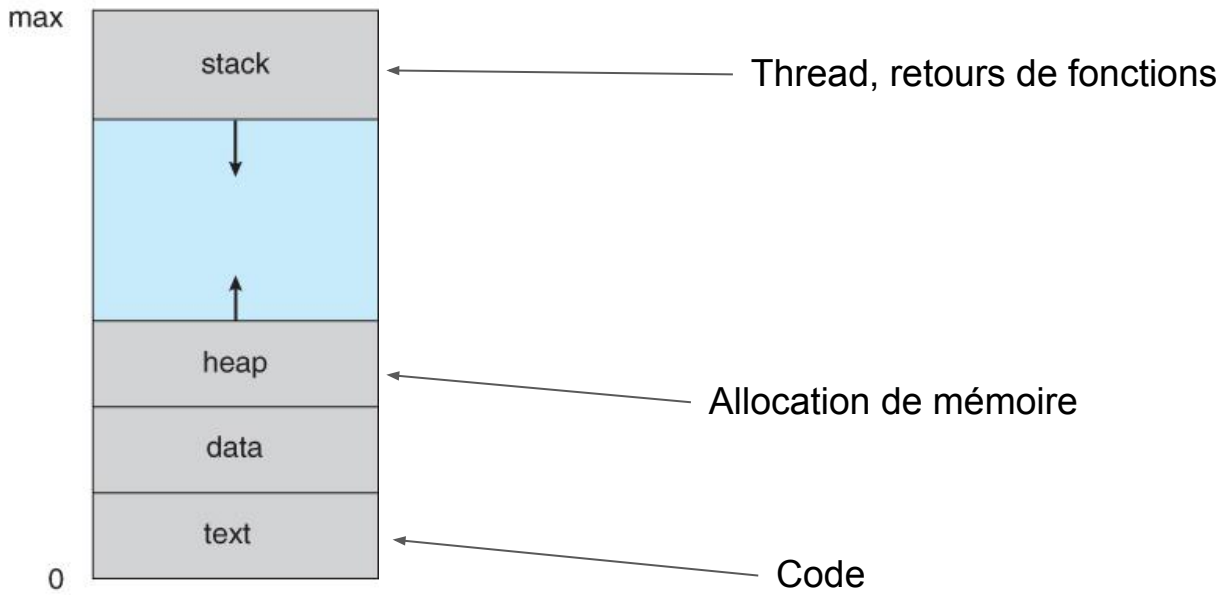
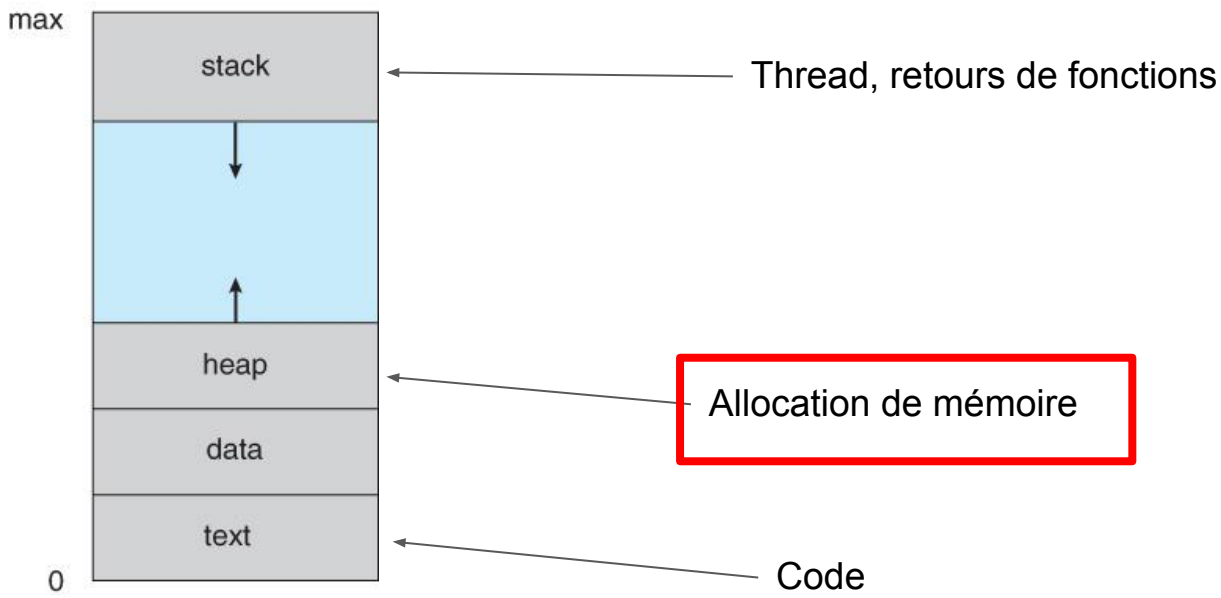
Gestion de la mémoire
Langage C

Jérôme Fink

Mémoire d’un processus

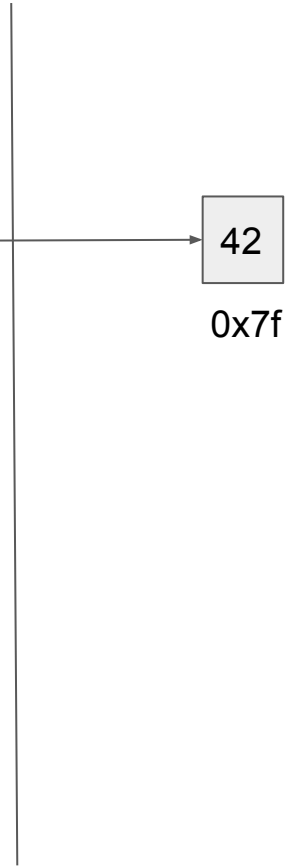


Mémoire d’un processus



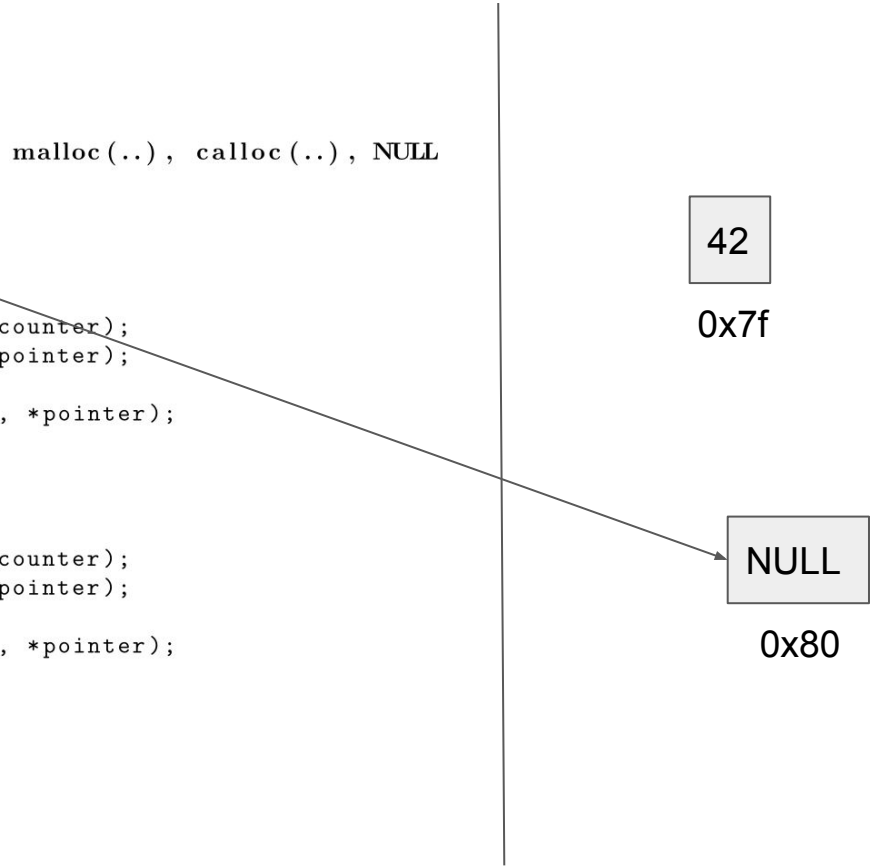
La mémoire en C

```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;
}
```



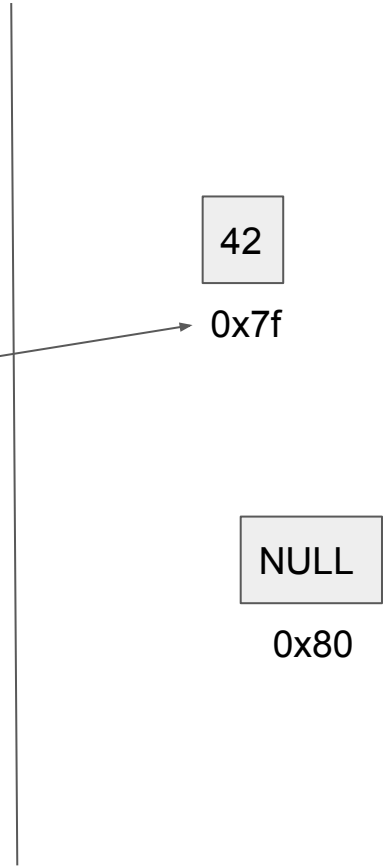
La mémoire en C

```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;
}
```



La mémoire en C

```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;
}
```

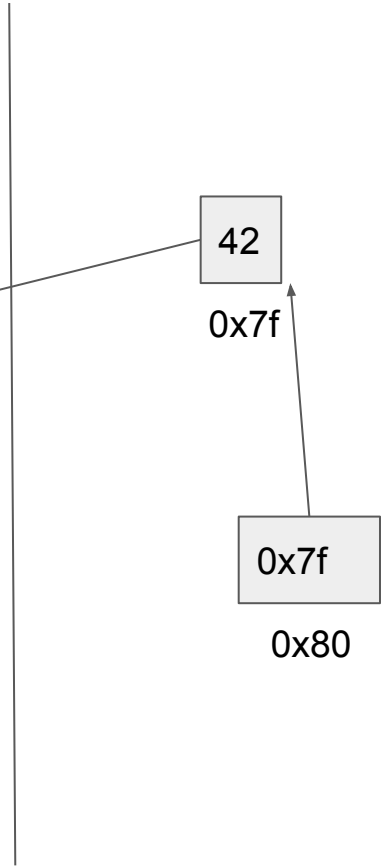


La mémoire en C

```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;
}
```

La mémoire en C

```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;
}
```



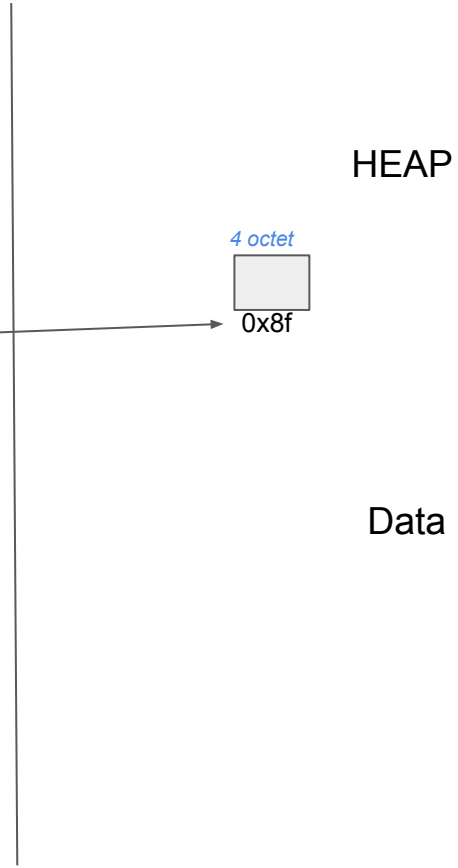
Opérateurs mémoires

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

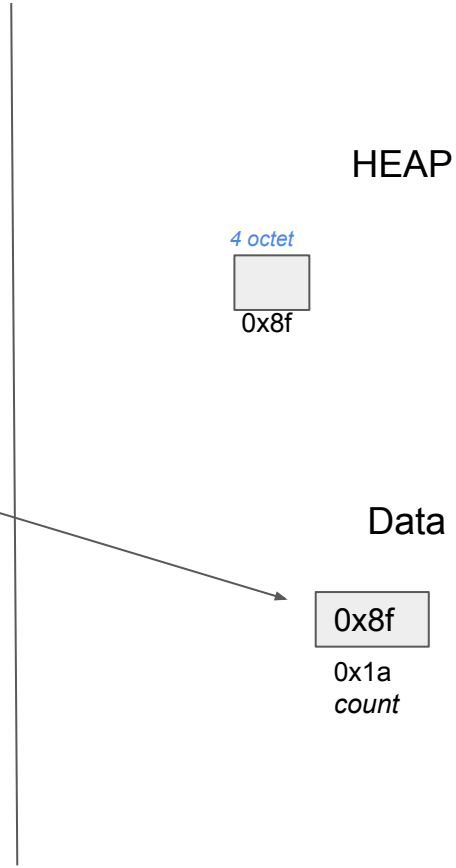
Allocation dynamique

```c
#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;
}
```



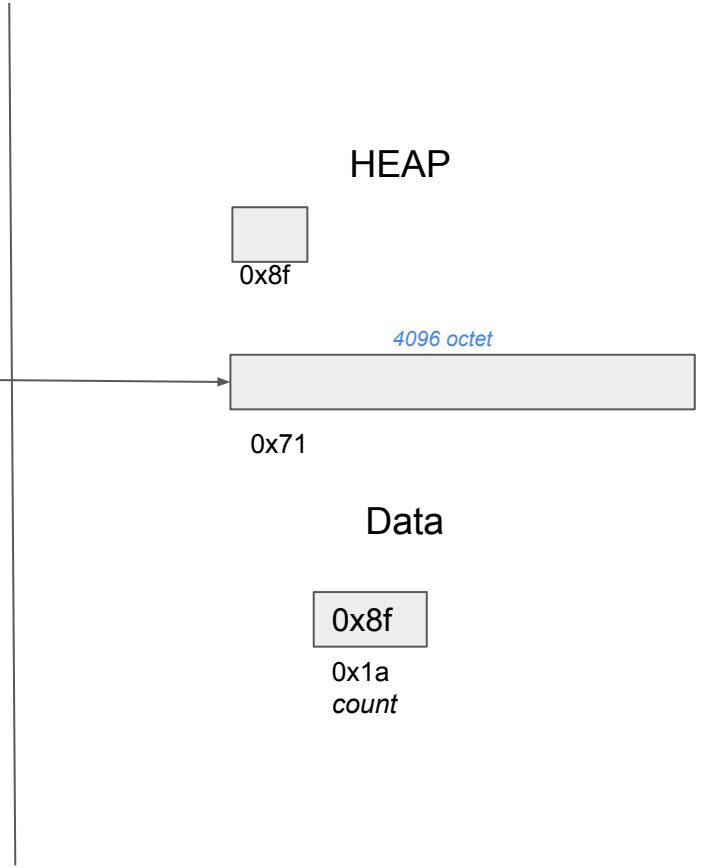
Allocation dynamique

```c
#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;
}
```



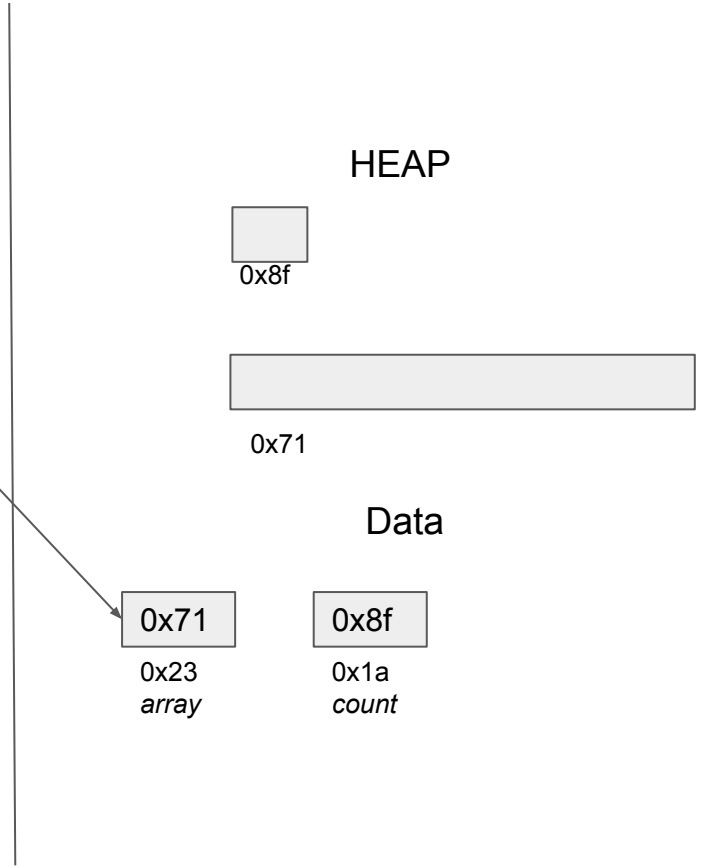
Allocation dynamique

```c
#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;
}
```



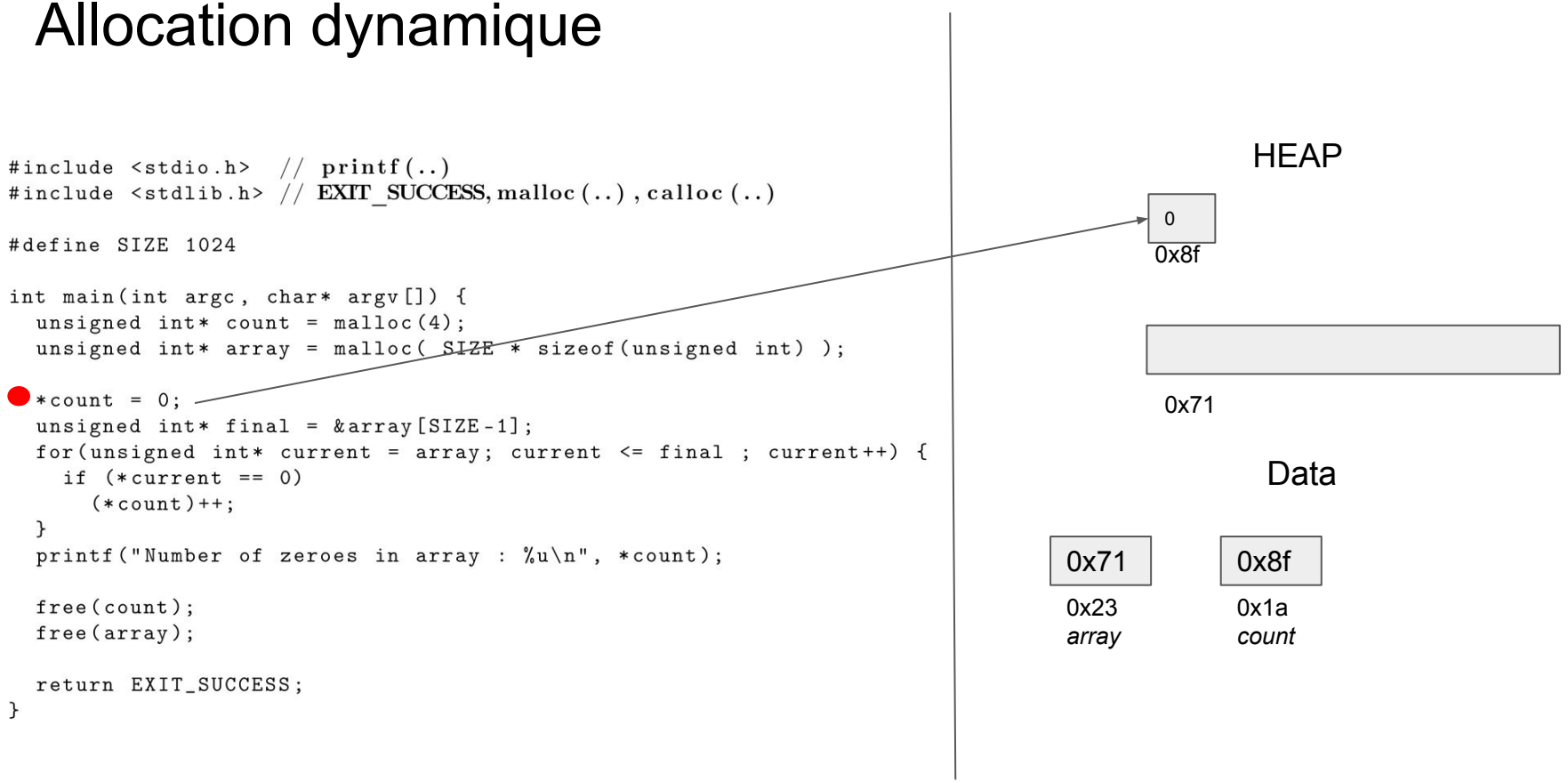
Allocation dynamique

```c
#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;
}
```



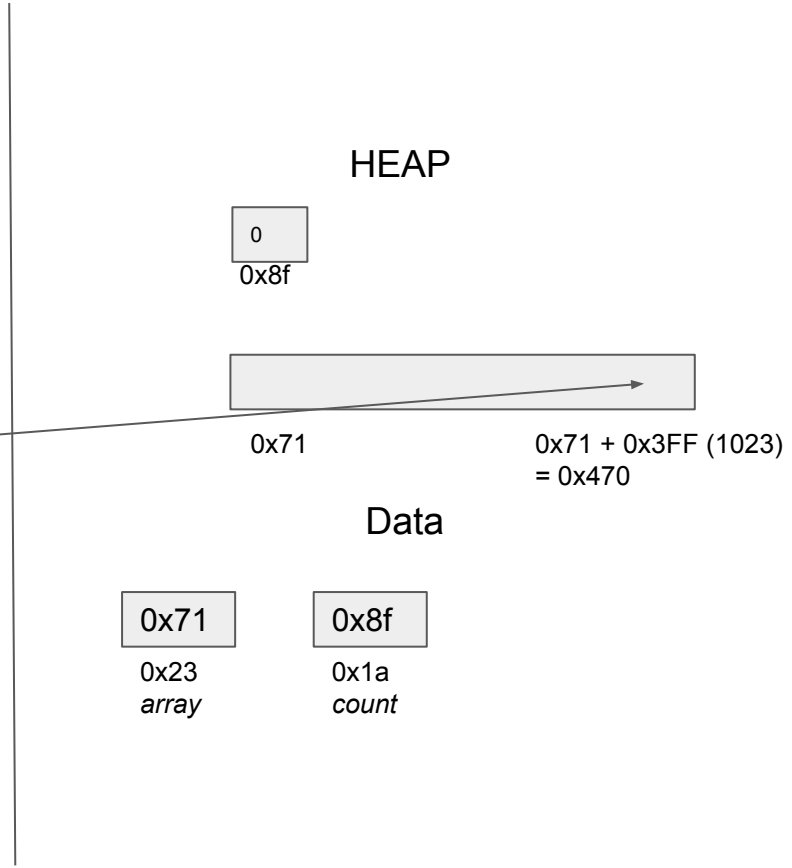
Allocation dynamique

```c
#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;
}
```



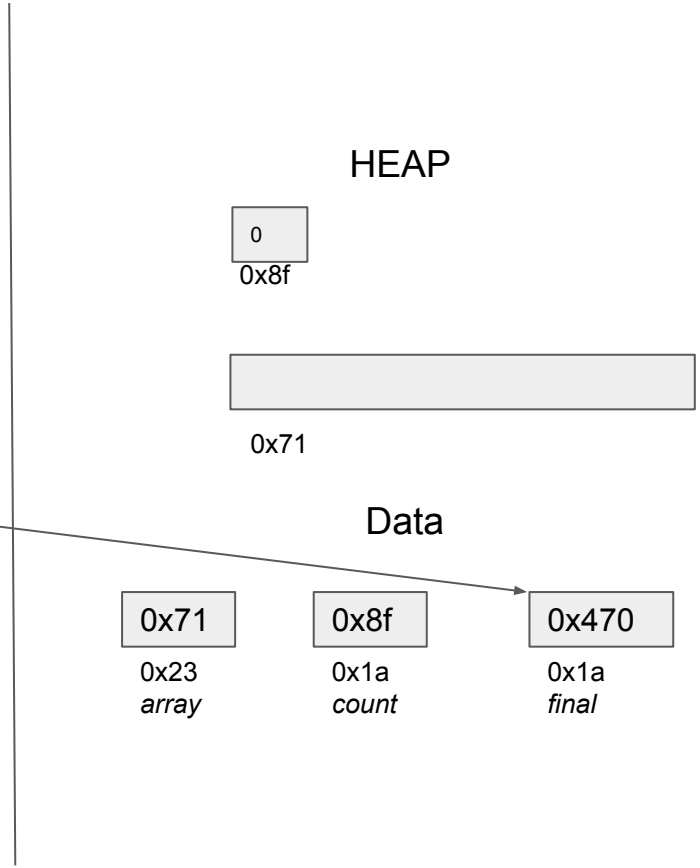
Allocation dynamique

```c
#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;
}
```



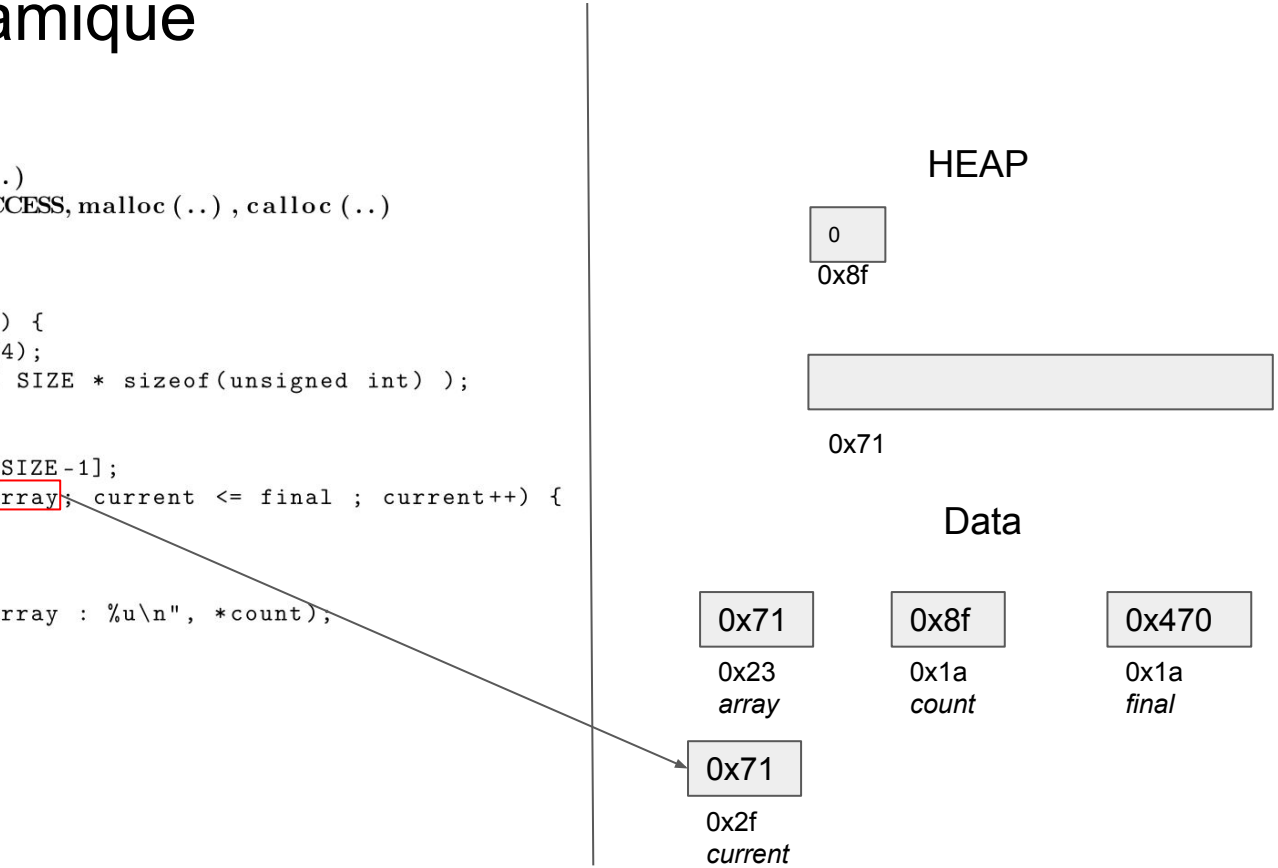
Allocation dynamique

```c
#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;
}
```



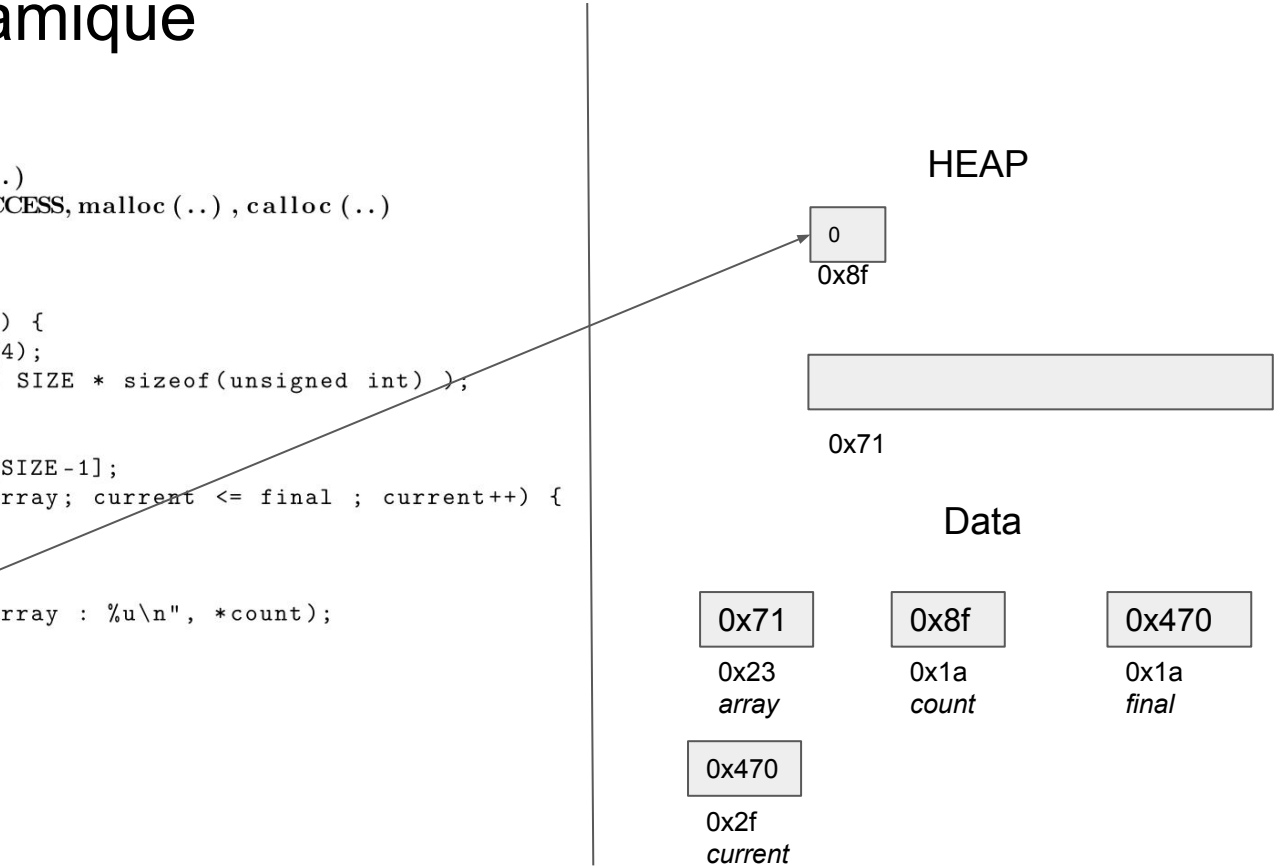
Allocation dynamique

```c
#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;
}
```



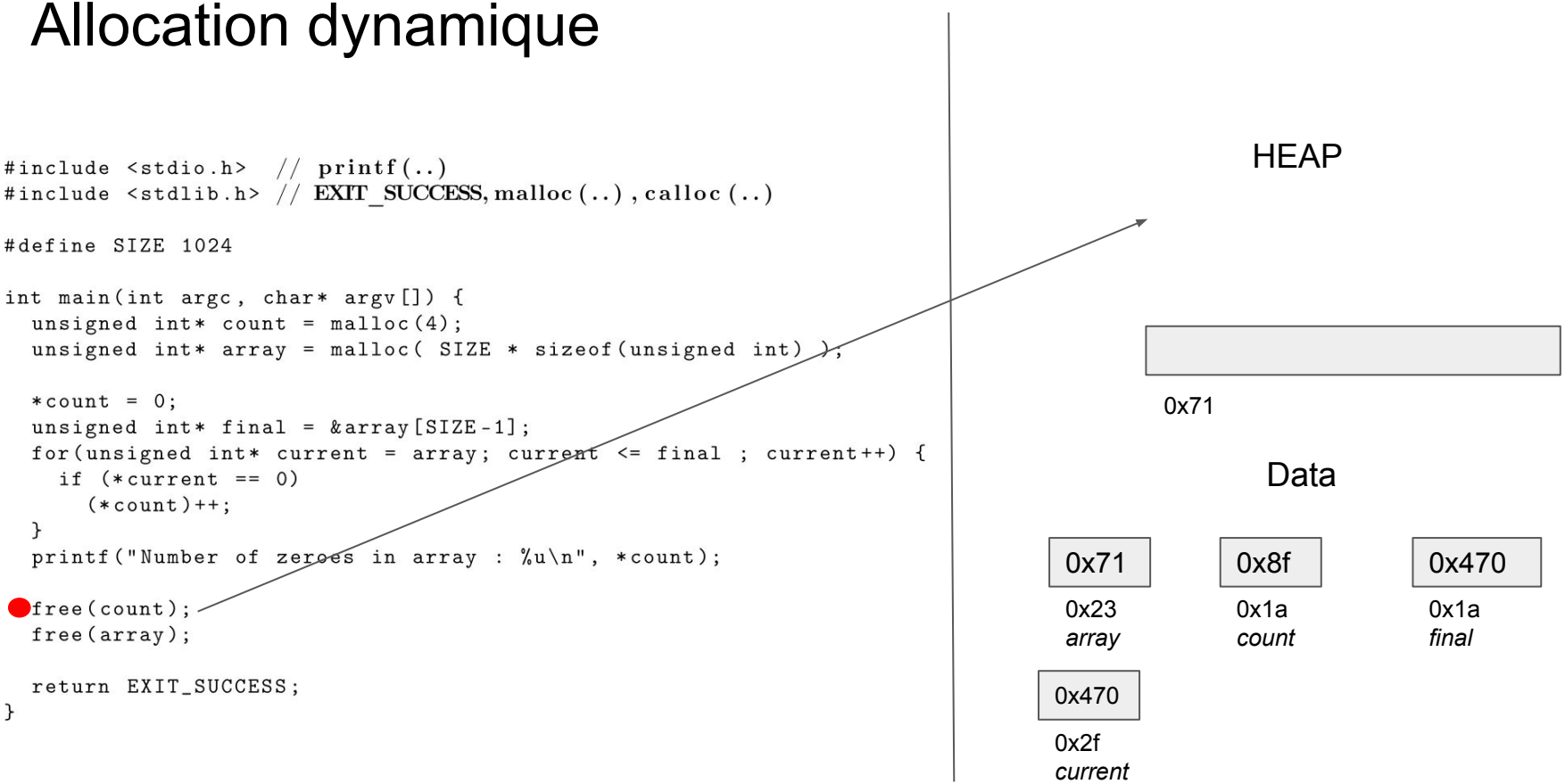
Allocation dynamique

```c
#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;
}
```



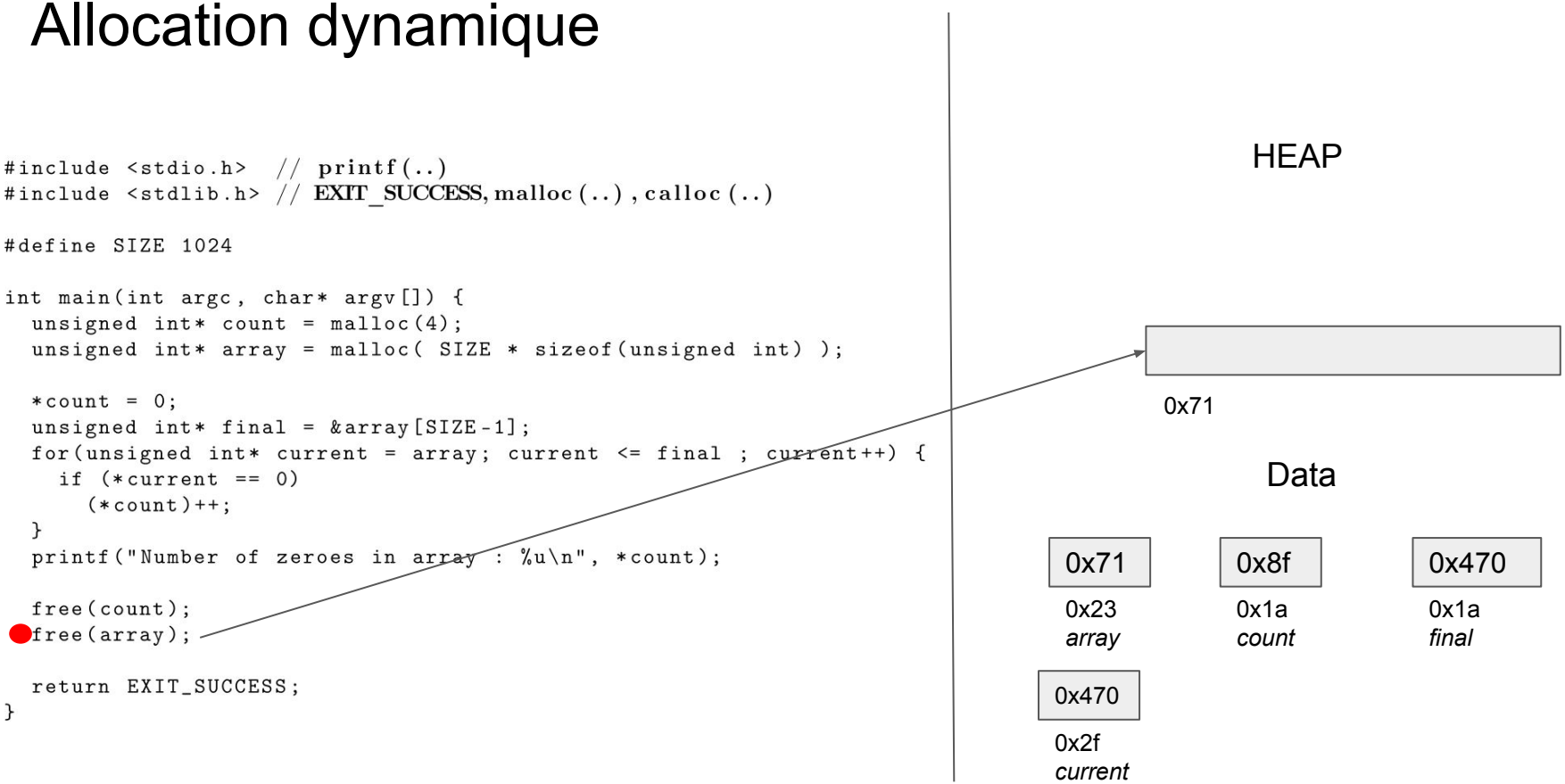
Allocation dynamique

```c
#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

```c
#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

```c
#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;
}
```