Язык Си: стандарты, основные концепции. Исполнение программы.

Стандарты языка Си

Начало 70-х:	появление языка Си
1978:	Kernighan, Ritchie (K&R)
1989:	ANSI C (C89)
1999:	C99
2011:	C11

Основные требования к языку Си

(мои «измышлизмы»)

- Язык должен быть эффективным как ассемблер
- На нем должно быть удобно программировать по сравнению с ассемблером
- Программы должны быть переносимы на уровне исходных текстов

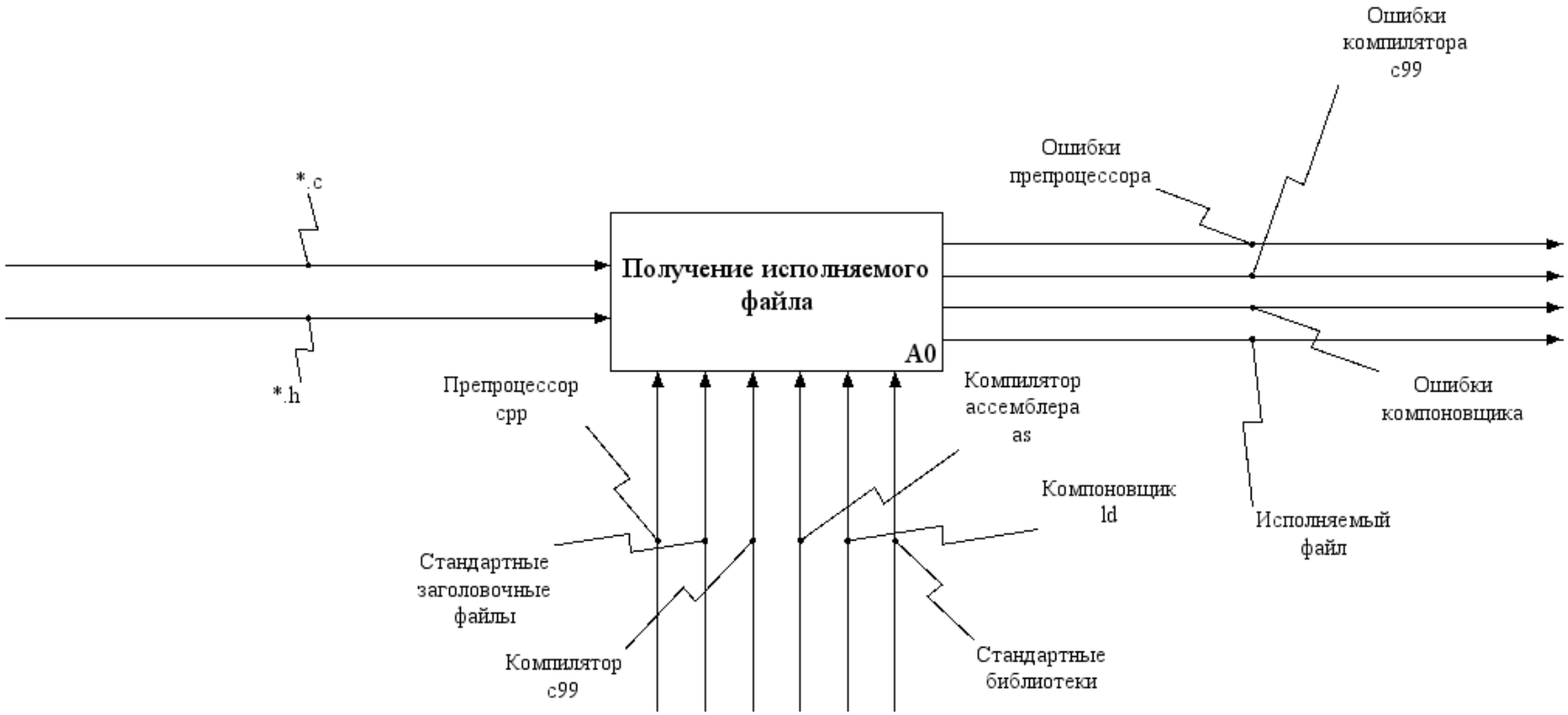
Основные концепции языка Си

- Си - язык сравнительно "низкого" уровня
- Си - "маленький" язык с однопроходным компилятором
- Си предполагает, что программист знает, что делает

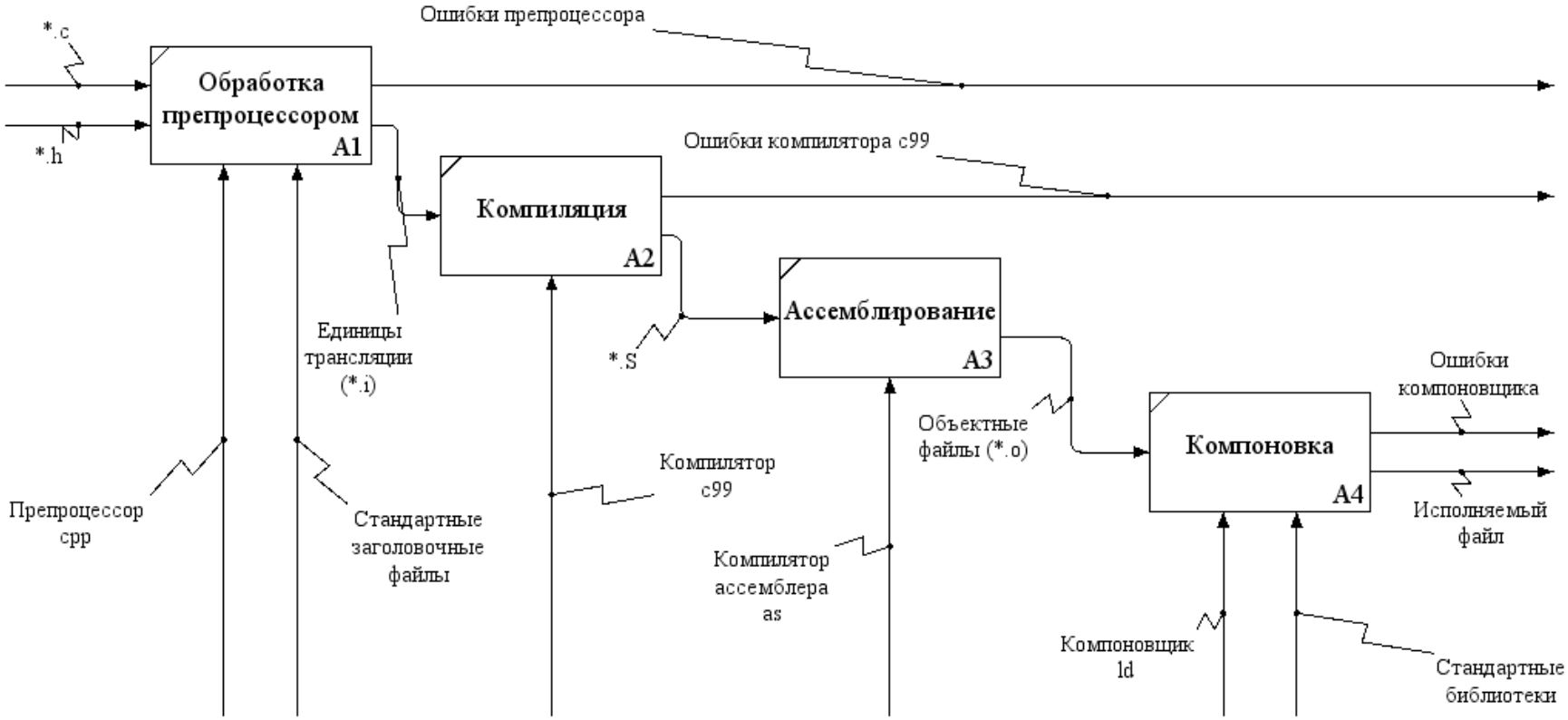
Получение исполняемого файла

```
 1. /*
 2. Первая программа на Си
 3. */
 4.
 5. #include <stdio.h>
 6.
 7. int main(void)
 8. {
 9.     puts(“Hello, world!”);
10.
11.     return 0;
12. }
```

Получение исполняемого файла



Получение исполняемого файла



A1: обработка препроцессором

Препроцессор выполняет:

- вырезание комментариев;
- текстовые замены (директива define);
- включение файлов (директива include).

Файл, получаемый в результате работы препроцессора, называется *единицей трансляции*.

```
cpp -o hello.i hello.c (или cpp hello.c > hello.i)
```

```
hello.c – 181 байт, hello.i – 19271 байт
```

Результат работы препроцессора

```
...

int __attribute__((__cdecl__)) __attribute__ ((__nothrow__))
                                                puts (const char*);

...

int main(void)
{
    puts("Hello, world!");

    return 0;
}
```

A2: трансляция на язык ассемблера

Компилятор выполняет трансляцию программы, написанной на Си, на язык ассемблера.

Язык ассемблера - система обозначений, используемая для представления в удобочитаемой форме программ, записанных в машинном коде. [wikipedia]

```
c99 -S -masm=intel hello.i
```

```
hello.c - 181 байт, hello.s - 5151 байт
```

Результат работы компилятора

```
...
   .section .rdata,"dr"
LC0:
   .ascii "Hello, world!\0"
   .text
...
   mov  DWORD PTR [esp], OFFSET FLAT:LC0
   call _puts
   mov  eax, 0
```

A3: ассемблирование в объектный файл

Ассемблер выполняет перевод программы на языке ассемблера в исполнимый машинный код.

В результате работы ассемблера получается *объектный файл*: блоки машинного кода и данных, с неопределенными адресами ссылок на данные и процедуры в других объектных модулях, а также список своих процедур и данных.

```
as –o hello.o hello.s
```

hello.c – 181 байт, hello.o – 1858 байт

Результат работы ассемблера

```
00000185 <_main>:
 185:     55                      push   %ebp
 186:     89 e5                   mov    %esp,%ebp
 188:     83 e4 f0                and    $0xfffffff0,%esp
 18b:     83 ec 10                sub    $0x10,%esp
 18e:     e8 00 00 00 00          call   193 <_main+0xe>
 193:     c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 19a:     e8 00 00 00 00          call   19f <_main+0x1a>
 19f:     b8 00 00 00 00          mov    $0x0,%eax
 1a4:     c9                      leave
 1a5:     c3                      ret
 1a6:     90                      nop
 1a7:     90                      nop
```

```
00000000 b .bss
00000000 d .data
00000000 r .eh_frame
00000000 r .rdata
00000000 r .rdata$zzz
00000000 t .text
         U ___main
         U __filbuf
         U __flsbuf
         U __imp___iob
         U _fgetpos
         U _fopen
0000013b T _fopen64
00000155 T _ftello64
00000000 T _getc
0000008e T _getchar
00000185 T _main
00000041 T _putc
000000dc T _putchar
         U _puts
```

A4: компоновка

Компоновщик принимает на вход один или несколько объектных файлов и собирает по ним исполнимый файл.

Компоновщик может извлекать объектные файлы из специальных коллекций, называемых библиотеками.

```
ld –o hello.exe hello.o ...библиотеки
```

```
hello.c – 181 байт, hello.exe – 91450 байт
```

Опции компилятора и компоновщика

```
gcc [опции] [выходной_файл] файл_1 [файл_2]
```

- -std=name (в нашем случае name = c99)
- -pedantic
- -Wall
- -Werror
- -c (--compile)
- -o <имя>
- -g[level] (--debug)

Примеры запуска компилятора

```
// 1. Препроцессор
gcc -E main.c > main.i
// 2. Трансляция на язык ассемблера
gcc -S main.i
// 3. Ассемблирование
gcc -c main.s
// 4. Компоновка
gcc -o main.exe main.o

// Вместо 1-3 можно написать
gcc -c main.c
```

Примеры запуска компилятора

```
// Вместо 1-4 можно написать
gcc -o main.exe main.c
```

Следующие опции обязательны для использования

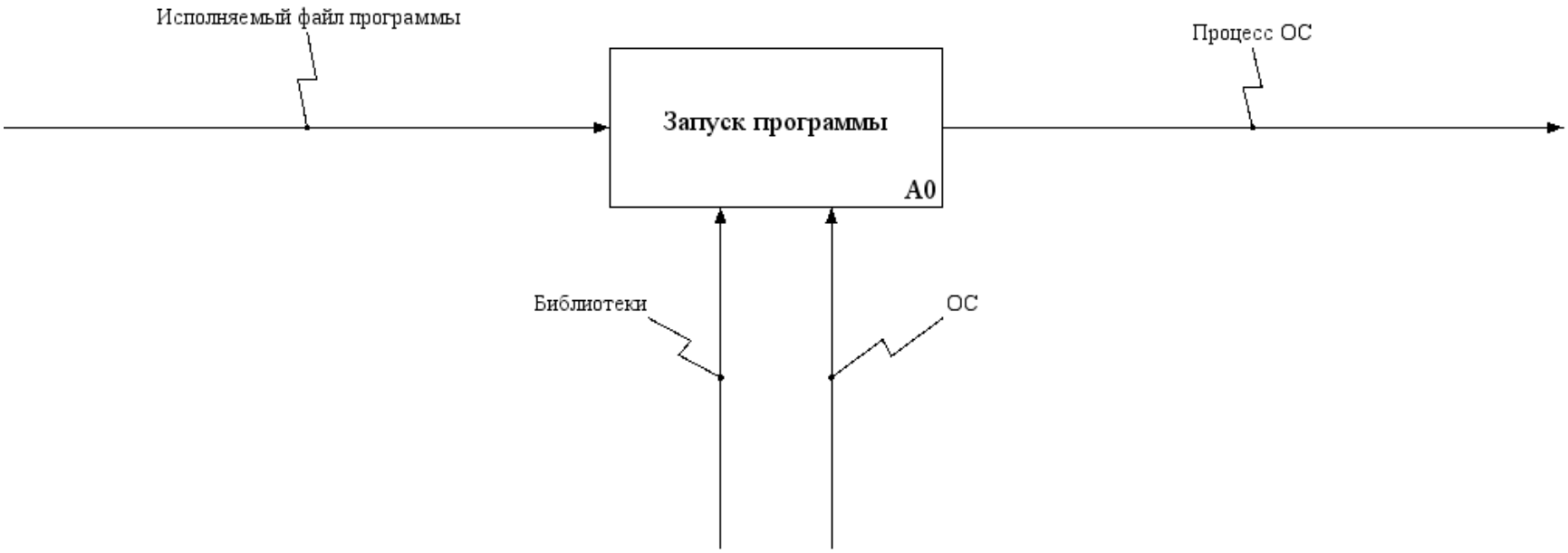
- **–std=c99**
- **–Wall**
- **–Werror**
- **-pedantic** (в некоторых случаях)

```
gcc -std=c99 -Wall -Werror -o main.exe main.c
```

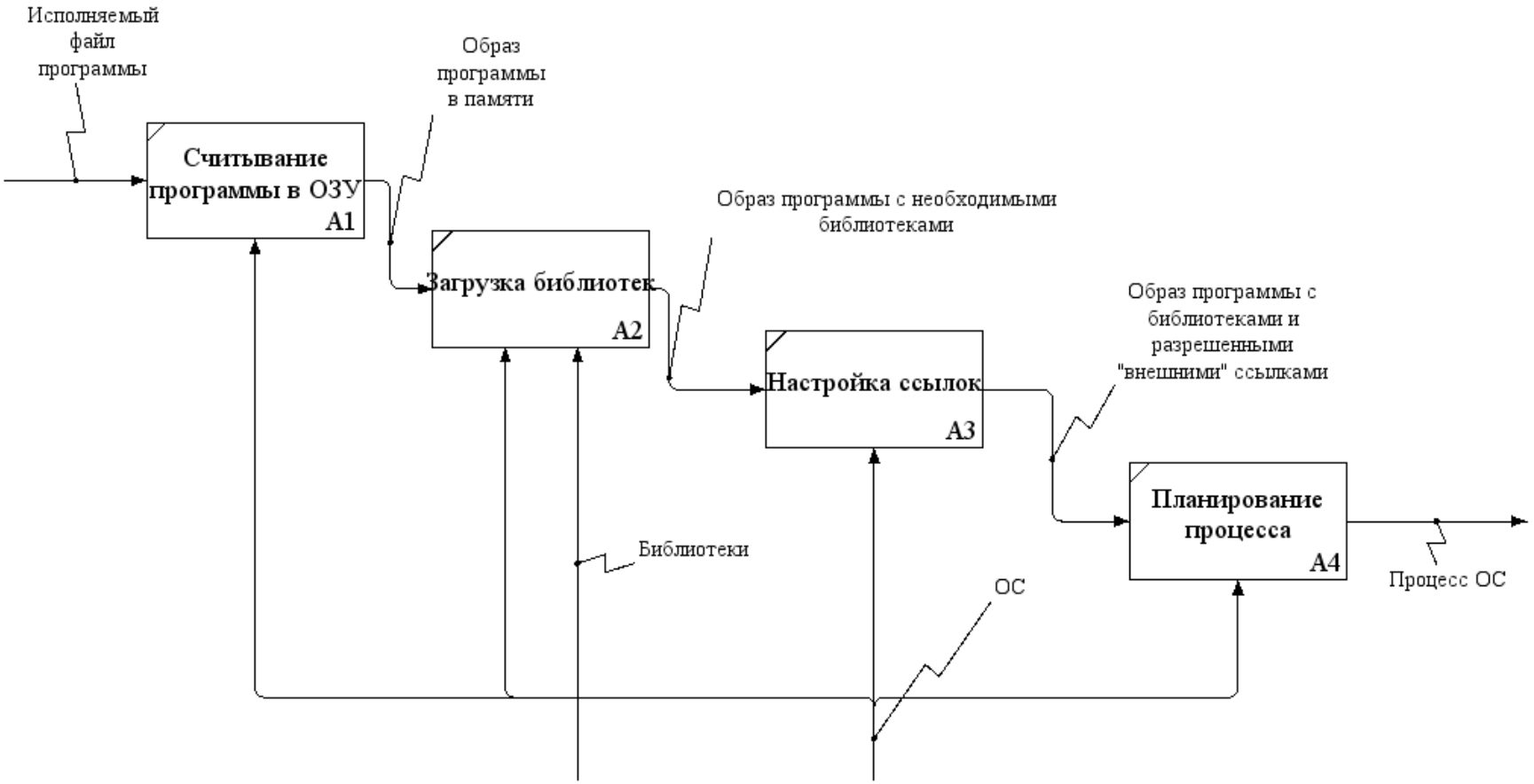
Представление о формате исполняемого файла

Заголовок 1
…
Заголовок N
Секция text
Секция bss
Секция data
Секция rodata
Таблица импорта
…

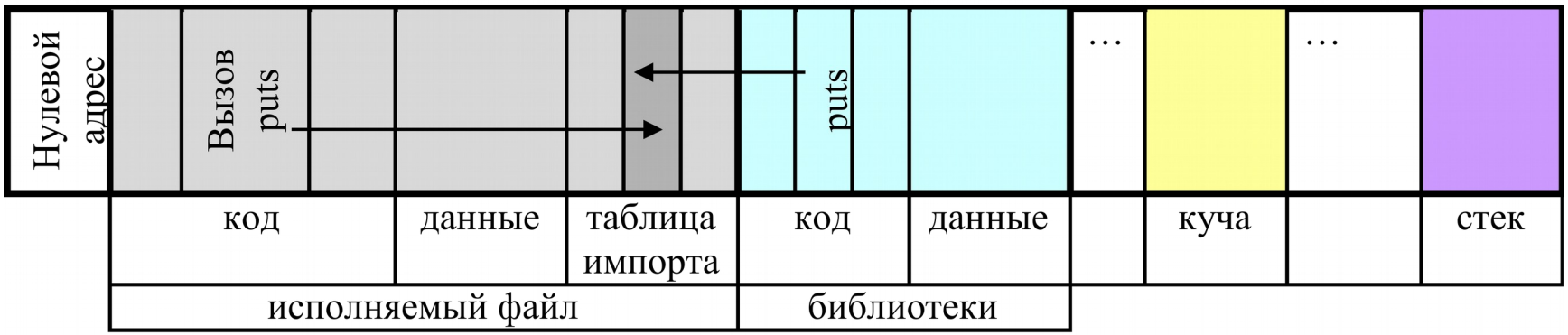
Запуск программы



Запуск программы



Абстрактная память и процесс.



Функция main

```
int main(void);
int main(int, char** argv);
```

Значение, возвращаемое main

```
@echo off
hello.exe
if errorlevel 1 goto err
if errorlevel 0 goto ok
goto fin
:err
echo ERROR!
goto fin
:ok
echo OK
:fin
```

Литература

1. Черновик стандарта C99
2. Dennis M. Ritchie, The Development of the C Language
3. Артур Гриффитс, GCC: Настольная книга пользователей, программистов и системных администраторов.
4. John R. Levine, Linkers & Loaders
5. David Drysdale, Beginner's Guide to Linkers (есть перевод на хабре)