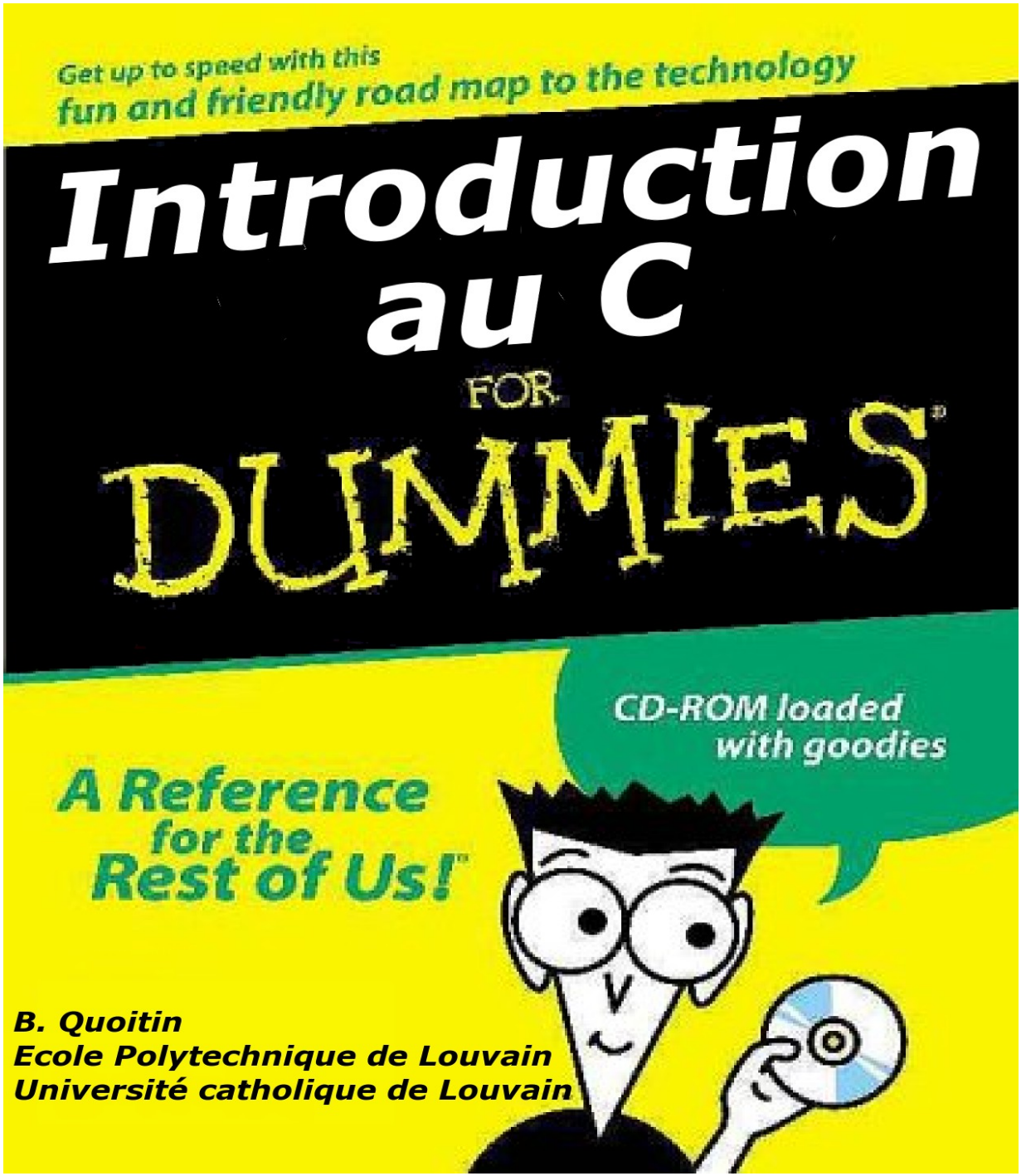

Get up to speed with this
fun and friendly road map to the technology
Introduction au C
FOR
DUMMIES
CD-ROM loaded with goodies
A Reference for the Rest of Us!
B. Quoitin
Ecole Polytechnique de Louvain
Université catholique de Louvain

Table des matières

Les pointeurs, un jeu de piste ?

Les pointeurs, un jeu de piste ?

- Résolution de `toIntPlus()`

```
#include <stdio.h>
#include <string.h>

int toIntPlus(char * str, char ** badcar_ptr)
{
  ????
}

int main() {
  char str[9]= "1234a";
  char * badcar_ptr;
  int resultat= toIntPlus(str,    ????    );
  if (    ????     )
    printf("Erreur de conversion.\n");
  else
    printf("Résultat: %d\n", resultat);
  return 0;
}
```

Les pointeurs, un jeu de piste ?

- Résolution de `toIntPlus()`

```
#include <stdio.h>
#include <string.h>

int toIntPlus(char * str, char ** badcar_ptr)
{
   ????
}

int main() {
   char str[9]= "1234a";
   char * badcar_ptr;
   int resultat= toIntPlus(str,    ????    );
   if (    ????     )
      printf("Erreur de conversion.\n");
   else
      printf("Résultat: %d\n", resultat);
   return 0;
}
```

Je vois...
Je voiiisss...
Oui, je vois un '&'
suivi par deux ''*

Les pointeurs, un jeu de piste ?

NON !!!

Exercices

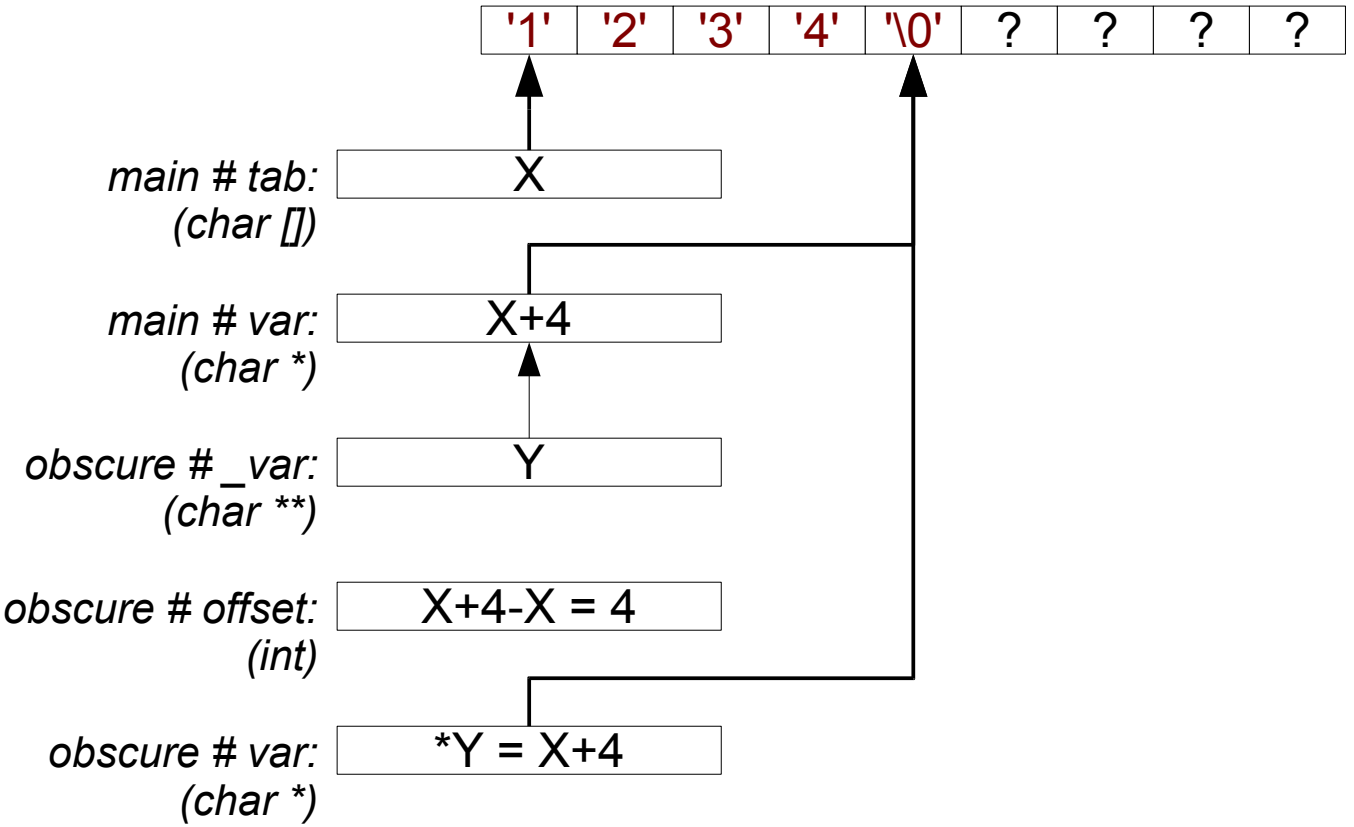
```
char tab[32]= "Yohoho, une bouteille de rhum !";

int var1= tab[1]-tab[2];

int var2= &tab[2]-&tab[1];

char * var3= tab + strlen(tab);

int * var4= (int *) tab;

int * var5= (int *) tab[1];

char c1= *var3;

char c2= *((char *) var4);

char c3= *(tab + 10);

char c4= *((char *) (var4 + 2));
```

Exercice

```c
#include <stdio.h>
#include <string.h>

void obscure(char ** _var, int offset) {
  char * var= *_var;
  char i= 0;
  for (i= 0; i < offset; i++) {
    *var= *(var-offset)+offset;
    var++;
  }
  *var= var-(*_var)-offset;
}

int main() {
  char tab[9]= "1234";
  char * var= tab+strlen(tab);
  obscure(&var, var-tab);
  printf("Résultat: \"%s\"\n", tab);
  return 0;
}
```

Exercice

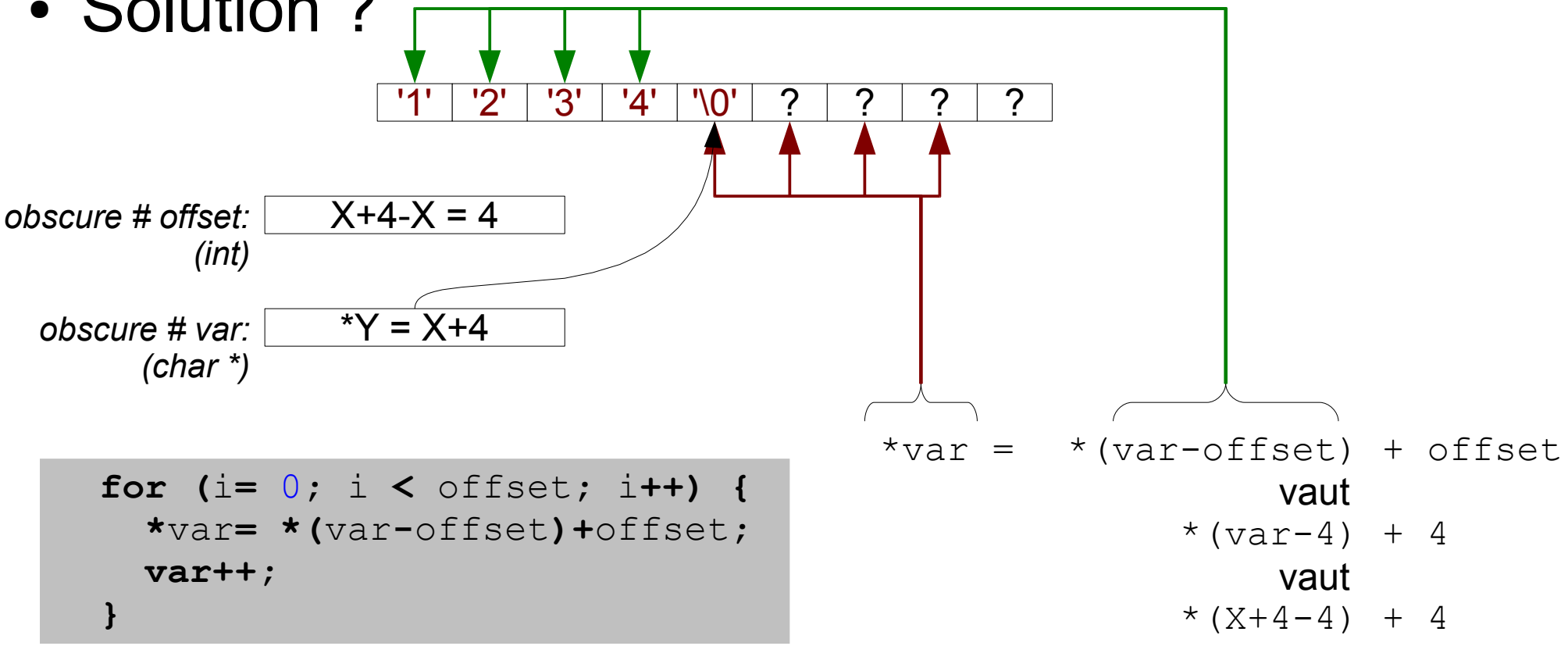
- Solution ?



Note: X et Y sont deux adresses en mémoire.
Il n'est pas nécessaire de connaître leurs valeurs réelles.

Exercice

- Solution ?




```
for (i= 0; i < offset; i++) {
  *var= *(var-offset)+offset;
  var++;
}
```

Manipulations de bits

Manipulations de bits

- Opérateur AND (&)

23		51		?
00010111	&	00110011	=	????????

23	00010111
51	00110011
19	00010011

Manipulations de bits

- Opérateur OR (|)

23		51		?
00010111	|	00110011	=	????????

23	00010111
51	00110011
55	00110111

Manipulations de bits

- Opérateur XOR (^)

23		51		?
00010111	^	00110011	=	????????

23	00010111
51	00110011
36	00100100

Manipulations de bits

- Opérateur NOT (~)

~	23
	00010111

=

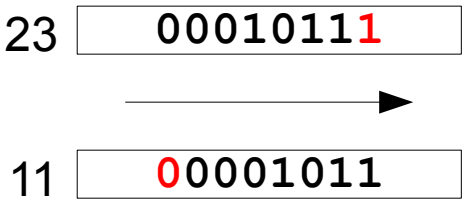
?
????????

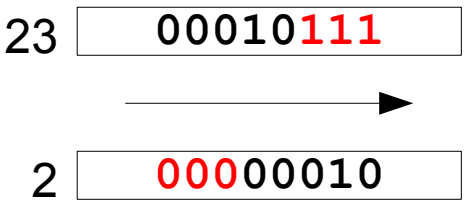
23	00010111

232	11101000

Manipulations de bits

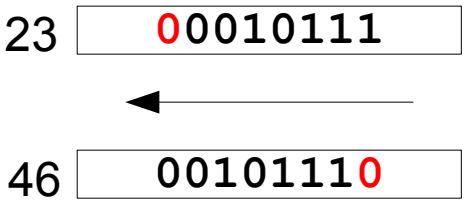
- Décalage à droite (>>)

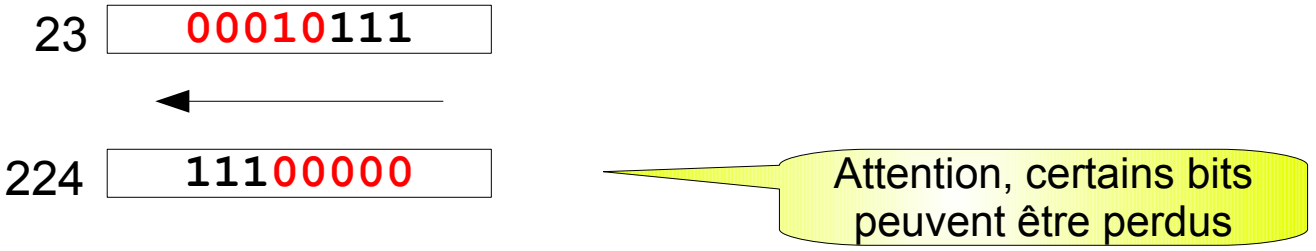




Manipulations de bits

- Décalage à gauche (<<)





Manipulations de bits

- Décalage de bits

`x >> n` est équivalent à $x / 2^n$

`x << n` est équivalent à $x * 2^n$

Attention, c'est valable uniquement si la taille de x est suffisante

- En particulier

`1 << n` est équivalent à 2^n

Manipulations de bits

- Déterminer la valeur d'un bit

```
unsigned int num= 68;
char bit;

for (bit= sizeof(num)*8-1; bit >= 0; bit--) {
  if (num & (1 << bit))
    printf("1");
  else
    printf("0");
}
```

Question subsidiaire: que se passe-t-il si on remplace le type de la variable '`bit`' par '`unsigned char`' (au lieu de '`char`') ?

Manipulations de bits

- Changer la valeur d'un bit

```
unsigned int num= 68;
char bit;

bit= 3;
num= num | (1 << bit);

bit= 6;
num= num & ~(1 << bit);
```

Manipulations de bits

- Gérer une liste d'options binaires (ON / OFF)

```
#define OPTION_1    0x01
#define OPTION_2    0x02
#define OPTION_3    0x04
#define OPTION_4    0x08
#define OPTION_5    0x10
#define OPTION_6    0x20
#define OPTION_7    0x40
#define OPTION_8    0x80

unsigned char options;

options= OPTION_2 | OPTION_6;

if (options & OPTION_6)
  printf("L'option 6 est ON.");
else
  printf("L'option 6 est OFF.");
```

Manipulations de bits

- Exemple concret: `open()`

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

int fd= open("your_mind", O_WRONLY | O_CREAT, 0644);
if (fd < 0) {
  perror("Erreur: échec de création de fichier");
  exit(EXIT_FAILURE);
}
```

Si on regarde dans `/usr/include/fcntl.h`

```
#define O_RDONLY    0x0000
#define O_WRONLY    0x0001
#define O_CREAT     0x0200
```

Question subsidiaire: quels seront les droits d'accès au fichier créé ?

Précisions

Précisions

- Inclusion de fichiers headers

```
#include <fcntl.h>

#include "mon_header.h"
```

Recherche dans le même répertoire que le fichier qui fait l' `#include`

- Chemin de recherche

```
gcc -Ichemin1 -Ichemin2 ...

gcc -Wall -Werror -DDEBUG -I. -o ex5_1 ex5_1.c

gcc -Wall -Werror -Isrc/include -o bin/prog src/prog.c
```

Suggestions d'exercices

Exercice 1

- Gestion de liste chaînée triée

```
#ifndef __LIST_H__
#define __LIST_H__

typedef struct list_t {
  char *           word;
  struct list_t * next;
} list_t;

/** Initialise la racine d'une liste */
void list_init(list_t * list);

/** Libère la mémoire d'une liste */
void list_free(list_t * list);

/** Retourne le nombre de mots stockés dans la liste. */
unsigned int list_size(list_t * list);

... (suite à la page suivante)
```

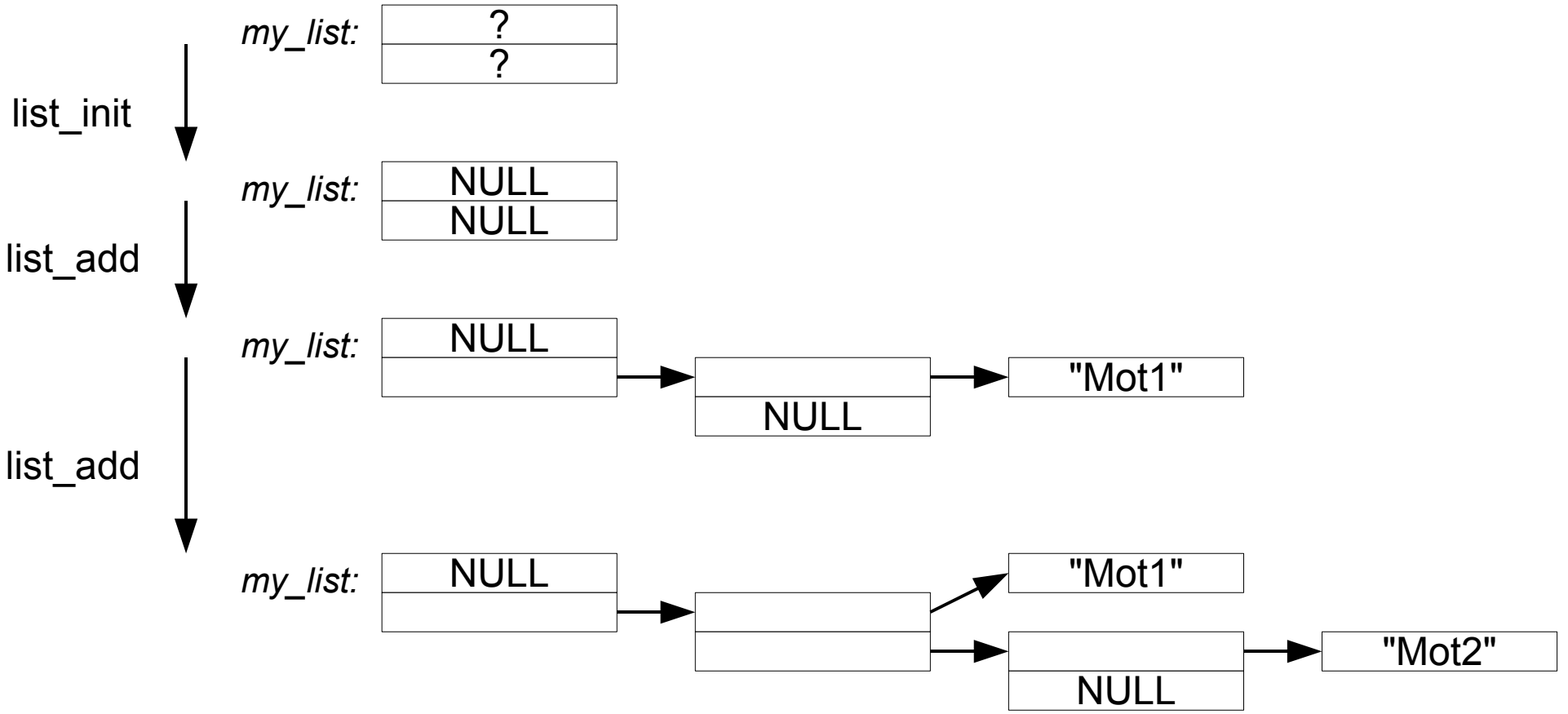
Exercice 1 (suite)

- Gestion de liste chaînée triée

```
(suite de la page précédente) ...

/** Ajoute un mot dans la liste. Insertion conformément à
 *  l'ordre lexicographique sur les chaînes de cars.
 *  Un mot peut être ajouté plusieurs fois. */
void list_add(list_t * list, const char * word);

/** Retourne la position du mot dans la liste.
 *  Retourne un index négatif si le mot n'est pas dans la
 *  liste.
 */
int list_indexof(list_t * list, const char * word);

/** Retourne le mot situé à la position 'index'. */
const char * list_at(list_t * list, unsigned int index);

#endif /* __LIST_H__ */
```

Exercice 1 (suite)

- Gestion de liste chaînée triée



Exercice 1 (suite)

- Gestion de liste chaînée triée

```
#include <stdio.h>
#include "list.h"

int main(int argc, char * argv)
{
  list_t my_list;
  unsigned int i;

  list_init(&my_list);
  list_add(&my_list, "Mot1");
  list_add(&my_list, "Mot2");
  list_add(&my_list, "AutreMot");
  for (i= 0; i < list_size(&my_list); i++) {
    printf("%s\n", list_get(&my_list, i));
  }
  list_free(&my_list);

  return EXIT_SUCCESS;
}
```

Exercice 1 (suite)

- Gestion de liste chaînée triée

Etant donné un fichier texte contenant une liste de mots

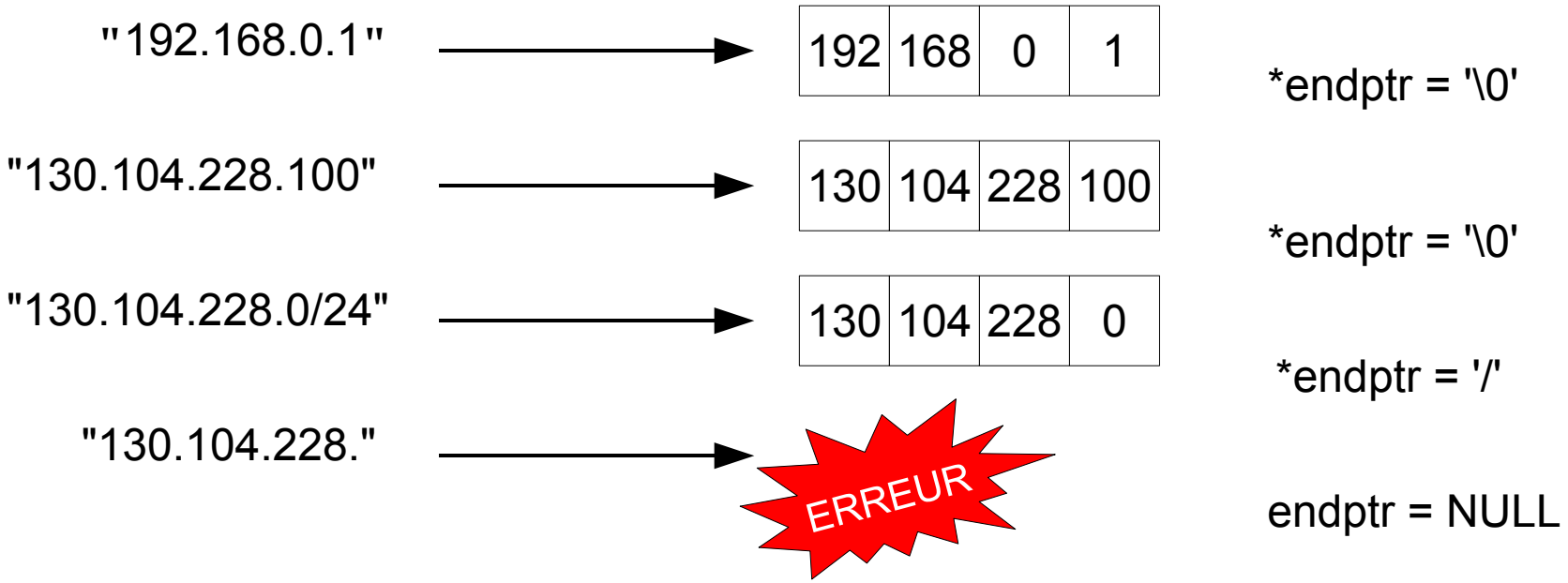
```
Transistor
Diode
Resistance
```

Lire le fichier et en charger le contenu dans une liste chaînée triée.
Afficher le contenu de la liste à l'écran.

```
toto@intel01$ ./prog mots.txt
Diode
Resistance
Transistor
toto@intel01$
```

Exercice 2

- Conversion d'une adresse IP




```
#define IP_ADDR_LEN 4
struct ip_addr_t {
  unsigned char addr[IP_ADDR_LEN];
};
```

Exercice 2

- Conversion d'une adresse IP

```
/** Convertit une adresse IP au format "A.B.C.D" en
 *  une adresse ip_addr_t.
 *  \param str_addr pointe vers une chaîne de caractère
 *         valide (terminée par '\0').
 *  \param addr pointe vers une zone de mémoire valide de
 *         taille sizeof(ip_addr_t) dans laquelle sera
 *         stocké le résultat de la conversion.
 *  \param endptr pointe vers une zone de mémoire valide
 *         de taille sizeof(char *) dans laquelle sera stockée
 *         l'adresse du premier caractère suivant l'adresse IP
 *         ou NULL si aucune conversion n'a pu être effectuée.
 * \return 0 en cas de succès, < 0 en cas d'erreur.
 */
int ip_str2addr(char * str_addr, struct ip_addr_t * addr,
                char ** endptr);
```

Note: la fonction `strtol()` pourrait vous être utile.

FIN

Questions ?