Libera Longo 2023-10-19 Sistemi operativi

cravatta chitarra

ANSI C

ANSI: istituto/ente Americano

IEEE ISO

http://www.cs.unibo.it/~renzo/so/materiale2324/domande/20230928_domande_unix_c

size of (int) = 4

(long) = 8

(int *) = 8

non su tutte le macchine

Arduino = 2

32 bit = 4

64 bit = 4

sizeof viene valutato dal compilatore → "costante" con codice portabile

puntatori

```
int v = 42
int *p = &v
*p += 3
```

p puntatore

controlli nel uso del linguaggio C

v | 42

p | 3456

NULL valore 0 puntatore invalido

fine catena puntatori (lista)

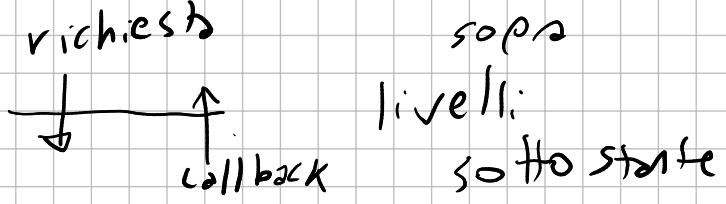
void

```
int f();          i parametri non controllarli
int f(void);      controlla i parametri e ti dico io che non ne ho
```

void * tipo predefinito

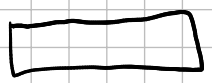
vuol dire "puntatore generico"

programmazione a livelli? prob. reti protocolli

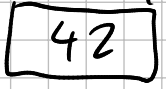


aritmetica dei puntatori

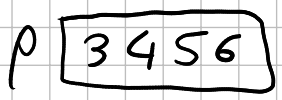
- puntatore ± intero ritorna il puntatore al intero successivo + 4 byte



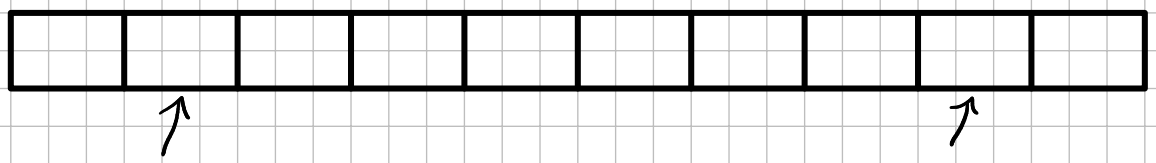
```
int v = 42
int *p = &v
```



elementi a fianco a v



- puntatore − puntatore ritorna un intero quanti elementi tra l'uno e l'altro



quindi:

puntatore ± intero → puntatore

puntatore − puntatore → intero

incremento / decremento puntatore

```
p++                 *(++p) = 45
p = p+1             int v = 42;
                    int *p = &v;
                    long l = (long) p;   brutto
                    int ip = (int) p;    assumo
```

p è a 64 bit ip prenderebbe i 32 bit meno significativi del indirizzo

come convertire? tipi definiti su <stdint.h>

uintptr_t

uint64_t deve essere 64 bit!

int16_t (non mi interessa "long")

uintptr_t uint pointer type

↳ sempre 8 byte

cambia in funzione del architettura ma non del tipo puntatori

Arduino è una macchina di Harvard non di Von Neumann

come stampare puntatori

%d %s %p voglio stampare un puntatore

strutture/ unioni, operatore . (punto)

```
struct point {
    int x;
    int y;
};
```

sizeof point mi aspetto 8 byte

4 byte 4 byte

```
struct point p0
p0.x = 2; p0.y = 3;
```

assegnare in una struct il campo

struct point *pp = &p0;

puntatori a structure, operatore →

pp | 4422

(*pp).x

pp → x

(*pp).x si scrive abbreviato pp->x

le strutture si possono assegnare

struct point p1;

p1 = p0;

i campi vengono copiati

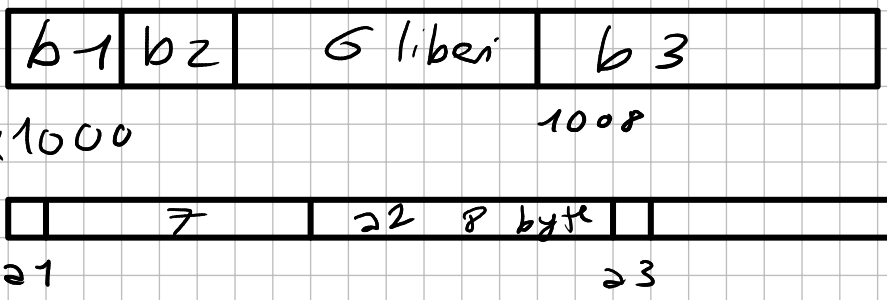
il compilatore chiama la memcopy

- https://www.cs.unibo.it/~renzo/so/ Esperimenti lezione_live(exp)

http://soho.cs.unibo.it:8888/

```
//file structcp.c
#include <stdio.h>
int main(int argc, char *argv[]) {
        struct point {
                int x;
                int y;
        };

        struct point p0, p1;

        p0 = p1;
}


//struni.c
#include <stdio.h>

union intch {
        int i;
        struct {
                char c0;
                char c1;
                char c2;
                char c3;
        };
};

int main(int argc, char *argv[]) {
        union intch ic;
        ic.i = 0x01020304;
        printf("%d %d %d %d\n", ic.c0, ic.c1, ic.c2, ic.c3);
}
```

alignment.c



struct A { ... }__attribute__((packed));

allineato tutto → inefficiente

ma entrambi sarebbero 10 byte (struct A, B)

struct a sa

sa.a2 = 10

st 64 store di 64

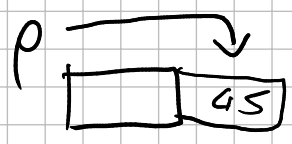
=> store di un long

```
//alignx.c
#include <stdio.h>

struct A {
        char a1;
        long a2;
        char a3;
};
//}__attribute__((packed));

int main() {
        struct A sa;
        sa.a2 = 44;
}
```

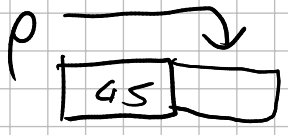
```
libera@DEBIAN:~/so_lezioni/lecture_examples/20231019$ ls
alignment.c  alignx.c  structcp.c  struni.c
libera@DEBIAN:~/so_lezioni/lecture_examples/20231019$ gcc -S alignx.s alignx.c
libera@DEBIAN:~/so_lezioni/lecture_examples/20231019$ cat alignx.s
        .file   "alignx.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movq    $44, -24(%rbp)
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Debian 12.2.0-14) 12.2.0"
        .section        .note.GNU-stack,"",@progbits
```

p a us a

*(++p) = 45



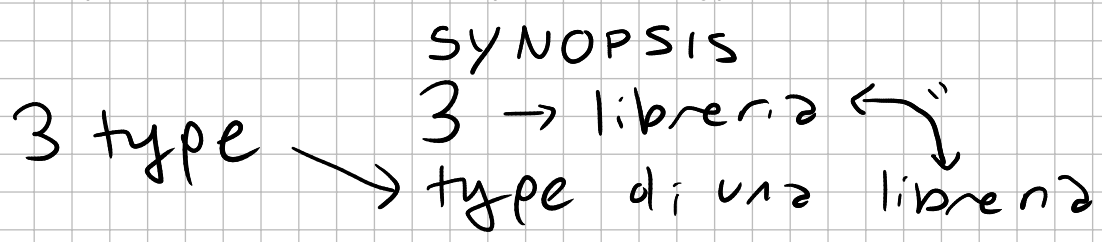
invece *(p++) = 45

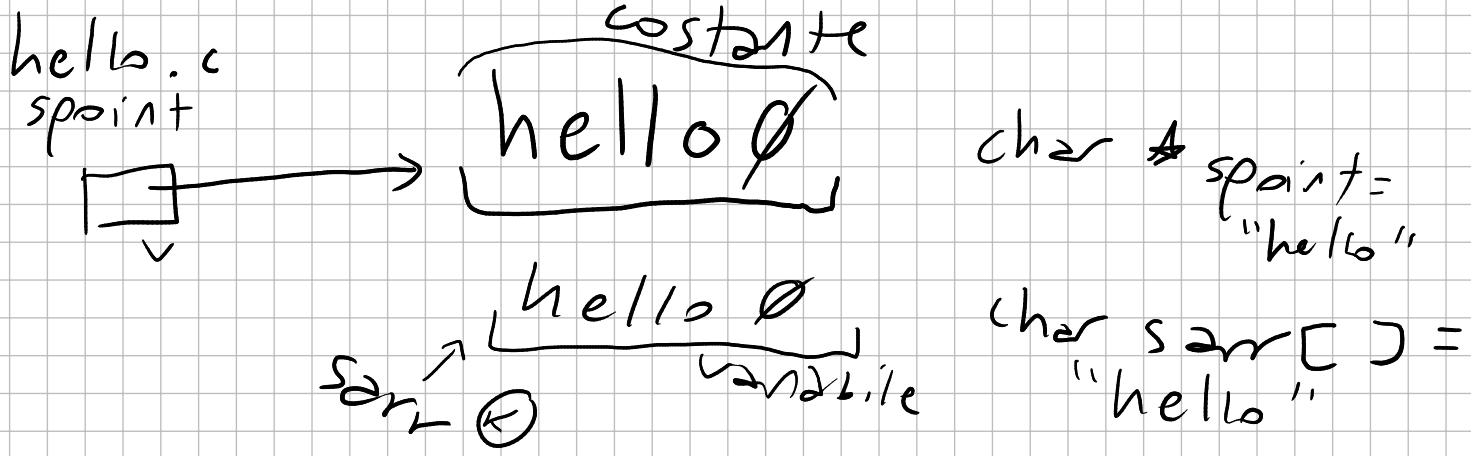


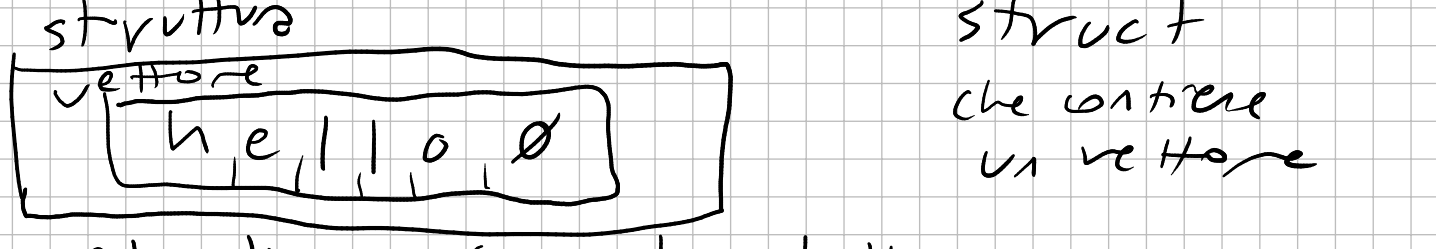
struct vs union ?

vediamo esempi facendo man ... indipendentemente da cosa fanno sti qua...

man epoll_ctl(2) man epoll_event(3type)







le strutture come ho detto prima viene copiata.

```c
#include <stdio.h>

char *spoint="hello";
char sarr[]="hello";
struct strs {
        char s[6];
} sstruct = {"hello"};

void foo(char *s) {
        s[4]=0;
}

void bar(struct strs s) {
        s.s[4]=0;
        printf("from bar %s\n", s.s);
}

int main(int argc, char *argv[]) {
        printf("%s %s %s\n", spoint, sarr, sstruct.s);
        foo(sarr);
        printf("%s %s %s\n", spoint, sarr, sstruct.s);
        bar(sstruct);
        printf("%s %s %s\n", spoint, sarr, sstruct.s);
        // test the following statements, one at a time
        //spoint = sarr;
        //sarr = spoint;
        foo(spoint);
        printf("%s %s %s\n", spoint, sarr, sstruct.s);
}
```

ok, ed evita segmentation fault

x fa cambiare una costante

si arrabbia

di questa riga

assegnamento di strutture/union: copia

costruttori di strutture

```
struct point p0 = {.y = 4, .x = 3};
union intch ic = {.c1 = 1, .c2 = 2};
void printpoint ... → File
                      structcp2.c
```

campi senza nome in strutture/union:

```
struct mystruct;
```

↳ tipi incompleti di strutture/union

virtualsquare.org

https://github.com/rd235/libfduserdata/blob/master/fduserdata.h

```
man fopen
```

Array, operatore []

```
int v[10];
int *p = v
```

v[3] = 10 significa *(v+3) = 10

p[3] = 5

oppure

```
int *p = v+1
```

scalati di 1

myecho.c myecho2.c

```
//ar1.c
void vv(int v[]) {
int main(int argc, char *argv[]) {
        //int v[10] = {1,2,3,4};
        int v[10] = {[3 ... 9] = 33, [7] = 35};

        for (int i = 0; i < sizeof(v)/sizeof(*v); i++)
                printf("%d ", v[i]);
        printf("\n");
}
```

```c
//arraypre.c
#include <stdio.h>

#define rows_of(X) (sizeof(X) / sizeof((X)[0]))

#define printTable(X) do { \
    int i; \
    printf("TABLE " #X ": size of element %d\n" \
        "(printed by the line %d of source file %s)\n",  \
        sizeof(*(table ## X)), __LINE__, __FILE__); \
    for (i = 0; i < rows_of(table ## X); i++)  \
      printf(#X " %02d %s\n",i,table ## X [i]); \
    } while (0)

char tableA[][50] = {"Sempre caro mi fu quest'ermo colle,",
  "e questa siepe, che da tanta parte",
  "dell'ultimo orizzonte il guardo esclude."};
char *tableB[] = {"Sempre caro mi fu quest'ermo colle,",
  "e questa siepe, che da tanta parte",
  "dell'ultimo orizzonte il guardo esclude."};

int main(int argc, char *argv[1]) {
  int i;
  printTable(A);
  printf("\n");
  printTable(B);
}
```

t[1][2] = "q"

2° riga

[1]

[2]

3° carattere

⇒ "q"

#X

parametro

"A"

giustapponendo

preprocessore (quello che interpreta le #include #define ecc) come l'hai capita?

gcc -E arraypre.c

"A" "B"

"AB"

table ## X → preprocessore (è "editor automatico" .c → .c editato)

__LINE__ } compilatore

__FILE__ }

__LINE__ linea di: printTable(A) → 23

__FILE__ nome del file → "arraypre.c"