LEPL1503/LSINC1503 - Cours 2

O. Bonaventure, B. Legat, M. Baerts

☐ Full Width Mode ☐ Present Mode

Table of Contents

Variables, pointeurs et doubles pointeurs

Visualisation

Arguments de main

- `int argc` : Le nombre d'arguments (incluant le nom de l'exécutable)

- `char **argv` : liste des arguments initialisée par le système d'exploitation au lancement du programme

```c
#include <stdio.h>
int main(int argc, char **argv)
{
    for (int i = 0; i < argc; i++)
    {
        printf("arg[%d]: %s\n", i, argv[i]);
    }
}
```

```
Running : `/tmp/jl_2hchCd/bin ab cd ef`
```

```
arg[0]: /tmp/jl_2hchCd/bin
arg[1]: ab
arg[2]: cd
arg[3]: ef
```

Matrices en C

Comment construire une matrice ?

```c
float A[2][2] = {{1, 0}, {2, 3}};
A[1][1] = 4;
```

Une telle matrice sera stockée sur le **stack** (variable locale) ou dans la zone données (variable globale)

Matrices sur le heap

Comment mettre une matrice 2x2 sur le heap ?

```
float *matrice1 = (float *)malloc(4 * sizeof(float));
matrice1[1][1] = 4;
```

```
/tmp/jl_GbHfnW/main.c:5:14: error: subscripted value is not an array, point
er, or vector
    5 |     matrice1[1][1] = 4;
      |     ~~~~~~~~~~~^~
1 error generated.
```

Première approche, un malloc par ligne

C (C17 + GNU extensions)

```
1  #include <stdlib.h>
2
3  int main(int argc, char **argv) {
4    float **matrice2 = (float **)ma
5    float *ligne0 = (float *)malloc
6    float *ligne1 = (float *)malloc
7    matrice2[0] = ligne0;
8    matrice2[1] = ligne1;
9    matrice2[1][1] = 4;
10 }
```

Edit Code & Get AI Help

line that just executed
next line to execute

< Prev | Next >

Step 1 of 7

Visualized with pythontutor.com

Stack | Heap

main

argc	int ?
argv	pointer to char* ?
matrice2	pointer to float* ?
ligne0	pointer to float ?
ligne1	pointer to float ?

Note: ? refers to an uninitialized value

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

Deuxième approche, un malloc pour toutes les valeurs

Initialisation d'une valeur

Comment initialiser la première valeur ?

```c
float **matrice = (float **)malloc(2 * sizeof(float *));
float *ligne0 = (float *)malloc(2 * sizeof(float));
float *ligne1 = (float *)malloc(2 * sizeof(float));
matrice[0] = ligne0;
matrice[1] = ligne1;
matrice[0][0] = 1; // 1
*(matrice[0]) = 1; // 2
*(matrice) = 1;    // 3
```

wooclap

https://app.wooclap.com/JAPRXX

Comment faire matrice[0][1] = 0 ?

```c
float **matrice = (float **)malloc(2 * sizeof(float *));
float *ligne0 = (float *)malloc(2 * sizeof(float));
float *ligne1 = (float *)malloc(2 * sizeof(float));
matrice[0] = ligne0;
matrice[1] = ligne1;
*(*matrice + 1) = 0;   // 1
*(matrice++) = 0;      // 2
*(matrice[0] + 1) = 0; // 3
```

wooclap

https://app.wooclap.com/JAPRXX

Comment faire matrice[1][0] = 2 ?

```
float **matrice = (float **)malloc(2 * sizeof(float *));
float *ligne0 = (float *)malloc(2 * sizeof(float));
float *ligne1 = (float *)malloc(2 * sizeof(float));
matrice[0] = ligne0;
matrice[1] = ligne1;
*(*(++matrice)) = 2;   // 1
*(*(matrice + 1)) = 2; // 2
*(*matrice + 1) = 2;   // 3
```

wooclap

https://app.wooclap.com/JAPRXX

Libération de la mémoire

L'importance de l'ordre de désallocation

Si la matrice a été créée par

```
float **matrice = (float **)malloc(2 * sizeof(float *));
float *ligne0 = (float *)malloc(2 * sizeof(float));
float *ligne1 = (float *)malloc(2 * sizeof(float));
matrice[0] = ligne0;
matrice[1] = ligne1;
```

▶ **Comment libérer la mémoire avec `free` ?**

wooclap

https://app.wooclap.com/JAPRXX

Toujours free dans le sens inverse de malloc

C (C17 + GNU extensions)

```
 1 #include <stdlib.h>
 2
 3 int main(int argc, char **argv) {
 4   float **matrice = (float **)mal
 5   matrice[0] = (float *)malloc(2
 6   matrice[1] = (float *)malloc(2
 7   free(matrice);
 8   free(matrice[0]);
 9   free(matrice[1]);
10 }
```

Edit Code & Get AI Help

line that just executed
next line to execute

< Prev Next >

Done running (6 steps)

Stack Heap

main

argc	int 1
argv	pointer to char* 💀
matrice	pointer to float* 💀

Note: 💀 means a pointer points to memory that is either misaligned with data boundaries. 💀 locations are *approxi* not match the pointer's real address. Select 'byte-level vie to see more details:

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

ERROR: Invalid read of size 8
(Stopped running after the first err r. Please fix your code.)

see UNSUPPORTED FEATURESundefined

Un vecteur "Java" en C

Première implémentation

```c
struct vector_t
{
    int size;
    float *v;
};
```

```c
struct vector_t * init(int size, float val) {
```

```c
float get(struct vector_t *t, int i) {
```

```c
void set(struct vector_t *t, int i, float val) {
```

```c
void destroy(struct vector_t *vect) {
```

wooclap

https://app.wooclap.com/JAPRXX

Une autre implémentation ...

Toutes les fonctions doivent vérifier leurs arguments et retourner

- -1 en cas d'erreur

- 0 en cas de succès

```
struct vector_t {
  int length; // nombre d'élements
  float *tab; // tableau avec les réels
};
```

Initialisation

```
struct vector_t {
  int length; // nombre d'élements
  float *tab; // tableau avec les réels
};
```

Quels arguments ?

```
/*
 * @pre
 * alloue la mémoire pour un vecteur de size éléments
 * initialisés à la valeur val
 * @return -1 si erreur, 0 sinon
 */

int init(int size, float val,
         struct vector_t v,   // 1
         struct vector_t **v, // 2
         struct vector_t *v,  // 3
)
```

wooclap

https://app.wooclap.com/JAPRXX

Allocation de la mémoire

```
struct vector_t {
  int length; // nombre d'élements
  float *tab; // tableau avec les réels
};
```

```c
int init(int size, float val, struct vector_t **v)
{
    if ((size < 0) || v == NULL)
        return -1;
    *v = (struct vector_t *)malloc(sizeof(struct vector_t));
    if (*v == NULL)
    {
        return -1;
    }
    (*v)->tab = (float *)malloc(size * sizeof(float));
    if ((*v)->tab == NULL)
    {
        return -1;
    }
    (*v)->length = size;
    for (int i = 0; i < size; i++)
    {
```

▶ **Comment assigner la valeur `val` au `i`ième élément ?**

```c
  }
  return 0;
}
```

wooclap

https://app.wooclap.com/JAPRXX

Visualization de init

Récupération d'un élément

```
struct vector_t {
  int length; // nombre d'élements
  float *tab; // tableau avec les réels
};
```

```c
/*
 * @pre
 * @post retourne le ième élément du tableau dans val
 *       -1 en cas d'erreur, 0 sinon
 */

int get(struct vector_t *v, int i,
        float val   // 1
        float **val // 2
        float *val  // 3
) {
```

▶ **Comment implémenter `get` ?**

wooclap

https://app.wooclap.com/JAPRXX

Modification d'un élément

```c
/*
 * @pre
 * @post place val comme ième élément
 *             du tableau
 *       -1 en cas d'erreur, 0 sinon
 */

int set(struct vector_t *v, int i,
struct vector_t  v     // 1
struct vector_t ** v   // 2
struct vector_t * v    // 3
) {
```

▶ **Comment implémenter `set` ?**

wooclap

https://app.wooclap.com/JAPRXX

Libération de la mémoire

Comment libérer la mémoire quand le vecteur est devenu inutile ?

```
/*
 * libère la mémoire utilisée pour le vecteur
 * -1 en cas d'erreur, 0 sinon
 */
```

```
int destroy(struct vector_t *v) {
```

wooclap

https://app.wooclap.com/JAPRXX

Pointeur de fonction

`mapreduce` donne un exemple de fonction d'ordre supérieur en C

Retourner un tableau

Heap ou stack ?

Laquelle de ces deux fonctions présente une façon correcte de retourner un tableau ?

```c
int *stack()
{
    int v[] = {1, 2, 3};
    return v;
}

int *heap()
{
    int *v = (int *)malloc(3 * sizeof(int));
    v[0] = 1;
    v[1] = 2;
    v[2] = 3;
    return v;
}
```

wooclap

https://app.wooclap.com/JAPRXX

Retourner un tableau : visualization

C (C17 + GNU extensions)

```
 9  int *heap()
10  {
11      int *v = (int *)malloc(3 * si
12      v[0] = 1;
13      v[1] = 2;
14      v[2] = 3;
15      return v;
16  }
17
18  int main()
19  {
20      int *v1 = heap();
21      printf("%d %d %d\\n", v1[0],
22      free(v1); // don't forget!
23      int *v2 = stack();
24      printf("%d %d %d\\n", v2[0],
25      return 0;
26  }
```

Edit Code & Get AI Help

line that just executed
next line to execute

< Prev Next >

Done running (15 steps)

Print output (drag lower right corner to resize)

```
1 2 3\n
```

Stack Heap

main

v1 — pointer to int 💀

v2 — pointer to int NULL (0x0)

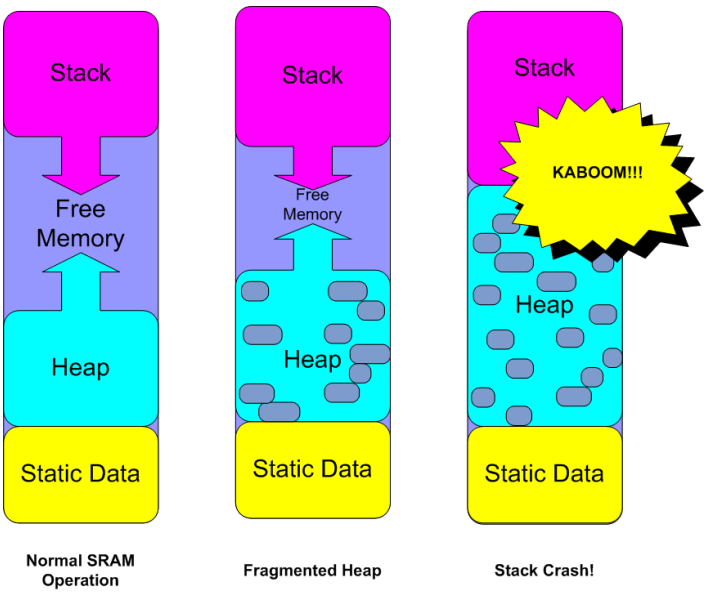
Note: 💀 means a pointer points to memory that is either misaligned with data boundaries. 💀 locations are *approxi* not match the pointer's real address. Select 'byte-level vie to see more details:

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

ERROR: Invalid read of size 4
(Stopped running after the first err r. Please fix your code.)

Structure différente du stack et heap

- Stack : Pile de mémoire locale des functions, structure connue à la **compilation**
- Heap : Alloué et désalloué avec des tailles connues à l'**exécution**. L'OS fait de son mieux pour organiser cette mémoire et réassigner les fragments formés suite aux désallocations



Source

The End

```
section (generic function with 1 method)
```

```
section(s) = md"# $s"
```

```
frametitle (generic function with 1 method)
```

```
frametitle(s) = md"## $s"
```

```
using PlutoUI, PlutoUI.ExperimentalLayout, PlutoTeachingTools, HypertextLiteral: @htl
```

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

```
compile_lib (generic function with 2 methods)
```

```
begin
abstract type Code end

struct CCode <: Code
    code::String
end

macro c_str(s)
    return :($CCode($(esc(s))))
end

struct CppCode <: Code
    code::String
end

macro cpp_str(s)
    return :($CppCode($(esc(s))))
end

struct CLCode <: Code
    code::String
end

macro cl_str(s)
    return :($CLCode($(esc(s))))
end

struct JavaCode <: Code
    code::String
end

macro java_str(s)
    return :($JavaCode($(esc(s))))
end

source_extension(::CCode) = "c"
source_extension(::CppCode) = "cpp"
source_extension(::CLCode) = "cl"
source_extension(::JavaCode) = "java"

compiler(::CCode, mpi::Bool) = mpi ? "mpicc" : "clang"
function compiler(::CppCode, mpi::Bool)
    @assert !mpi
    return "clang++"
end

inline_code(code::AbstractString, ext::String) = HTML("""<code
class="language-$ext">$code</code>""")
```

```
inline_code(code::Code) = inline_code(code.code, source_extension(code))

function md_code(code::AbstractString, ext::String)
    code = "````" * ext * '\n' * code
    if code[end] != '\n'
        code *= '\n'
    end
    return Markdown.parse(code * "````")
end
md_code(code::Code) = md_code(code.code, source_extension(code))
function Base.show(io::IO, m::MIME"text/html", code::Code)
    return show(io, m, md_code(code))
end

function compile(
    code::Code;
    lib,
    emit_llvm = false,
    cflags = ["-O3"],
    mpi::Bool = false,
    use_system::Bool = mpi || "-fopenmp" in cflags, # `-fopenmp` will not work with pure Clang_jll, it needs openmp installed as well
    verbose = 0,
)
    path = mktempdir()
    main_file = joinpath(path, "main." * source_extension(code))
    bin_file = joinpath(path, ifelse(emit_llvm, "main.llvm", ifelse(lib, "lib.so", "bin")))
    write(main_file, code.code)
    args = String[]
    if !use_system && code isa CppCode
        # `clang++` is not part of `Clang_jll`
        push!(args, "-x")
        push!(args, "c++")
    end
    append!(args, cflags)
    if lib
        push!(args, "-fPIC")
        push!(args, "-shared")
    end
    if emit_llvm
        push!(args, "-S")
        push!(args, "-emit-llvm")
    end
    push!(args, main_file)
    push!(args, "-o")
    push!(args, bin_file)
    try
        if use_system
            cmd = Cmd([compiler(code, mpi); args])
```

```
            if verbose >= 1
                @info("Compiling : $cmd")
            end
            run(cmd)
        else
            Clang_jll.clang() do exe
                cmd = Cmd([exe; args])
                if verbose >= 1
                    @info("Compiling : $cmd")
                end
                run(cmd)
            end
        end
    catch err
        if err isa ProcessFailedException
            return
        else
            rethrow(err)
        end
    end
    return bin_file
end

function emit_llvm(code; kws...)
    llvm = compile(code; lib = false, emit_llvm = true, kws...)
    InteractiveUtils.print_llvm(stdout, read(llvm, String))
    return code
end

function compile_lib(code::Code; kws...)
    return codesnippet(code), compile(code; lib = true, kws...)
end

function compile_and_run(code::Code; verbose = 0, args = String[], valgrind::Bool =
false, mpi::Bool = false, num_processes = nothing, show_run_command =
!isempty(args) || verbose >= 1, kws...)
    bin_file = compile(code; lib = false, mpi, verbose, kws...)
    if !isnothing(bin_file)
        cmd_vec = [bin_file; args]
        if valgrind
            cmd_vec = ["valgrind"; cmd_vec]
        end
        if mpi
            if !isnothing(num_processes)
                cmd_vec = [["-n", string(num_processes)]; cmd_vec]
            end
            cmd_vec = ["mpiexec"; cmd_vec]
        end
        cmd = Cmd(cmd_vec)
        if show_run_command
```

```
            @info("Running : $cmd") # `2:end-1` to remove the backsticks
        end
        try
            run(cmd)
        catch err
            @warn(string(typeof(err)))
        end
    end
    return codesnippet(code)
end

function wrap_in_main(content)
    code = content.code
    if code[end] == '\n'
        code = code[1:end-1]
    end
    return typeof(content)("""
#include <stdlib.h>

int main(int argc, char **argv) {
$(MultilineStrings.indent(code, 2))
}
""")
end

function wrap_compile_and_run(code; kws...)
    compile_and_run(wrap_in_main(code); kws...)
    return code
end

# TODO It would be nice if the user could select a dropdown or hover with the mouse and see the full code
function codesnippet(code::Code)
    lines = readlines(IOBuffer(code.code))
    i = findfirst(line -> contains(line, "codesnippet"), lines)
    if isnothing(i)
        return code
    end
    j = findlast(line -> contains(line, "codesnippet"), lines)
    return typeof(code)(join(lines[i+1:j-1], '\n'))
end

struct Example
    name::String
end

function code(example::Example)
    code = read(joinpath(dirname(dirname(dirname(@__DIR__))), "examples", example.name), String)
    ext = split(example.name, '.')[end]
```

```
    if ext == "c"
        return CCode(code)
    elseif ext == "cpp" || ext == "cc"
        return CppCode(code)
    elseif ext == "cl"
        return CLCode(code)
    else
        error("Unrecognized extension `$ext`.")
    end
end

function compile_and_run(example::Example; kws...)
    return compile_and_run(code(example); kws...)
end

function compile_lib(example::Example; kws...)
    return compile_lib(code(example); kws...)
end
```

```
tutor (generic function with 1 method)
```

```
# No need to go beyond `620` because pythontutor will just add an internal slider an not show full code
function tutor(code::Code; width = 800, height = min(620, 260 + 20 * countlines(IOBuffer(code.code))), frameborder = 0, curInstr = 0)
    esc_code = HTTP.escapeuri(code.code)
    src = "https://pythontutor.com/iframe-embed.html#code=$esc_code&cumulative=false&py=$(source_extension(code))&curInstr=$curInstr"
    return HTML("""<iframe width="$width" height="$height" frameborder="$frameborder" src="$src"></iframe>""")
end
```

```
wooclap (generic function with 1 method)
```

```
function wooclap(link)
    logo = HTML("""<img alt="Wooclap Logo" src="https://www.wooclap.com/images/wooclap-logo.svg">""")
    url = "https://app.wooclap.com/$link"
    a = HTML("""<a style="margin-left: 80px;" href="https://app.wooclap.com/$link"><tt>$url</tt></a>""")
    qr = img("https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=$url", :height => 50, name = "$link.png")
    return three_columns(logo, a, qr)
end
```

```
img (generic function with 3 methods)
```

```
begin
struct Path
    path::String
end

function imgpath(path::Path)
    file = path.path
    if !('.' in file)
        file = file * ".png"
    end
    return joinpath(joinpath(@__DIR__, "images", file))
end

function img(path::Path, args...; kws...)
    return PlutoUI.LocalResource(imgpath(path), args...)
end

struct URL
    url::String
end

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

function img(url::URL, args...; kws...)
    r, _ = save_image(url, args...; kws...)
    return @htl("<a href=$(url.url)>$r</a>")
end

function img(file::String, args...; kws...)
    if startswith(file, "http")
        img(URL(file), args...; kws...)
    else
        img(Path(file), args...; kws...)
    end
end
end
```

```
three_columns (generic function with 1 method)
```

```
three_columns(left, center, right) = hbox([
    left,
    Div(html" ", style = Dict("flex-grow" => "1")),
    center,
    Div(html" ", style = Dict("flex-grow" => "1")),
    right,
])
```

```
qa (generic function with 2 methods)
```

```
begin
function qa(question, answer)
    return @htl("<details><summary>$question</summary>$answer</details>")
end
function _inline_html(m::Markdown.Paragraph)
    return sprint(Markdown.htmlinline, m.content)
end
function qa(question::Markdown.MD, answer)
    # `html(question)` will create `<p>` if `question.content[]` is `Markdown.Paragraph`
    # This will print the question on a new line and we don't want that:
    h = HTML(_inline_html(question.content[]))
    return qa(h, answer)
end
end
```