
- джойстик,
- видеокарта с OpenGL и Direct3D.

Поддерживаемые платформы:
Windows, macOS, Linux, iOS, Android.



Скачать архив SDL2-devel-...-mingw.zip:

<https://github.com/libsdl-org/SDL/releases/tag/release-2.24.0>

К опциям компилятора добавить:

```
-ISDL2/x86_64-w64-mingw32/include/SDL2  
-LSDL2/x86_64-w64-mingw32/lib -w -lmingw32 -lSDL2main -lSDL2
```

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

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

    SDL_Window *window;           // Declare a pointer

    SDL_Init(SDL_INIT_VIDEO);     // Initialize SDL2

    // Create an application window with the following settings:
    window = SDL_CreateWindow(
        "An SDL2 window",        // window title
        SDL_WINDOWPOS_UNDEFINED, // initial x position
        SDL_WINDOWPOS_UNDEFINED, // initial y position
        640,                     // width, in pixels
        480,                     // height, in pixels
        SDL_WINDOW_OPENGL        // flags - see below
    );

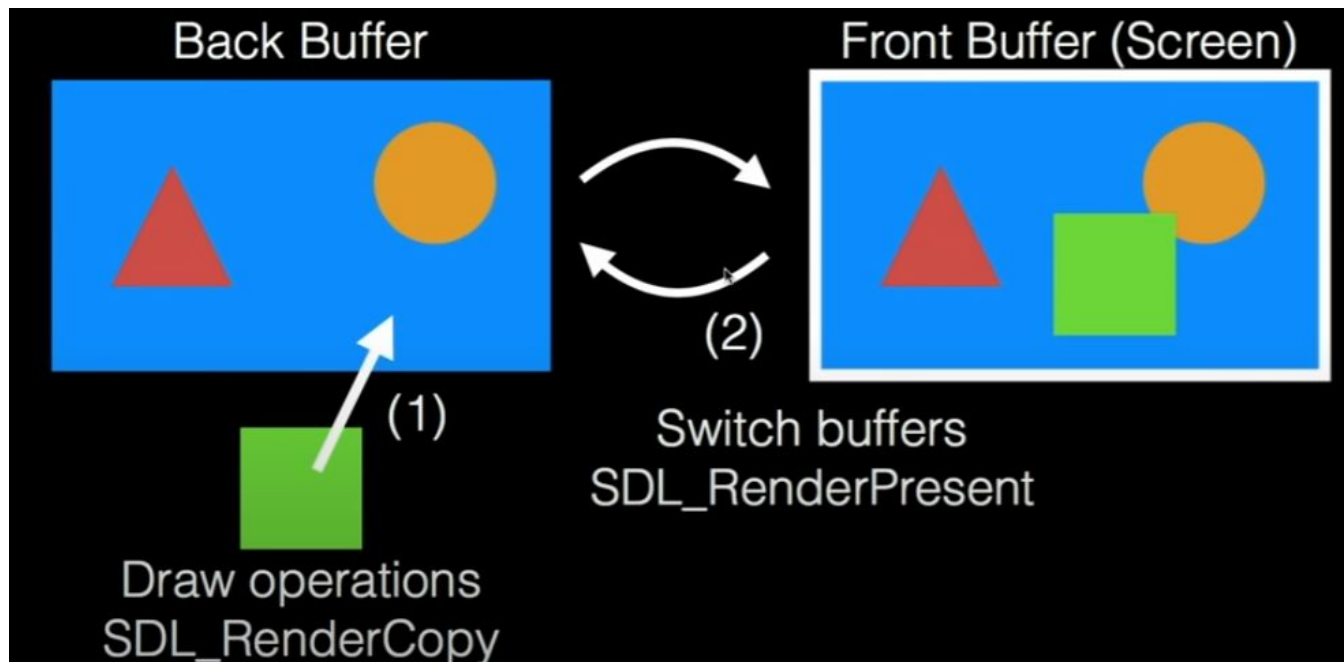
    // Check that the window was successfully created
    if (window == NULL) {
        // In the case that the window could not be made...
        printf("Could not create window: %s\n", SDL_GetError());
        return 1;
    }

    // The window is open: could enter program loop here (see SDL_PollEvent())

    SDL_Delay(3000); // Pause execution for 3000 milliseconds, for example

    // Close and destroy the window
    SDL_DestroyWindow(window);

    // Clean up
    SDL_Quit();
    return 0;
}
```



- Официальная документация с примерами использования (англ.): <https://wiki.libsdl.org/APIByCategory>
- Набор уроков по SDL2 (англ.): <https://lazyfoo.net/tutorials/SDL/index.php>