LEPL1503/LSINC1503 - Cours 1

O. Bonaventure, B. Legat, M. Baerts

☐ Full Width Mode ☐ Present Mode

Table of Contents

Pointeurs

Comment échanger deux variables entières ?

```c
#include <stdio.h>

void swap(int a, int b)
{
    int c = a;
    a = b;
    b = c;
}

int main()
{
    int x = 1;
    int y = 2;
    swap(x, y);
    printf("%d %d\n", x, y);
}
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs en C

```c
int x = 123;
int *ptr;
ptr = &x;
```

Echange du contenu de variables

C (C17 + GNU extensions)

```
 1  #include <stdio.h>
 2
 3  void swap(int a, int b)
 4  {
 5      int c = a;
 6      a = b;
 7      b = c;
 8  }
 9
10  int main()
11  {
12      int x = 1;
13      int y = 2;
14      swap(x, y);
15      printf("%d %d\n", x, y);
16  }
```

Edit Code & Get AI Help

line that just executed
next line to execute

< Prev Next >

Step 1 of 11

Visualized with pythontutor.com

Print output (drag lower right corner to resize)

Stack Heap

main
x int ?
y int ?

Note: ? refers to an uninitialized value

C/C++ details: none [default view]

Considérons le code ci-dessous

```c
#include <stdio.h>

int main()
{
    int x[] = {10, 20, 30, 40, 50, 60, 70, 80};
    int y[] = {2, 4, 6, 8};
    int *x_ptr = x;
    int *y_ptr = &(y[0]);
}
```

- `x_ptr` est l'adresse du premier élément du tableau `x`
- `y_ptr` est l'adresse du premier élément du tableau `y`

Arithmétique des pointeurs

- Il est possible de réaliser des calculs sur les pointeurs
- Si `ptr` pointe vers une zone de type donné en mémoire, alors `ptr+1` pointe vers la zone suivante de **ce type** en mémoire.

Pourquoi pas juste le byte suivant ? Par exemple, expliquez l'output suivant:

```c
#include <stdio.h>
int main()
{
    int x[] = {2026, 6, 2};
    char *y = (char *)x;
    printf("%lu %d %d\n", sizeof(int), x[0], x[1]);
    printf("%lu %d %d\n", sizeof(char), y[0], y[1]);
}
```

```
4 2026 6
1 -22 7
```

Arithmetique des pointeurs

L'entier 2026 est représenté sur 32 bits: 4 bytes de 8 bits chacun

```
"00000000000000000000011111101010"
```

```
1 bitstring(Cint(2026))
```

Un `char` ne représente qu'en seul byte, donc `y[0]` voit les bytes suivant:

```
"11101010"
```

```
1 bitstring(Cint(2026))[end-7:end]
```

Comme on utilise `%d`, c'est interprété comme un entier signé, les 7 premiers bits représentent 106

```
106
```

```
1 Cint(0b1101010)
```

Le bit de signe signifie qu'il faut soustraire 128, on arrive à `-22`

```
-22
```

```
1 Cint(0b1101010) - 128
```

Le byte suivant représente 7

```
"00000111"
```

```
0x07
```

```
1 0b111
```

Exécution du code

Questions

```c
#include <stdio.h>

int main()
{
    int x[] = {10, 20, 30, 40, 50, 60, 70, 80};
    int y[] = {2, 4, 6, 8};
    int *x_ptr = x;
    int *y_ptr = &(y[0]);
}
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs vers des entiers

```
int tab[] = {2, 4, 8, 16};
printf("%d \n", *(tab + 1));
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs vers des entiers

```
int tab[] = {2, 4, 8, 16};
printf("%d \n", *(tab) + 2);
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs vers des entiers

```
int tab[] = {2, 4, 8, 16};
printf("%d \n", *tab + 3);
```

wooclap

https://app.wooclap.com/EPL1503

Prototypes de fonctions

Lequel de ces prototypes est possible pour une fonction qui retourne le maximum d'un vecteur de réels ?

```
double max1(double v);
double max2(double *v);
double max3(double *v, int n);
double *max4(double *v, int n);
```

wooclap

https://app.wooclap.com/EPL1503

Qu'affiche ce code ?

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int x[] = {10, 20, 30};
    int *ptr = x;
    for (int i = 0; i < 4; i++)
    {
        sum += *(ptr);
        ptr++;
    }
    printf("Somme: %d\n", sum);
    return 0;
}
```

Somme d'entiers

C (C17 + GNU extensions)

```
 1  #include <stdio.h>
 2
 3  int main()
 4  {
 5      int sum = 0;
 6      int x[] = {10, 20, 30};
 7      int *ptr = x;
 8      for (int i = 0; i < 4; i++)
 9      {
10          sum += *(ptr);
11          ptr++;
12      }
13      printf("Somme: %d\n", sum);
14      return 0;
15  }
```

Edit Code & Get AI Help

line that just executed
next line to execute

< Prev Next >

Step 1 of 19

Visualized with pythontutor.com

Print output (drag lower right corner to resize)

Stack Heap

main

sum — int ?

x — array [0] [1] [2] — int ? | int ? | int ?

ptr — pointer to int ?

Note: ? refers to an uninitialized value

C/C++ details: none [default view]

Chaînes de caractères en C

Chaînes de caractères en java

Chaines de caractère en C

wooclap

https://app.wooclap.com/EPL1503

printf

```
Process(`man 3 printf`, ProcessExited(0))
```

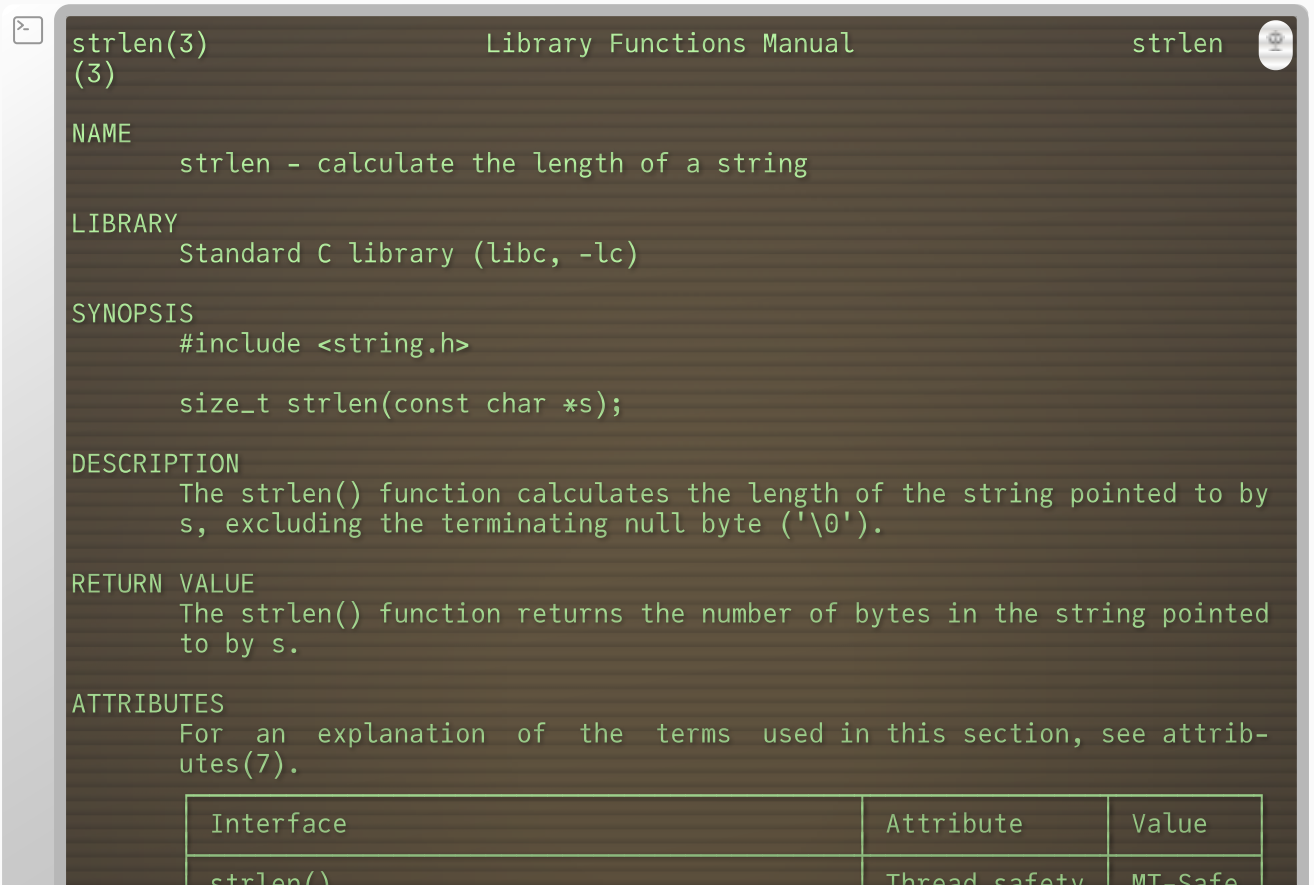
```
1 run(`man 3 printf`)
```

```
printf(3)                  Library Functions Manual                  printf
(3)

NAME
       printf,  fprintf,  dprintf,  sprintf,  snprintf, vprintf, vfprintf, vd-
       printf, vsprintf, vsnprintf - formatted output conversion

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <stdio.h>

       int printf(const char *restrict format, ...);
       int fprintf(FILE *restrict stream,
                   const char *restrict format, ...);
       int dprintf(int fd,
                   const char *restrict format, ...);
       int sprintf(char *restrict str,
                   const char *restrict format, ...);
       int snprintf(char str[restrict .size], size_t size,
                   const char *restrict format, ...);

       int vprintf(const char *restrict format, va_list ap);
       int vfprintf(FILE *restrict stream,
                   const char *restrict format, va_list ap);
       int vdprintf(int fd,
                   const char *restrict format, va_list ap);
       int vsprintf(char *restrict str
```

Comment calculer la longueur d'une chaîne de caractères en C ?

```
Process(`man 3 strlen`, ProcessExited(0))
```

```
1 run(`man 3 strlen`)
```



```
strlen(3)                  Library Functions Manual                  strlen
(3)

NAME
       strlen - calculate the length of a string

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <string.h>

       size_t strlen(const char *s);

DESCRIPTION
       The strlen() function calculates the length of the string pointed to by
       s, excluding the terminating null byte ('\0').

RETURN VALUE
       The strlen() function returns the number of bytes in the string pointed
       to by s.

ATTRIBUTES
       For  an  explanation  of  the  terms  used in this section, see attrib-
       utes(7).

       | Interface | Attribute | Value |
       | strlen()  | Thread safety | MT-Safe |
```

Comment écrire cette fonction en C ?

Même exemple avec des pointeurs

Qu'affiche

Pointeurs et chaînes de caractères

Qu'affiche ?

```
char *string = "abcdef";
printf("%s\n", string + 2);
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs et chaînes de caractères

Qu'affiche ?

```
char *string = "abcdef";
printf("%c\n", *(string + 3));
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs et chaînes de caractères

```
char string[] = "abcdef";
*(string + 1) = 'X';
printf("%s\n", string);
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs et chaînes de caractères

```
char string[] = "abcdef";
*(string + 1) = 'X';
printf("%s\n", string);
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs et chaînes de caractères

```c
char *string = "abcdef";
string++;
printf("%s\n", string);
```

wooclap

https://app.wooclap.com/EPL1503

Pointeurs et chaînes de caractères

```c
char string[] = "abcdef";
*(string + strlen(string)) = 'G';
printf("%s\n", string);
```

wooclap

https://app.wooclap.com/EPL1503

Beware!

Le `\0` a été supprimé, printf affichera cette chaîne uniquement si il y a un `\0` en mémoire à l'adresse qui suit celle de `G`

Pointeurs et chaînes de caractères

```c
char string[] = "abcdef";
*(string + strlen(string) + 1) = 'Z';
```

```c
printf("%s\n", string);
```

wooclap

https://app.wooclap.com/EPL1503

Beware!

Vous avez écrit en mémoire à une adresse en dehors de la chaine de caractère et avez potentiellement modifié la valeur d'une autre variable ou pire...

Pointeurs et chaînes de caractères

Lesquelles de ces déclarations sont valides ?

```c
char *string1 = "abcdef";
char string2[] = "abcdef";
char string3 = "ab";
char *string4 = 'A';
char string5 = 'B';
char *string6 = "C";
```

Organisation des processus en mémoire

Mémoire

Error message from ArgTools

```
RequestError: Connection timed out after 30010 milliseconds while requesting
https://lepl1503.info.ucl.ac.be/syllabus/theorie/_images/figures-001-c.png
```

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function(easy::Downloads.Curl.Easy) ...show types...
   from *Downloads.jl:479*
2. **with_handle**(f::Downloads.var"#26#27"{…}, handle::Downloads.Curl.Easy) ...show types...
   from *Curl.jl:105*
3. anonymous function
   from *Downloads.jl:390*
4. **arg_write**(f::Downloads.var"#24#25"{…}, arg::IOStream) ...show types...
   from *ArgTools.jl:134*
5. anonymous function
   from *Downloads.jl:389*
6. arg_read
   from *ArgTools.jl:76*
7. **#request#9**(url::String; input::Nothing, output::IOStream, method::Nothing, headers::Vector{…}, timeout::Float64, progress::Nothing, verbose::Bool, debug::Nothing, throw::Bool, downloader::Nothing, interrupt::Nothing) ...show types...
   from *Downloads.jl:388*
8. request

from *Downloads.jl:355*

9. anonymous function(output::IOStream) ...show types...
   from *Downloads.jl:273*
10. **#open#330**(f::Downloads.var"#7#8"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
    from *io.jl:410*
11. open_nolock
    from *ArgTools.jl:35*
12. **arg_write**(f::Function, arg::String)
    from **ArgTools** → *ArgTools.jl:103*
13. #download#5
    from *Downloads.jl:272*
14. download
    from *Downloads.jl:252*
15. **#RobustLocalResource#30**(url::String, path::String, html_attributes::Pair{…}; cache::Bool) ...show types...
    from *robustlocalresource.jl:20*
16. RobustLocalResource
    from *robustlocalresource.jl:7*
17. **#save_image#3**
    from Other cell: *line 24*

```
22 function save_image(url::URL, html_attributes...; name = split(url.url, '/')[end], kws...)
23     path = joinpath("cache", name)
24     return PlutoTeachingTools.RobustLocalResource(url.url, path, html_attributes...), path
25 end
26
```

cell preview

18. Show more...

```
1 img("https://lepl1503.info.ucl.ac.be/syllabus/theorie/_images/figures-001-c.png", :width => 200)
```

👀 *Reading hidden code*

Segment text

```
objdump -S a.out

a.out:     file format elf32-i386

Disassembly of section .init:

08048290 <_init>:
 8048290:   55                      push   %ebp
 8048291:   89 e5                   mov    %esp,%ebp
 8048293:   53                      push   %ebx
 8048294:   83 ec 04                sub    $0x4,%esp
 8048297:   e8 00 00 00 00          call   804829c <_init+0xc>
 804829c:   5b                      pop    %ebx
 804829d:   81 c3 88 13 00 00       add    $0x1388,%ebx
 80482a3:   8b 93 fc ff ff ff       mov    -0x4(%ebx),%edx
 80482a9:   85 d2                   test   %edx,%edx
 80482ab:   74 05                   je     80482b2 <_init+0x22>

Text
```

malloc(3)

```
Process(`man 3 malloc`, ProcessExited(0))
```

```
1 run(`man 3 malloc`)
```

```
malloc(3)                  Library Functions Manual                  malloc
(3)

NAME
       malloc, free, calloc, realloc, reallocarray - allocate and free dynamic
       memory

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <stdlib.h>

       void *malloc(size_t size);
       void free(void *_Nullable ptr);
       void *calloc(size_t nmemb, size_t size);
       void *realloc(void *_Nullable ptr, size_t size);
       void *reallocarray(void *_Nullable ptr, size_t nmemb, size_t size);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       reallocarray():
           Since glibc 2.29:
               _DEFAULT_SOURCE
           glibc 2.28 and earlier:
               _GNU_SOURCE

DESCRIPTION
   malloc()
```

Points d'attention

La fonction `malloc` peut échouer à allouer de la mémoire

- Vous devez **toujours** tester la valeur de retour de `malloc`
- Si malloc a retourné `NULL`, votre code doit réagir correctement

Points d'attention

La fonction `malloc` *réserve* de la mémoire **mais ne l'initialise pas**

- En bon programmeur, vous devez évidemment initialiser la mémoire **avant** de l'utiliser
- `calloc` initialise la mémoire pour vous à zéro

Points d'attention

- Le pointeur retourné par `malloc` a 2 rôles
  - Il vous permet d'accéder à la zone mémoire que vous avez alloué
    - Vous devrez le manipuler dans le code
  - C'est l'identifiant utilisé par `malloc` / `free` qui sera nécessaire pour libérer la mémoire ultérieurement
    - Gardez une copie de cette adresse dans votre code

Points d'attention

- Toute zone mémoire allouée par `malloc` doit être libérée explicitement par `free` avant la fin de l'exécution du programme
  - Il n'y a pas de garbage collector en C
  - Si vous oubliez de libérer de la mémoire que vous avez alloué, vous causez un memory leak

Pile

Error message from ArgTools

```
RequestError: Connection timed out after 30032 milliseconds while requesting
https://lepl1503.info.ucl.ac.be/syllabus/theorie/_images/figures-001-c.png
```

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function(easy::Downloads.Curl.Easy) ...show types...
   from Downloads.jl:479

2. **with_handle**(f::Downloads.var"#26#27"{…}, handle::Downloads.Curl.Easy) ...show types...
   from *Curl.jl:105*
3. anonymous function
   from *Downloads.jl:390*
4. **arg_write**(f::Downloads.var"#24#25"{…}, arg::IOStream) ...show types...
   from *ArgTools.jl:134*
5. anonymous function
   from *Downloads.jl:389*
6. arg_read
   from *ArgTools.jl:76*
7. **#request#9**(url::String; input::Nothing, output::IOStream, method::Nothing, headers::Vector{…}, timeout::Float64, progress::Nothing, verbose::Bool, debug::Nothing, throw::Bool, downloader::Nothing, interrupt::Nothing) ...show types...
   from *Downloads.jl:388*
8. request
   from *Downloads.jl:355*
9. anonymous function(output::IOStream) ...show types...
   from *Downloads.jl:273*
10. **#open#330**(f::Downloads.var"#7#8"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
    from *io.jl:410*
11. open_nolock
    from *ArgTools.jl:35*
12. **arg_write**(f::Function, arg::String)
    from **ArgTools** → *ArgTools.jl:103*
13. #download#5
    from *Downloads.jl:272*
14. download
    from *Downloads.jl:252*
15. **#RobustLocalResource#30**(url::String, path::String, html_attributes::Pair{…}; cache::Bool) ...show types...
    from *robustlocalresource.jl:20*
16. RobustLocalResource
    from *robustlocalresource.jl:7*

17. **#save_image#3**

from Other cell: *line 24*

```
function save_image(url::URL, html_attributes...; name = split(url.url, '/')[end], kws...)
    path = joinpath("cache", name)
    return PlutoTeachingTools.RobustLocalResource(url.url, path, html_attributes...), path
end
```

cell preview

18. Show more...

```
img("https://lepl1503.info.ucl.ac.be/syllabus/theorie/_images/figures-001-c.png", :width => 200)
```

Reading hidden code

Informations sur la pile/stack

- paramètres passés aux fonctions
  - optimisation : certains paramètres sont passés dans des registres du CPU
- variables locales aux fonctions
- adresse de retour des fonctions

Exemples

Y-a-t-il une différence de performance entre ces 2 fonctions ?

```c
#define MILLION 1000000

struct large_t
{
    int i;
    char str[MILLION];
};

int sum(struct large_t s1, struct large_t s2)
{
    return (s1.i + s2.i);
}

int sumptr(struct large_t *s1, struct large_t *s2)
{
    return (s1->i + s2->i);
}
```

Localisation en mémoire

Dans le code ci-dessous, dans quelle zone de la mémoire se trouve le contenu du tableau `tab[]`

```c
int main(int argc, char **argv) {
   int tab[] = {2, 4, 8, 16};
```

- segments text
- données
- heap
- stack

Localisation en mémoire

Dans le code ci-dessous, dans quelle zone de la mémoire se trouve **le pointeur** `tab[]`

```c
int main(int argc, char **argv) {
   int *tab = (int *)malloc(4 * sizeof(int));
```

- segments text
- données
- heap

- stack

Localisation en mémoire

Dans le code ci-dessous, dans quelle zone de la mémoire se trouve le contenu du tableau `tab`

```c
int main(int argc, char **argv) {
   int *tab = (int *)malloc(4 * sizeof(int));
```

- segments text
- données
- heap
- stack

Accès après free

```c
int *tab2 = (int *)malloc(4 * sizeof(int));
*tab2 = 2;
*(tab2 + 1) = 4;
free(tab2);
printf("%d\n", *(tab2 + 1));
```

▶ **Qu'affiche ce code ?**

Free depuis un pointeur dans le tableau

```c
int *tab2 = (int *)malloc(4 * sizeof(int));
*tab2 = 2;
tab2++;
*tab2 = 4;
free(tab2);
printf("%d\n", *(tab2));
```

▶ **Qu'affiche ce code ?**

The End

Utils

wooslide (generic function with 1 method)

woo =

wooclap

https://app.wooclap.com/EPL1503

```
woo = wooclap("EPL1503")
```

```
using PlutoUI, PlutoUI.ExperimentalLayout, HypertextLiteral, PlutoTeachingTools, SimpleClang
```

```
import HTTP, Clang_jll, MultilineStrings, InteractiveUtils
```

tutor (generic function with 1 method)

wooclap (generic function with 1 method)

```
img (generic function with 3 methods)

two_columns (generic function with 1 method)

three_columns (generic function with 1 method)

qa (generic function with 2 methods)
```