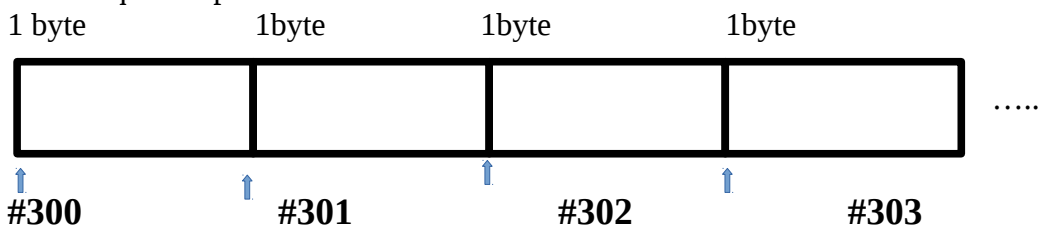


Com treballar amb un array de bytes.

1-. Punters amb C

Tenim aquest espai de memòria:



```
int a;  
char b;  
char *c;
```

De quin tipus és **a**?

(int) a → (int)
↑ ↓
a = 2

Com modifiquem el valor de **a**?

De quin tipus és **b**?

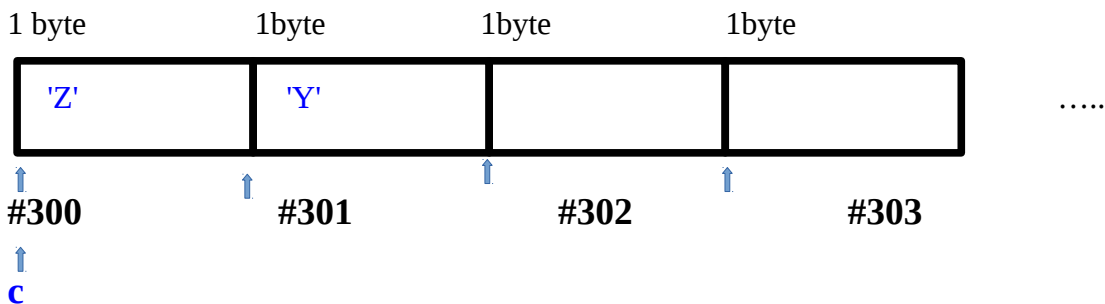
(char) b → (char)
↑ ↓

Com modifiquem el valor de **b**?

b = 'A' //b = 65;

De quin tipus és **c**?

(char *) c → (char *)



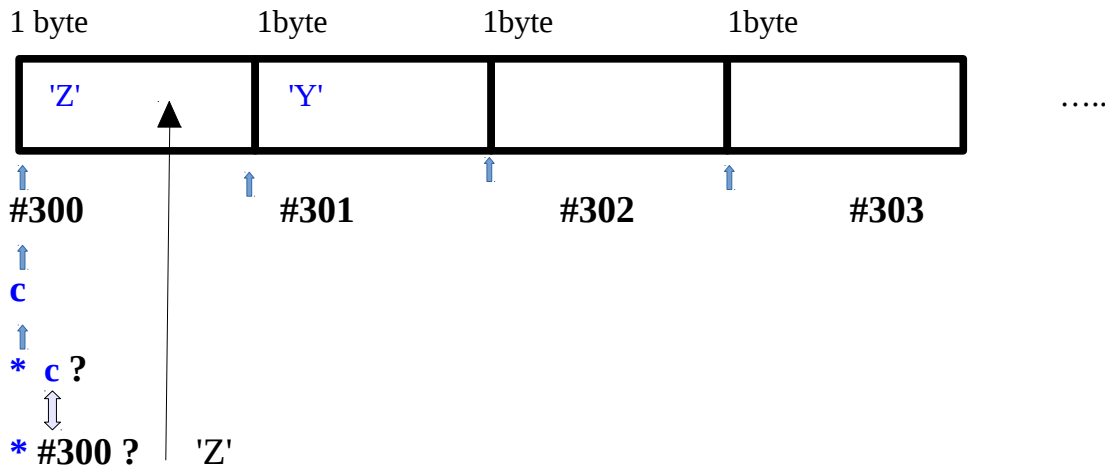
Quin valor té **c**? `printf("%d", c);`

300 (posició de memòria #300)

$$\begin{array}{ccc} (\text{char} *) c & \longrightarrow & (\text{char} *) \\ \updownarrow & & \updownarrow \\ c & = & 300 \end{array}$$

c = 300

(char) (*c) \longrightarrow (char)
 \updownarrow
 *c = 'Z'

$$*_C = 'Z'$$


* Com declarem un array de bytes?

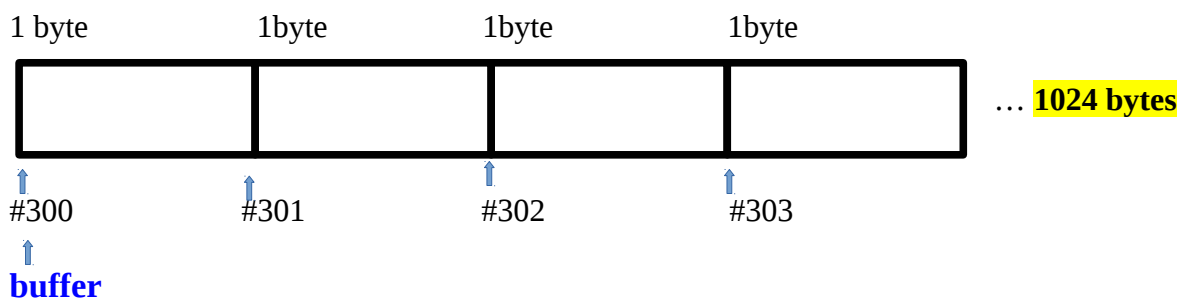
```
char buffer[1024]; ~ = { char * buffer1;
                          buffer1 = malloc(1024);
```

Memòria dinàmica

```
void foo(char *param1){....}
```

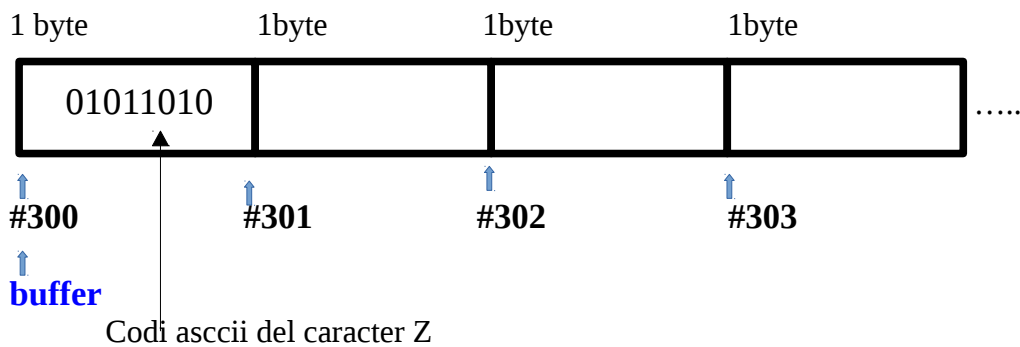
foo(buffer);		Ok
foo(buffer1);		Ok

* Què val buffer?



`printf("%d", buffer);` \longrightarrow 300 (posició de memòria #300)

* Com accedim a una posició de memòria?



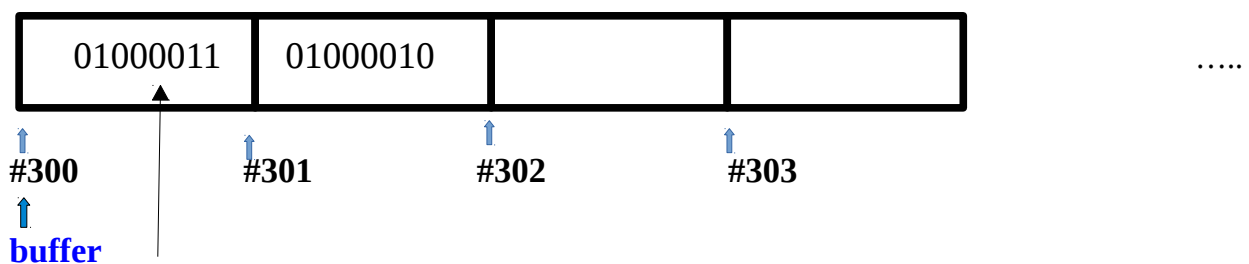
`*buffer` ➡ 'Z'
=
`buffer[0]` ➡ 'Z'

Exemple1:

`printf("%c", *buffer);` ➡ 'Z'
=
`printf("%c", buffer[0]);` ➡ 'Z'

Exemple2:

`*buffer = 'c';`
=
`buffer[0] = 'c';`



Codi ascii del caracter C

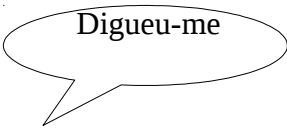
*Com poso una D en la posició #301?

Em moc una posició: `#300+1=#301` `printf("%d", buffer+1)` ➡ #301
Contingut de la nova posició: `*(#301) = 'd'` `printf("%c", *(buffer+1))` ➡ d
=
`buffer[1] = 'd';`

*** Com desem un short en el nostre array de bytes?**

```
char buffer[1024];  
unsigned short code = 1;
```

1-. Com desem la variable code en buffer per enviar-la per la xarxa?



Digueu-me

3 maneres:

1-. Utilitzant la funció memcpy:

```
void *memcpy(void *dest, const void *src, size_t n);
```

```
char buffer[1024];
unsigned short code = 1;
code = htons(code); //passem el short a format xarxa!!
memcpy(buffer, &code, sizeof(unsigned short));
```

2-. Utilitzant la macro del **common.h**:

```
#define stshort(sval, addr) ( *((short *) (addr))=htons(sval) )
```

```
char buffer[1024];
unsigned short code = 1;
stshort(code, buffer); //Ja ho ha transformat en format xarxa!!!
```



(
Quants bytes podem escriure en les següents variables?

```
int a;
a = 2;
```

```
char b;
b = 'Z';
```

```
int * c = malloc(sizeof(int));
*c = 3;
```

```
char * d = malloc(sizeof(int));
*d = 'Z'; //d[0]='Z'
```

```
short * e = malloc(sizeof(short));
*e = 4;
```

)

*** Analitzem la macro stshort(code,buffer)**

Stores a **short value** in a **memory** position.

```
#define stshort(sval, addr) ( *((short *) (addr))=htons(sval) )
```

↑
variable de tipus short que volem desar en una posició, adreça, de memòria.

```
#define stshort(sval, addr) ( *((short *) (addr))=htons(sval) )
```

↑
Posició, adreça, de memòria on volem desar **sval**.

```
#define stshort(sval, addr) ( *((short *) (addr))=htons(sval) )
```

De quin tipus és **addr** ↑ ?

En el nostre cas concret?

En el codi:

```
char buffer[1024];
stshort(code, buffer);
```

Quin efecte té el següent cast?

```
#define stshort(sval, addr) ( *((short *)(addr))=htons(sval) )
```

Com escric dos bytes en una posició de memòria que apunta a un short?

```
short * e = malloc(sizeof(short));
```

```
*e = 4;
```

* (posició de memòria) = 2 bytes

```
#define stshort(sval, addr) ( *((short *)(addr))=htons(sval) )
```

Com convertim un short de format host a format network?

```
#define stshort(sval, addr) ( *((short *)(addr))=htons(sval) )
```

Hem desat en la posició de memòria **addr** un **short** convertit en **format xarxa** 😊

3-. Aprofitant el que hem après de la macro:

```
char buffer[1024];
```

```
unsigned short code = 1;
```

```
*((short *)buffer) = htons(code); //passem el short a format xarxa!!
```

** Com desmem dos shorts en el nostre array de bytes?*

```
char buffer[1024];
```

```
unsigned short code = 1;
```

```
unsigned short field1 = 2;
```

** Com enviem el nostre array de bytes?*

```
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
```

```
send(s, buffer, sizeof(short), 0);
```

** Com llegim l'array de bytes que rebem?*

2 maneres:

1-. Utilitzant la funció memcpy:

```
void *memcpy(void *dest, const void *src, size_t n);
```

```
char buffer[1024];
```

```
unsigned short code;
```

```
recv(s, buffer, sizeof(buffer), 0);
```

```
memcpy(code, buffer, sizeof(buffer));
```

```
code = ntohs(code);
```

2-. Aprofitant el que hem après de la macro **stshort**:

```
char buffer[1024];
```

```
unsigned short code;
```

```
recv(s, buffer, sizeof(buffer), 0);
```

```
code = ntohs( *((short *)buffer) ); //convertim de format network a format host
```

3-. Utilitzant la macro del common.h:

```
#define ldshort(addr) ( ntohs(*(short*)(addr)) )
```

```
#define ldshort(addr) ( ntohs(*(short*)(addr)) )
```

```
#define ldshort(addr) ( ntohs(*(short*)(addr)) )
```

```
#define ldshort(addr) ( ntohs(*(short*)(addr)) )
```



Retorna un short, ja convertit en format host, emmagatzemat en el punter **addr**.

```
char buffer[1024];  
unsigned short code;
```

```
recv(s, buffer, sizeof(buffer), 0);  
code = ldshort(buffer);
```



2-. Com hem d'utilitzar les estructures de l'esquelet?

Cada una de les estructures cosisteix en l'espai de memòria necessari per emmagatzemar les dades d'un o varis camps del missatge que heu d'enviar.

Per exemple:

```
/*  
    2 bytes  11 bytes   1 byte  
-----  
HELLO_RP    | 2      | Hello World | 0 |  
-----  
*/  
struct hello_rp{  
    unsigned short opcode;  ← 2 bytes  
    char msg[12];          12 bytes: 11 + 1  
};
```

```
/*  
DD_rule_espec: 2 bytes 4 bytes 2 bytes 2 bytes  
-----  
| 0 | net_ID | net_mask | dev      |  
-----  
*/  
struct direct_delivery_rule{  
    unsigned short type;      ← 2 bytes  
    struct in_addr netID;     ← 4 bytes  
    unsigned short mask;     ← 2 bytes  
    unsigned short device;   ← 2 bytes  
};
```

Cada camp de l'estructura coincideix amb un camp en l'array de bytes que heu d'enviar o rebre.

2.1-. Com escrivim les estructures en un array de bytes que volem enviar?

Aprofitem el que hem après de les macros:

```
char buffer[1024];
struct hello_rp hello;

//ompliu els camps de l'estructura.
//recordeu que el opcode ha d'estar en format xarxa: htons()
*((struct hello_rp *)buffer) = hello;
//expliquem-ho:
*((struct hello_rp *)buffer) = hello;
*((struct hello_rp *)buffer) = hello;
*((struct hello_rp *)buffer) = hello;
```

2.1.2-. Com avançaríeu el punter per escriure alguna cosa després d'haver escrit l'estructura hello_rp en el buffer?

```
int offset=0;
offset += sizeof(struct hello_rp);
//això em posicionaria el buffer en la posició 14. Conte que comencem en la posició 0.
```

2.2-. Com llegim les estructures d'un array de bytes que acabem de rebre?

Aprofitem el que hem après de les macros:

```
char buffer[1024];
struct hello_rp hello;

//recv(s, buffer, ....)
hello = *((struct hello_rp *)buffer);
```


3-. Com hem de treballar amb adreces

```
struct in_addr {  
    unsigned long s_addr;  
};
```

3.1-. Si tenim una cadena amb una IP en format decimal dotted quad: 158.109.79.123 Per convertir-la en **long** i en **format xarxa** :

/** Funció que converteix una adreça internet expressada com una cadena en format dotted quad (paràmetre **cp**) de cadena dotted quad a long (en format xarxa). Desa aquest long en el paràmetre **inp**.

* @param **cp**: Cadena en format dotted quad que volem transformar. Per exemple: "158.109.79.136".

* @param **inp** Estructura *struct in_addr* on volem desar l'adreça en format xarxa.

*/

```
int inet_pton(const char *cp, struct in_addr *inp);
```

/* inet_ → internet */

/* _pton → ascii to network */

struct in_addr address;

char *ip = "158.109.79.64";

inet_pton(ip, &address);

3.2-. Si tenim una adreça en format xarxa i la volem en en format decimal dotted quad: 158.109.79.123

/** Funció que converteix l'adreça internet especificada com a paràmetre '**in**', que està en format xarxa, de format xarxa a cadena de caràcters.

* @param **in**: Adreça en format xarxa que volem convertir en cadena de caràcters.

* @return La cadena de caràcters en format dotted quad que representa l'adreça passada

* com a paràmetre.

*/

```
char *inet_ntoa(struct in_addr in);
```

/* inet_ → internet */

/* _ntoa → network to ascii */

char ip[16]; //cadena de caràcters amb l'adreça en format dotted quad

struct in_addr address; //on tenim la l'adreça en format xarxa.

strcpy(address, inet_ntoa(address));