Pokročilé praktiky v C

LinuxDays 2025

Karel Kočí

4.10.2025

https://git.cynerd.cz/presentations/tree/2025-linuxdays

Svět C

Standardy: C89, C95, C99, **C11**, C17, **C23**

Kompilátory: **GCC**, **Clang**, MSVC, a další

Standardní knihovny: **GLibC**, **Musl**, BSD, Nuttx, CRT

\+ POSIX

```
gcc -std=c23 ...
clang -std=c23 ...
```

Před C11
Tohle už asi používáte

Bitfield

```c
union pixel {
  struct c {
    unsigned r : 5;
    unsigned g : 6;
    unsigned b : 5;
  } c;
  uint16_t val;
};
union pixel p = {.c.r = 0x1b, .c.g = 0x4, .c.b = 0xa};
printf("%.4" PRIx16 " == %" PRIx16 "\n",
 p.val, 0x1b | 0x4 << 5 | 0xa << 11);
```

```
$ gcc -std=c23 bitfield.c && ./a.out
509b == 509b
```

Struktura s veřejnou částí / Rozhraní

```
struct pub { void (*print_name)(struct pub *); };
typedef struct pub *pub_t;
#define pub_print_name(PUB) (PUB)→print_name(PUB)

struct priv { struct pub pub; const char *name; };
void pname_priv(struct pub *pub) {
  struct priv *priv = (struct priv *)pub;
  printf("%s", priv→name);
}
int main(int argc, char *argv[]) {
  struct priv priv = {.pub = {pname_priv}, .name = "FOO"};
  pub_t p = &priv.pub;
  pub_print_name(p);
  putchar('\n');
```

Struktura s veřejnou částí / Rozhraní

```
struct wrap { struct pub *subpub; struct pub pub; };
void pname_wrap(struct pub *pub) {
  struct wrap *wrap = (void *)pub - offsetof(struct wrap, pub);
  printf("Wrap: ");
  wrap→subpub→print_name(wrap→subpub);
}
int main(int argc, char *argv[]) {
   ...
  struct wrap wrap = {.pub = {pname_wrap}, .subpub = &priv.pub};
  wrap.pub.print_name(&wrap.pub);
  putchar('\n');
}
```

Double-pointer na funkci / Generická funkce

```c
typedef void (*cb_t)(void *, int);
typedef cb_t *scb_t;
struct cb { cb_t cb; const char *str; };
void cb(void *self, int v) {
  struct cb *s = self;
  printf("%s: %d\n", s→str, v);
}
#define callcb(CB, ...) (*(CB))(CB, __VA_ARGS__)
  struct cb c = {.cb = cb, .str = "Hello"};
  scb_t sc = &c.cb;
  callcb(sc, 42);
```

```
$ gcc -std=c23 funcptr.c && ./a.out
Hello: 42
```

“Novinky” v C

Anonymní struct a union (C11)

```
struct point {
  union {
    struct {
      uint32_t x, y;
    };
    uint64_t u64;
  };
};

struct point a = {.x = 0x56, .y = 0x38};
printf("%.16" PRIx64 "\n", a.u64);
```

Static assert (C11)

- **C11:** `_Static_assert`
- **C23:** `static_assert`

```
  static_assert(sizeof(intmax_t) ⩾ 16,
    "Platform must support 64bit integers")
```

```
$ gcc -std=c23 static_assert.c
static_assert.c:4:1: error: static assertion failed: "Platform must
support 128bit integers"
    4 | static_assert(sizeof(intmax_t) ⩾ 16, "Platform must support
128bit integers");
      | ^~~~~~~~~~~~~
```

Atomic (C11)

```c
#include <stdatomic.h>
_Atomic _Bool a;
_Atomic int val;
_Atomic struct foo *foo;
_Bool aa = atomic_load(&a);
atomic_store(&a, false);

#if (ATOMIC_BOOL_LOCK_FREE, 2)
// atomic_bool je vždy atomický bez zámku
#elif (ATOMIC_BOOL_LOCK_FREE, 1)
// atomic_bool někdy musí použít zámek
#else
// atomic_bool vždy použije zámek
#endif
```

```c
atomic_bool a;
atomic_int val;

aa = a;
a = false;
```

Atomické operace (C11)

```c
_Bool was = atomic_exchange(&a, false);
if (!atomic_compare_exchange_strong(&a, &was, true))
  printf("Somebody changed it in the meantime\n");

int iwas = atomic_fetch_add(&val, 1);
iwas = atomic_fetch_sub(&val, 1);
iwas = atomic_fetch_or(&val, 0xf);
iwas = atomic_fetch_xor(&val, 0x10);
iwas = atomic_fetch_and(&val, 0xff);
```

Atomický příznak (C11)

```
atomic_flag event = ATOMIC_FLAG_INIT;
_Bool was = atomic_flag_test_and_set(&event);
atomic_flag_clear(&event);
```

Vlákna a další (C11)

- `<threads.h>`
- `thrd_t`, `mtx_t`, `cnd_t`
- `thread_local` (`_Thread_local`) a `tss_t`

Detekce přetečení a podtečení (C23)

```
int8_t add, sub, mul;
assert(ckd_add(&add, 50, 85));
assert(ckd_sub(&sub, 10, 150));
assert(ckd_mul(&mul, 68, 2));
```

Pozor: `true` = nevalidní výsledek

Bitové operace (C23)

```c
#include <stdbit.h>
assert(stdc_leading_zeros(0xffU) == 24);
assert(stdc_leading_ones(0xff000000U) == 8);
assert(stdc_trailing_zeros(0xff000000U) == 24);
assert(stdc_trailing_ones(0xffU) == 8);
assert(stdc_first_leading_zero(0xff000000U) == 9);
assert(stdc_first_leading_one(0xffU) == 25);
assert(stdc_first_trailing_zero(0xffU) == 9);
assert(stdc_first_trailing_one(0xff000000U) == 25);
assert(stdc_count_zeros(0xffU) == 24);
assert(stdc_count_ones(0xffU) == 8);
assert(stdc_bit_width(0xf0U) == 8);
assert(stdc_bit_floor(0xf1U) == 0x80);
assert(stdc_bit_ceil(0xf1U) == 0x100);
```

_Generic (C11)

```c
#define traceval(X) \
  fprintf(stderr, _Generic((X), \
    char *: "%s\n", int: "%d\n", void *: "%p\n"), X)

  traceval("Hello");
  traceval(42);
  traceval(NULL);
```

```
$ gcc -std=c23 generic.c && ./a.out
Hello
42
(nil)
```

Pozor: Všechny varianty musí být kompilovatelné.
Každá varianta generuje errory i warningy!

Attributy (C23)

Bylo `__attribute__( ... )` a nyní `[[ ... ]]`.

Nestandardní C
C podle kompilátoru

Warning - switch

```c
int val = fgetc(stdin);
switch (val) {
  case ' ': printf("Secret space.... ");
  default: printf("Got: %c\n", val);
}
```

```
$ gcc -Wimplicit-fallthrough -Wall -std=c23 warn-switch.c && ./a.out
warn-switch.c: In function ‘main’:
warn-switch.c:7:5: warning: this statement may fall through [-
Wimplicit-fallthrough=]
    7 |     printf("Secret space.. ");
      |     ^~~~~~~~~~~~~~~~~~~~~~~~
warn-switch.c:8:3: note: here
    8 |   default:
      |   ^~~~~~~
```

Warning - switch

```
int val = fgetc(stdin);
switch (val) {
  case ' ':
    printf("Secret space.... ");
    [[gnu::fallthrough]];
  default:
    printf("Got: %c\n", val);
}
```

Warningy

- `[[gnu::fallthrough]]`
- `[[gnu::nonstring]]`
- `[[gnu::unused]]`

nonnull

```c
[[gnu::nonnull]] // [[gnu::nonnull(1)]]
void prnt(char *str) { printf("%s\n", str); }
  prnt("Some text");
  prnt(NULL);
```

```
$ gcc -std=c23 nonnull.c && ./a.out
nonnull.c: In function ‘main’:
nonnull.c:10:3: warning: argument 1 null where non-null expected [-
Wnonnull]
   10 |   prnt(NULL);
      |   ^~~~
nonnull.c:4:6: note: in a call to function ‘prnt’ declared ‘nonnull’
    4 | void prnt(char *str) {
      |      ^~~~
Some text
(null)
```

Další kontroly

- `[[clang::counted_by]]`, `[[gnu::counted_by]]`
- `[[gnu::malloc]]`
- `[[gnu::alloc_size]]`
- `[[gnu::fd_arg]]`, `[[gnu::fd_arg_read]]`, `[[gnu:fd_arg_write]]`
- `[[gnu:: format]]`, `[[gnu:: format_arg]]]`

https://gcc.gnu.org/onlinedocs/gcc/Common-Function-Attributes.html
https://gcc.gnu.org/onlinedocs/gcc/Common-Variable-Attributes.html
https://clang.llvm.org/docs/AttributeReference.html

Automatické uvolnění z heapy

```
static void _free(char **ptr) {
  printf("Freeing %p=%p\n", ptr, *ptr);
  free(*ptr);
}
  [[gnu::cleanup(_free)]] char *buf = malloc(18);
  [[gnu::cleanup(_free)]] char *null = NULL;
  printf("Pointers: %p=%p %p=%p\n", &buf, buf, &null, null);
```

```
$ gcc -std=c23 cleanup.c && ./a.out
Pointers: 0×7fffffdef68=0×4052a0 0×7fffffdef70=(nil)
Freeing 0×7fffffdef70=(nil)
Freeing 0×7fffffdef68=0×4052a0
```

Weak

```
[[gnu::weak]]
void weakfun(void) {
  printf("Weak function\n");
}
  weakfun();

void weakfun(void) { printf("Standard function\n"); }
```

```
$ gcc -std=c23 weak.c && ./a.out
Weak function
$ gcc -std=c23 weak.c weakfun.c && ./a.out
Standard function
```

Transparent union

```
union [[gnu::transparent_union]] un {
  void *ptr;
  intptr_t i;
};
void pasi(union un v) { printf("%p\n", v.ptr); }
  pasi((intptr_t)0x42);
  pasi((void *)&pasi);
```

```
$ gcc -std=c23 transunion.c && ./a.out
0×42
0×401150
```

Transparent union callback

```
union [[gnu::transparent_union]] un {
  char *ptr;
  intptr_t i;
};
void call(void (*cb)(union un)) { cb("FOO"); }
void cb(char *str) { printf("%s\n", str); }
  call(cb);
```

```
$ gcc -std=c23 transunionf.c && ./a.out
FOO
```

Constructor a Desctructor

```
[[gnu::constructor]]
static void _our_constructor(void) { printf("Constructor\n"); }
[[gnu::destructor]]
static void _our_destructor(void) { printf("Destructor\n"); }
int main(int argc, char *argv[]) {
  printf("main\n");
}
```

```
$ gcc -std=c23 constructor.c && ./a.out
Constructor
main
Destructor
```

Switch range (GCC, Clang, ...)

```c
int val = getchar();
switch (val) {
case '0' ... '9':
  printf("Number\n");
break;
default:
  printf("Not a number\n");
  break;
}
```

Děkuji za pozornost

Karel Kočí

https://git.cynerd.cz

https://gitlab.com/cynerd

https://git.cynerd.cz/presentations/tree/2025-linuxdays