



ОНЛАЙН-ОБРАЗОВАНИЕ


Онлайн-образование

Проверить, идет ли запись!





Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы

Препроцессор С



Легкоступ Виктор Валерьевич

The image features a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City, with numerous skyscrapers and buildings. A semi-transparent blue band with a white geometric network pattern of dots and lines stretches horizontally across the middle of the image. The word "Начало" is centered within this band in a white, sans-serif font.

Начало

Что такое макрос

Макрос - это именованный фрагмент текста

```
#include "stdafx.h"  
#include <stdio.h>  
#define COUNTS 100  
#define VIEW ALT  
#define SQUARE(x) ((x) * (x))
```

Макросы делятся на:

- object-like macros
- function-like macros

Зачем это надо?

Основное назначение макросов:

1. Подключение заголовков `#include <stdio.h>`
2. Подключение/создание платформо-зависимого кода `#ifdef __linux__`
`#define GO 1`
`#endif`
3. Создание аналога констант `#define COUNTS 100`
4. Создание простых функций `#define SQUARE(x) ((x)*(x))`
5. Включение отладочной информации `printf("Error in %s", __FILE__);`
6. Многое другое...

Основные сведения

Многие функции и квалификаторы являются макросами:

`NULL, isalpha, isfinite, assert, sin, cos, pow,...`

Посмотреть результат раскрытия макросов можно, воспользовавшись ключом E:

`gcc -E`

Будут созданы файлы `*.i`, `*.ii`

object-like macroses

Это аналоги констант

```
#define COUNTS 100  
#define VIEWPORT_EXIST  
#define N_MAX 127
```

```
int buf[N_MAX];  
#undef N_MAX // удаление определения
```


function-like macros

1. Обрамлять все тело и аргументы скобками:

```
#define DOUBLE(x) 2*x
DOUBLE(1+1)*8; // 2*1+1*8 = 10
#define DOUBLE(x) (2*(x))
DOUBLE(1+1)*8; // (2*(1+1))*8 = 32
```

2. Избегать двойного использования

```
#define MAX(a, b) ((a) > (b) ? (a) : (b))
int x = 1, y = 2;
int m = MAX(x, y++); // int m = ((x) > (y++) ? (x) : (y++))
```

3. Использовать для блоков фигурные скобки, а лучше do { ... } while(0)

```
#define DOUBLEINCREMENT(a, b) (a)++; (b)++;

int x = 1, y = 0;
if(x>y)
    DOUBLEINCREMENT(x, y);

// раскрывается в:
int x = 1, y = 0;
if(x>y)
    (x)++;
    (y)++;
```


Директивы условной компиляции

Проверка условий:

```
#if N == 1
    printf("N = 1");
#elif N == 2
    printf("N = 2");
#elif N > 2
    printf("N > 2");
#else
    printf("N is undefined");
#endif
```

Проверка определений:

```
#ifndef DEBUG
    printf("Production mode");
#else
    printf("Debug mode");
#endif
```

```
#ifdef __cplusplus
extern "C" {
#endif

#ifdef __cplusplus
}
#endif
```

```
#if defined DEBUG && !defined BETA
    printf("debug mode; final version");
#elif defined DEBUG && defined BETA
    printf("debug mode; beta version");
#else
    printf("undefined mode");
#endif
```

#ifdef работает даже для определений с пустой строкой
#if для пустых строк не выполнится

[#if defined MACRO is precisely equivalent to #ifdef MACRO](#)

Стрингификация и конкатенация текста

- Превращение аргумента в текст (#)

```
#define PEVAL(cmd) printf(#cmd ": %g\n", cmd);  
double* list;  
PEVAL(list[0]); // printf("list[0] ": %g\n", list[0]);
```

- Конкатенация текста (##)

```
#define SETUP_LIST(name) int name##_len = 0;  
SETUP_LIST(mylist); // int mylist_len = 0;;
```

Предопределенные макросы

Вывести список предопределенных макросов:
Это описано [ТУТ](#) и [ТУТ](#).

```
touch dummy.c; gcc -E -dM dummy.c
```

__STDC_VERSION__

The value 199409L signifies the 1989 C standard as amended in 1994, which is the current default; the value 199901L signifies the 1999 revision of the C standard; the value 201112L signifies the 2011 revision of the C standard; the value 201710L signifies the 2017 revision of the C standard (which is otherwise identical to the 2011 version apart from correction of defects). An unspecified value larger than 201710L is used for the experimental -std=c2x and -std=gnu2x modes.

__STDC_NO_ATOMICS__

__STDC_NO_COMPLEX__

__STDC_NO_THREADS__

__STDC_NO_VLA__ // (variable length array) массивы переменной длины

__cplusplus

__FILE__, __LINE__,

__func__, __FUNCTION__

__DATE__, __TIME__

__VA_ARGS__ (C11)

```
#ifdef __cplusplus
extern "C" {
// ...
#ifdef __cplusplus
}
#endif
```

```
#define eprintf(...) fprintf (stderr, __VA_ARGS__)
```


Предопределенные макросы

Нестандартные:

```
__COUNTER__  
__GNUC__, __GNUC_MINOR__, __GNUC_PATCHLEVEL__  
__MINGW32_MAJOR_VERSION, __MINGW32_MINOR_VERSION  
__clang__  
__clang_major__, __clang_minor__, __clang_patchlevel__  
_MSC_VER // для Visual Studio 2019 16.8 _MSC_VER равно 1928  
__TIMESTAMP__ // DATA + TIME  
NDEBUG (CMake)  
_DEBUG (MSVC)  
_WIN32, _WIN64  
__unix__  
__linux__  
__MACH__, __APPLE__  
__FreeBSD__, BSD  
__ANDROID__  
__CYGWIN__
```

```
#define GCC_VERSION (__GNUC__ * 10000 \  
                    + __GNUC_MINOR__ * 100 \  
                    + __GNUC_PATCHLEVEL__)  
/* Test for GCC > 3.2.0 */  
#if GCC_VERSION > 30200
```

Добавить свои определения:

```
gcc -DA= -dM -E      #define A  
gcc -DA -dM -E       #define A 1
```

Конфигурационные директивы

```
#pragma once // предотвращение дублирования файлов
#pragma pack(bytes) // выравнивание структур (внимательно с указателями)
#pragma pack(push)
#pragma pack(push,bytes)
#pragma pack(pop)
#pragma comment(lib,"wsock32.lib") // подключить библиотеку (MSVS)
#pragma message "message/warning/error" // выдаст сообщение и т.д. (MSVS)
#pragma "warning" // выдаст предупреждение (например TODO)
#pragma "error" // вызовет ошибку компиляции
#error "Unsupported compiler" // ошибка
#pragma GCC poison printf // вызов printf выдаст ошибку
#pragma startup func1 // вызовется перед main() (MSVS)
#pragma exit func2 // вызовется перед завершением (MSVS)
#pragma hdrstop // предварительная компиляция предшествующих заголовков (MSVS, Borland)
void __attribute__((constructor)) func1(); // вызовется перед main() (GCC)
void __attribute__((destructor)) func2(); // вызовется перед завершением (GCC)
```


Диагностические директивы

`#warning "This file is deprecated!"` // удобно использовать для TODO

`#error "You should not include this file!"` // удобно использовать для каких-то особых файлов, которые нельзя подключать к данному проекту (временно или постоянно)

Компиляторозависимые макросы

- GCC

```
#pragma GCC visibility push(visibility).    visibility=default/hidden/internal/protected
#pragma GCC diagnostic                      // вывести сообщение
#pragma GCC diagnostic warning "-Wformat"  // проверка форматирования printf
#pragma GCC diagnostic error "-Wformat"    // проверка форматирования printf
#pragma GCC diagnostic ignored "-Wformat"  // проверка форматирования printf
#pragma GCC diagnostic push
#pragma GCC diagnostic pop
```

- MSVS

```
#pragma comment
#pragma comment(lib, "emapi")
#pragma deprecated(func1, func2, func3)
```


Подключение заголовочных файлов .h

Подключение заголовков

- Способ 1

```
#include <system-header.h>
#include "nonsystem-header.h"
```

- Способ 2

```
gcc -I./ -isystem /opt/include
```

- Параметризованное подключение

```
#if VERSION == 1
    #define INCFILE "vers1.h"
#elif VERSION == 2
    #define INCFILE "vers2.h"
#else
    #define INCFILE "versN.h"
#endif

#include INCFILE
```

Предкомпиляция заголовочных файлов

Что это такое

```
/* precomp.h */
#include<stdio.h>
#include<math.h>
#include<string.h>

/* main.c */
#include "precomp.h"
#include <stdlib.h>
int main(){
    printf("hello!\n");
    return EXIT_SUCCESS; }
```

```
# Makefile
all: hello
hello: precomp.h.gch main.o
    $(CC) $(LDFLAGS) -o $@ main.o
main.o: main.c
    @$ (CC) -c $< $(CFLAGS) -include precomp.h -o $@
precomp.h.gch: precomp.h
    @$ (CC) $< $(CFLAGS) -o $@
clean:
    rm -f precomp.h.gch *.o $(TARGET)
```


Рефлексия

Вопросы?




Список литературы

Роббинс Д. - Отладка приложений для Microsoft .NET и Microsoft Windows

Столяров – Азы программирования, Парадигмы

Роберт Мартин - Чистый код

[Интересное использование препроцессора для тестирования кода](#)

The background of the image is an aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. The entire image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue area, creating a digital or technological aesthetic. The text is centered within this blue area.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате



Спасибо за внимание!
Приходите на следующие вебинары

Легкоступ Виктор Валерьевич