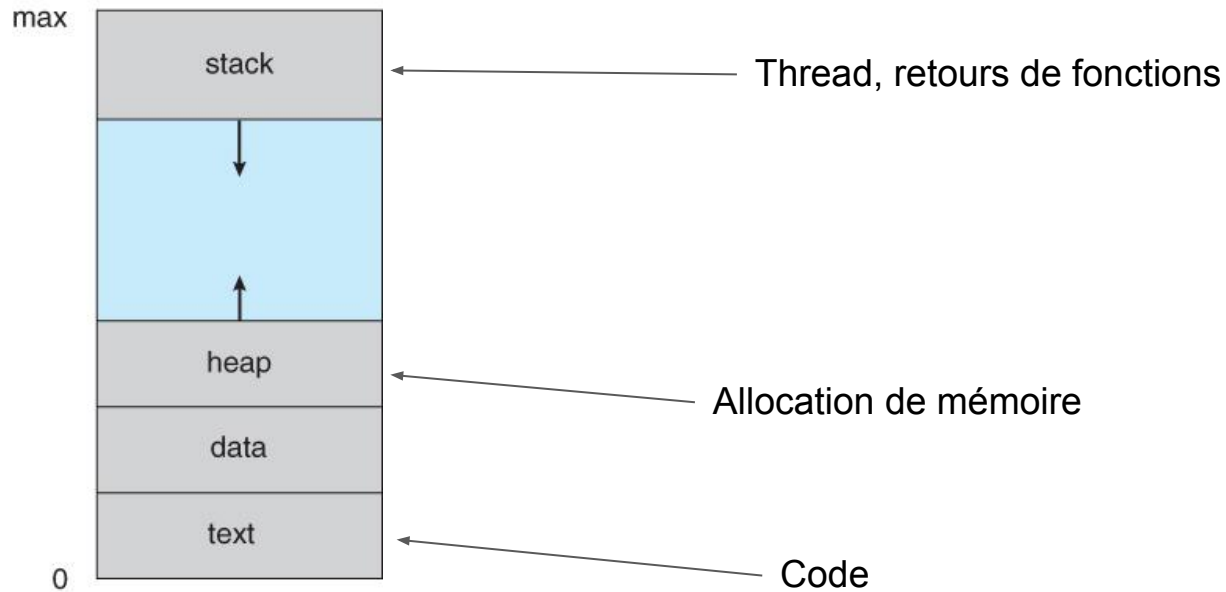


Gestion de la mémoire

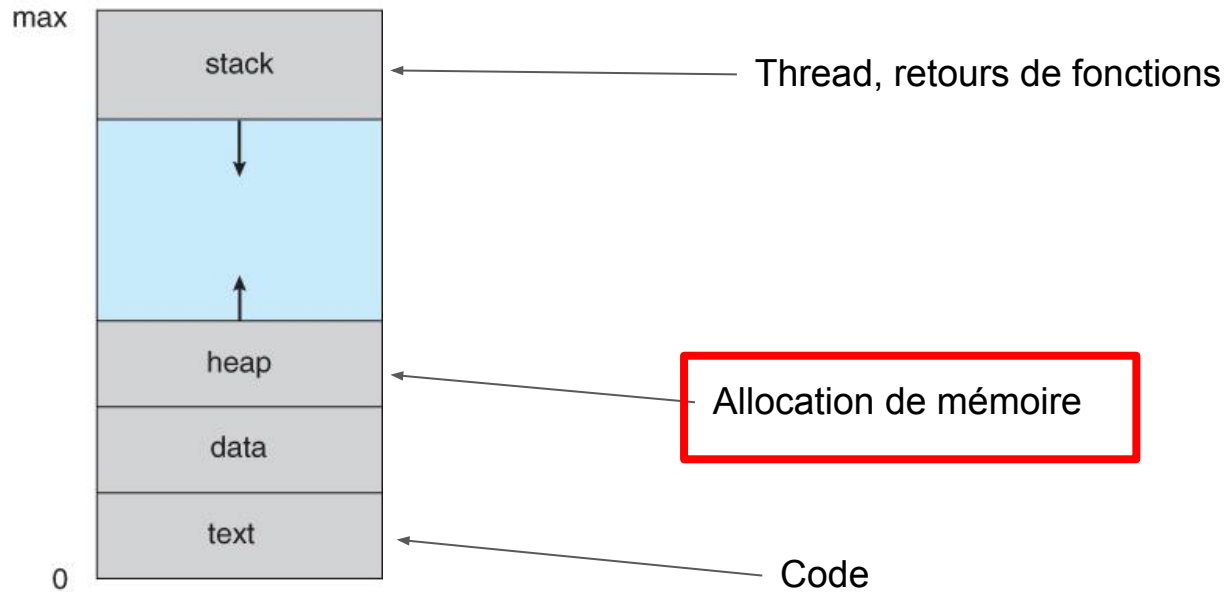
Langage C

Jérôme Fink

Mémoire d'un processus



Mémoire d'un processus



La mémoire en C

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..), NULL

int main(int argc, char* argv[]) {
    ● unsigned int counter = 42;
    unsigned int* pointer = NULL;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    pointer = &counter;
    (*pointer)++;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    return EXIT_SUCCESS;
}
```

42

0x7f

La mémoire en C

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..), NULL

int main(int argc, char* argv[]) {
    unsigned int counter = 42;
    unsigned int* pointer = NULL;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    pointer = &counter;
    (*pointer)++;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    return EXIT_SUCCESS;
}
```

42

0x7f

NULL

0x80

La mémoire en C

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..), NULL

int main(int argc, char* argv[]) {
    unsigned int counter = 42;
    unsigned int* pointer = NULL;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    ● pointer = &counter;
    (*pointer)++;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    return EXIT_SUCCESS;
}
```

42

0x7f

NULL

0x80

La mémoire en C

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..), NULL

int main(int argc, char* argv[]) {
    unsigned int counter = 42;
    unsigned int* pointer = NULL;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    ● pointer = &counter;
      (*pointer)++;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    return EXIT_SUCCESS;
}
```

42

0x7f

0x7f

0x80

La mémoire en C

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..), NULL

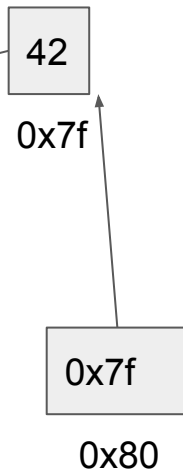
int main(int argc, char* argv[]) {
    unsigned int counter = 42;
    unsigned int* pointer = NULL;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    pointer = &counter;
    (*pointer)++;

    printf("Value of counter : %u\n", counter);
    printf("Value of pointer : %p\n", pointer);
    if(pointer != NULL)
        printf("Value at pointer : %u\n", *pointer);

    return EXIT_SUCCESS;
}
```



Opérateurs mémoires

Opérateur “*” : opérateur de déréférencement. Permet d’obtenir la valeur située à l’adresse contenue dans la variable.

Opérateur “&” : Opérateur de référencement. Permet d’avoir l’adresse d’une variable quelconque

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    ● unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP

4 octet



0x8f

Data

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

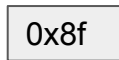
HEAP

4 octet



0x8f

Data



0x8f

0x1a
count

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

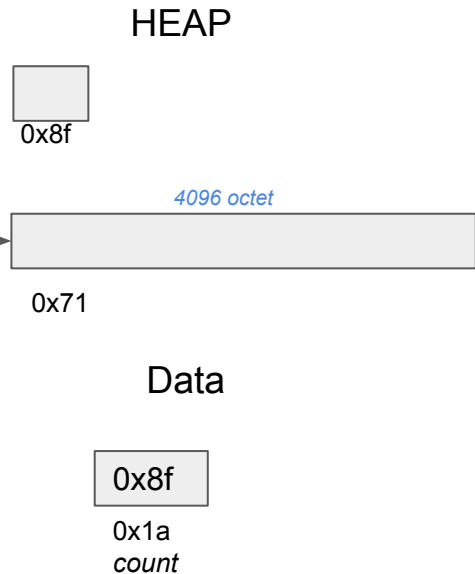
#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    ● unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

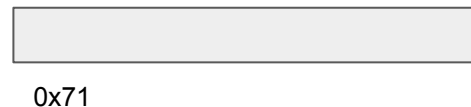
int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

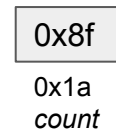
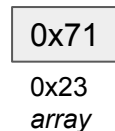
    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP



Data



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

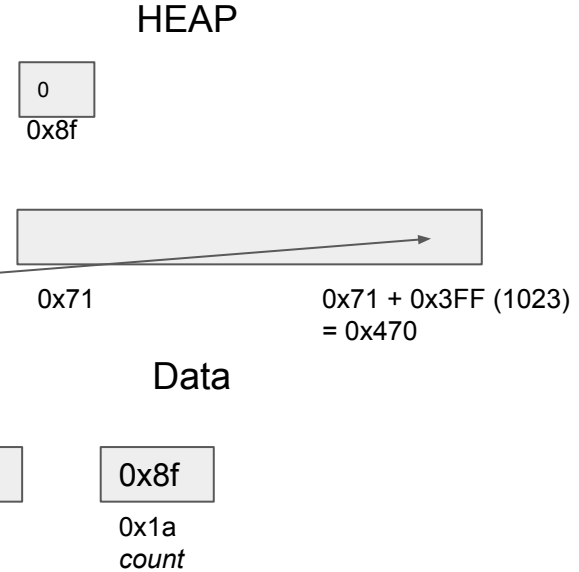
#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

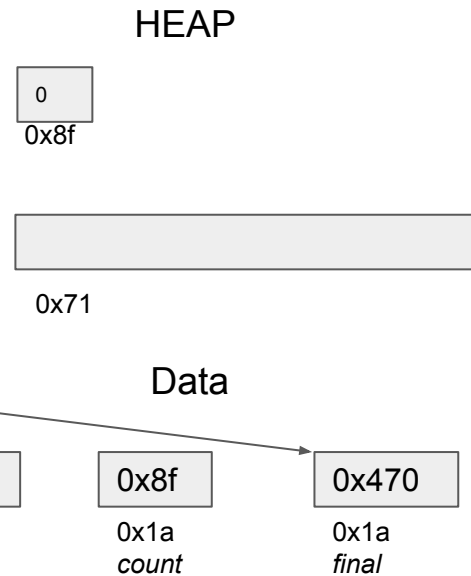
#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

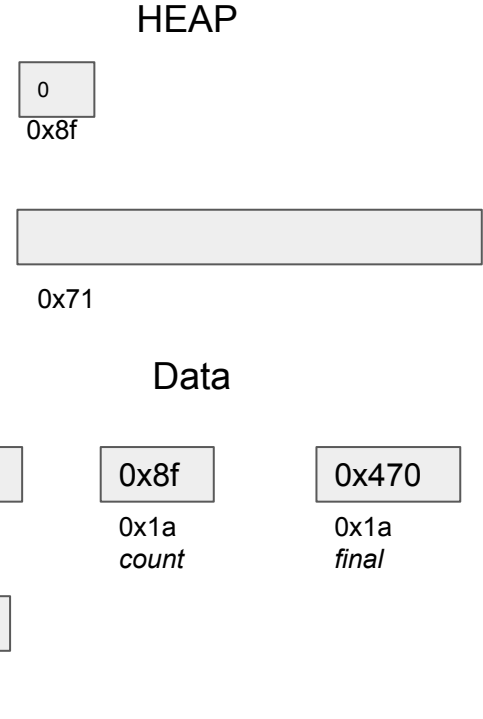
#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```



Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP

0
0x8f

0x71

Data

0x71
0x23
array

0x8f
0x1a
count

0x470
0x1a
final

0x470
0x2f
current

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP



0x71

Data

0x71

0x23
array

0x8f

0x1a
count

0x470

0x1a
final

0x470

0x2f
current

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP



0x71

Data

0x71

0x23
array

0x8f

0x1a
count

0x470

0x1a
final

0x470

0x2f
current

Allocation dynamique

```
#include <stdio.h> // printf(..)
#include <stdlib.h> // EXIT_SUCCESS, malloc(..), calloc(..)

#define SIZE 1024

int main(int argc, char* argv[]) {
    unsigned int* count = malloc(4);
    unsigned int* array = malloc( SIZE * sizeof(unsigned int) );

    *count = 0;
    unsigned int* final = &array[SIZE-1];
    for(unsigned int* current = array; current <= final ; current++) {
        if (*current == 0)
            (*count)++;
    }
    printf("Number of zeroes in array : %u\n", *count);

    free(count);
    free(array);

    return EXIT_SUCCESS;
}
```

HEAP

Data

0x71

0x23
array

0x470

0x2f
current

0x8f

0x1a
count

0x470

0x1a
final

