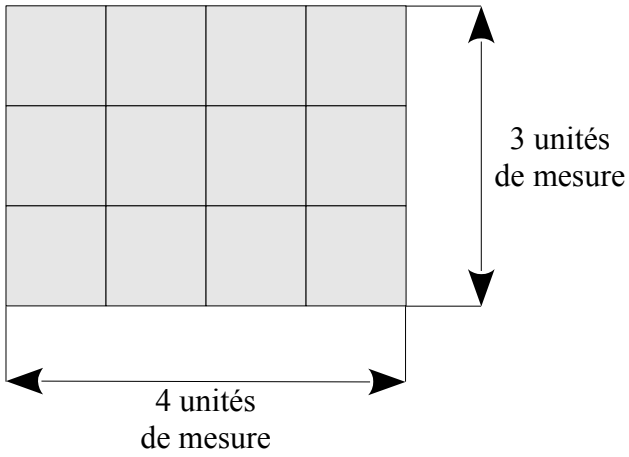
Programme parallèle de simulation du refroidissement d'une plaque de métal

Introduction :

On considère une plaque de métal dont on néglige l'épaisseur. La largeur et la longueur de cette plaque sont des paramètres du programme (pour cet exercice, on pourra, par exemple, considérer que la longueur est de 4 unités et que la largeur est de 3 unités).

La plaque est découpé en carrés de 1 unité de côté comme le montre le schéma ci-dessous.



La température de départ de chaque carré, ainsi que la température de l'air ambiant (autour de la plaque) sont des données qui doivent être fixées au démarrage du programme.

Pour cet exercice, on peut choisir :

- une température ambiante de 20°c ;
- une température de 30°c pour les carrés à l'exception d'un carré ayant une température de 50°c (un point chaud).

20°c
(air ambiant)

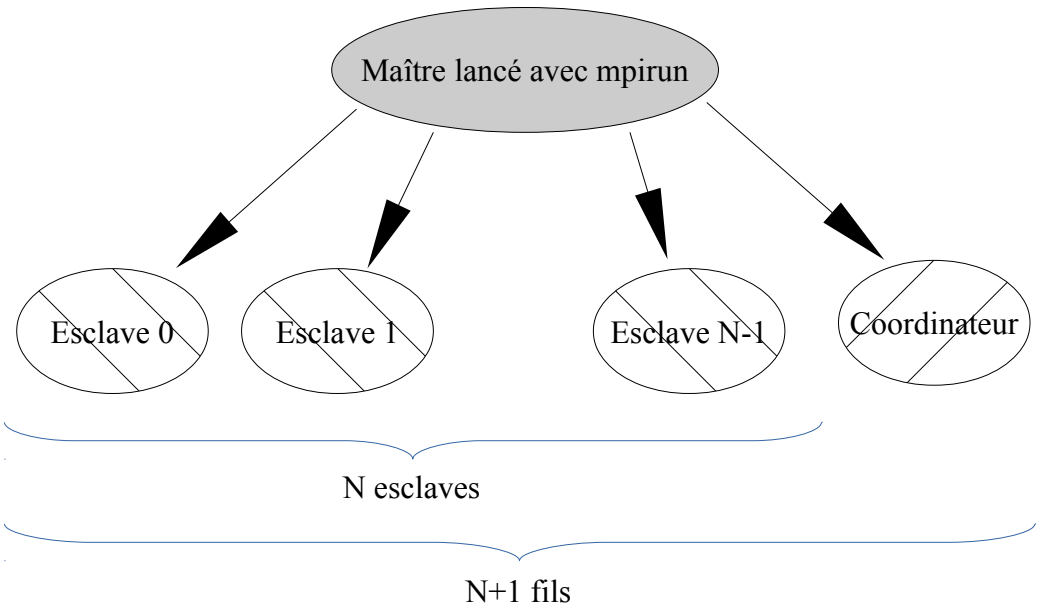
30°c	30°c	30°c	30°c
30°c	30°c	50°c	30°c
30°c	30°c	30°c	30°c

Jalon 1 :

Écrire un programme MPI « maître » qui génère (N+1) processus fils :

- 1 processus « coordinateur » ;
- N processus « esclaves » ;

La valeur de N correspond au nombre de carré au sein de la plaque de métal (chaque carré est pris en charge par un esclave).



Remarque : pour réaliser ce premier jalon, on peut utiliser les codes d'exemple fournis dans le support de cours pour illustrer l'utilisation de la fonction `MPI_Comm_spawn_multiple`.

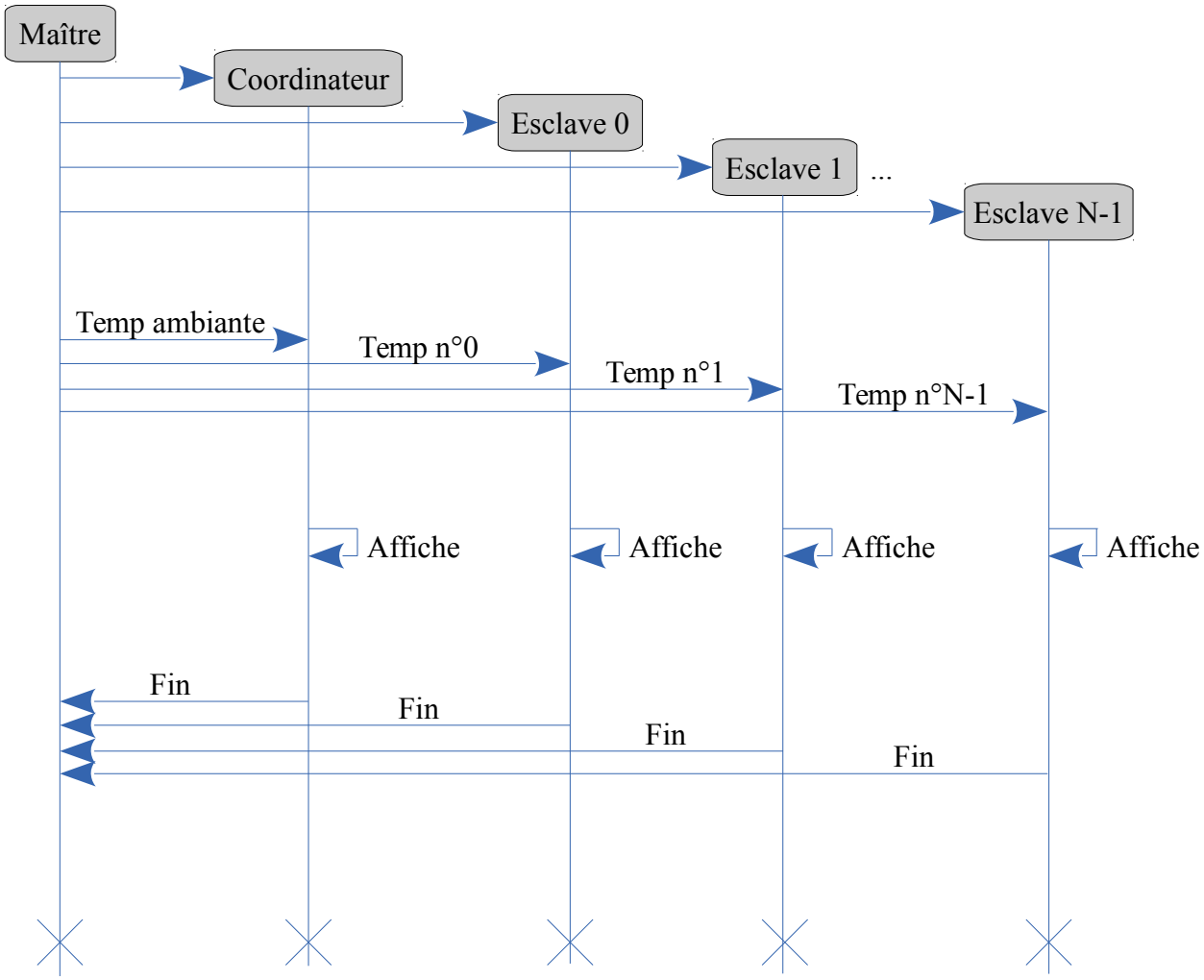
Jalon 2 :

Modifier le programme « maître » de manière à ce qu'il effectue les actions suivantes :

1. Création des N+1 fils.
2. Envoi de la température de départ aux N esclaves et envoi de la température ambiante au coordinateur.
3. Bloquer le maître sur la réception (cf. instruction bloquante `recv`), de la part des fils, d'un `char` ayant une valeur quelconque (la réception du `char` signifie que la simulation est terminée).
4. Arrêter le programme « maître » lorsque tous les `char` ont été reçus.

Modifier les programmes « esclave » et « coordinateur » de manière à ce qu'il effectue les actions suivantes :

1. Bloquer le programme sur la réception de la température de la part du maître
2. Afficher un message du type « L'esclave a recu la temperature xx »
3. Envoyer un char vers le maître pour lui signifier la fin de la simulation
4. Arrêter le programme « maître » lorsque tous les `char` ont été reçus.



Jalon 3 :

Modifier le programme « maître » de manière à ce qu'il effectue les actions suivantes :

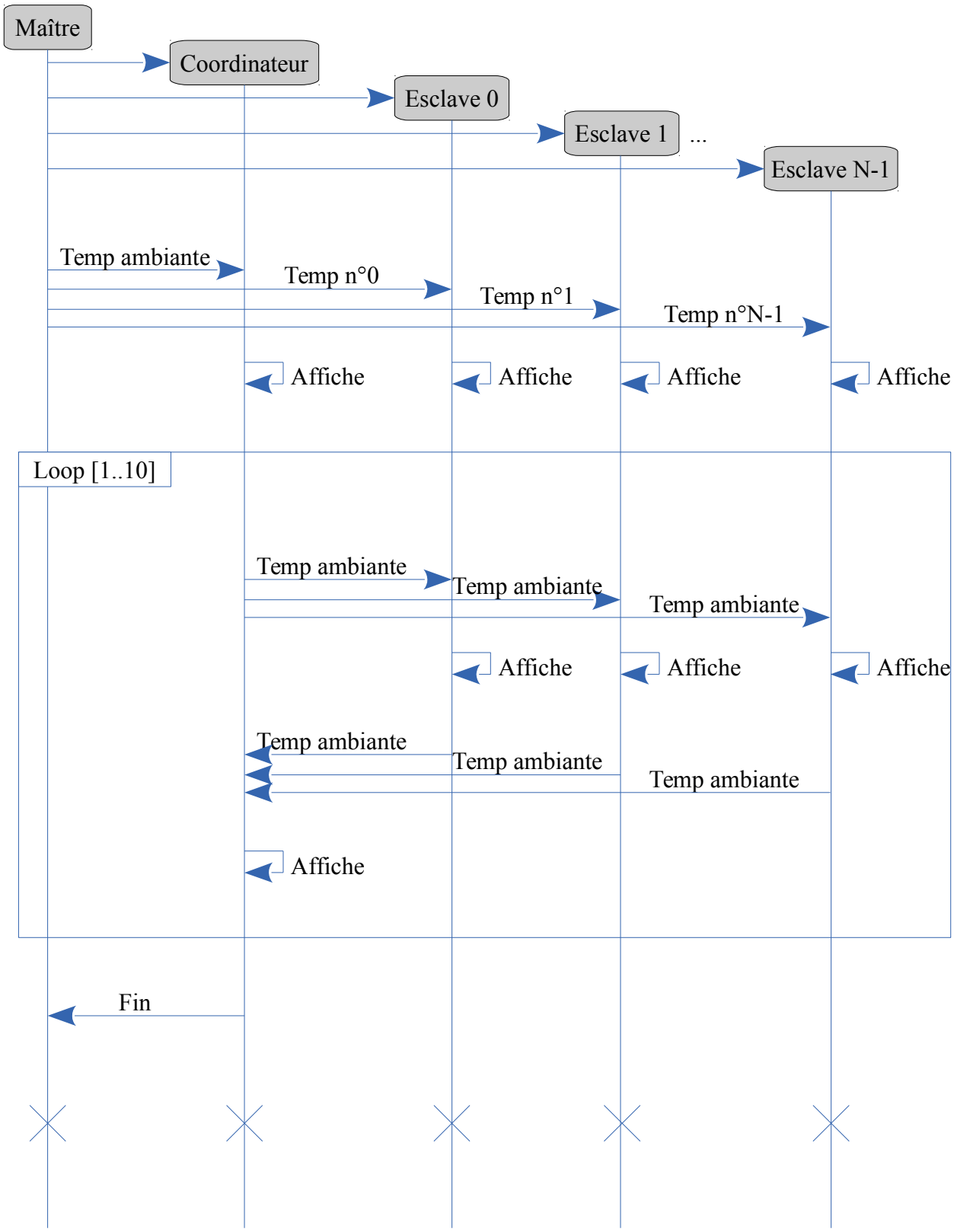
5. Création des N+1 fils.
6. Envoi de la température de départ aux N esclaves et envoi de la température ambiante au coordinateur.
7. Bloquer le maître sur la réception (cf. instruction bloquante `recv`), de la part du **coordinateur**, d'un `char` ayant une valeur quelconque (la réception du `char` signifie que la simulation est terminée).
8. Arrêter le programme « maître » lorsque ce char a été reçu.

Modifier le programme « coordinateur » de manière à ce qu'il effectue les actions suivantes :

1. Bloquer le programme sur la réception de la température ambiante de la part du maître
2. Afficher un message du type « Le coordinateur a recu la temperature ambiante »
3. Effectuer 10 fois les traitements suivants :
    a. Envoyer la température ambiante vers les esclaves (point de départ du pas de simulation)
    b. Attendre la réception de la température de la part des esclaves
    c. Afficher les températures reçues sous forme d'un tableau 2D pour faciliter la visualisation (fin du pas de simulation)
4. Envoyer un char vers le maître pour lui signifier la fin de la simulation
5. Arrêter le programme « coordinateur ».

Modifier le programme « esclave » de manière à ce qu'il effectue les actions suivantes :

1. Bloquer le programme sur la réception de la température de la part du maître
2. Afficher un message du type « L'esclave a recu la temperature xx de la part du maitre »
3. Effectuer 10 fois les traitements suivants :
    1. Attendre la réception de la température ambiante de la part du coordinateur (point de départ du pas de simulation)
    2. Afficher un message du type « L'esclave a recu la temperature ambiante de la part du coordinateur »
    3. Envoyer la température ambiante vers le coordinateur (fin du pas de simulation)
4. Arrêter le programme « esclave »


Maître
Coordinateur
Esclave 0
Esclave 1
...
Esclave N-1
Temp ambiante
Temp n°0
Temp n°1
Temp n°N-1
Affiche
Affiche
Affiche
Affiche
Loop [1..10]
Temp ambiante
Temp ambiante
Temp ambiante
Affiche
Affiche
Affiche
Temp ambiante
Temp ambiante
Temp ambiante
Affiche
Fin

Jalon 4 :

Modifier le programme « maître » de manière à ce qu'il effectue les actions suivantes :

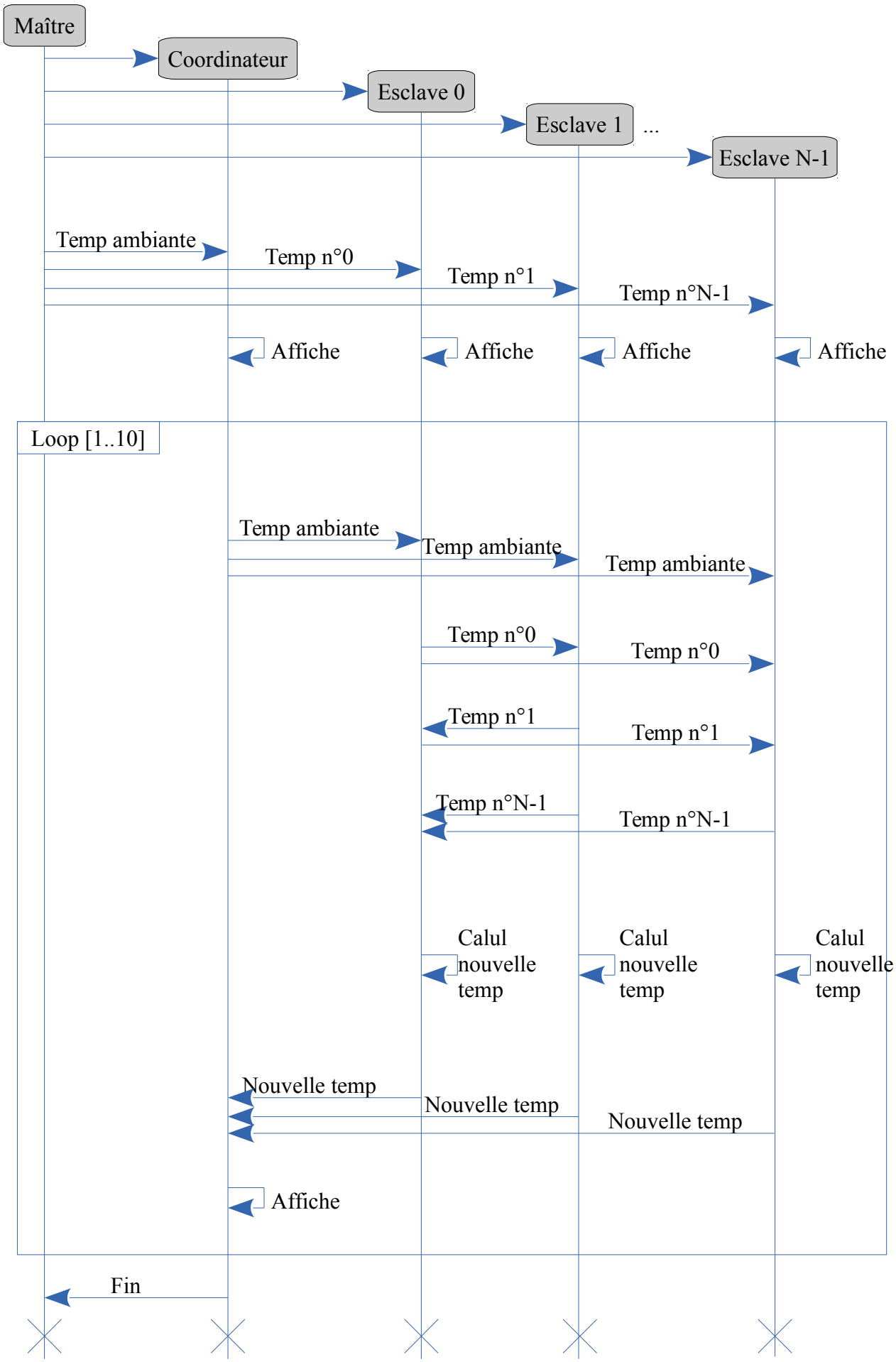
1. Création des N+1 fils.
2. Envoi de la température de départ aux N esclaves et envoi de la température ambiante au coordinateur.
3. **Envoi de la longueur de la plaque aux esclaves et au coordinateur**
4. Bloquer le maître sur la réception (cf. instruction bloquante `recv`), de la part du coordinateur, d'un `char` ayant une valeur quelconque (la réception du `char` signifie que la simulation est terminée).
5. Arrêter le programme « maître » lorsque ce char a été reçu.

Modifier le programme « coordinateur » de manière à ce qu'il effectue les actions suivantes :

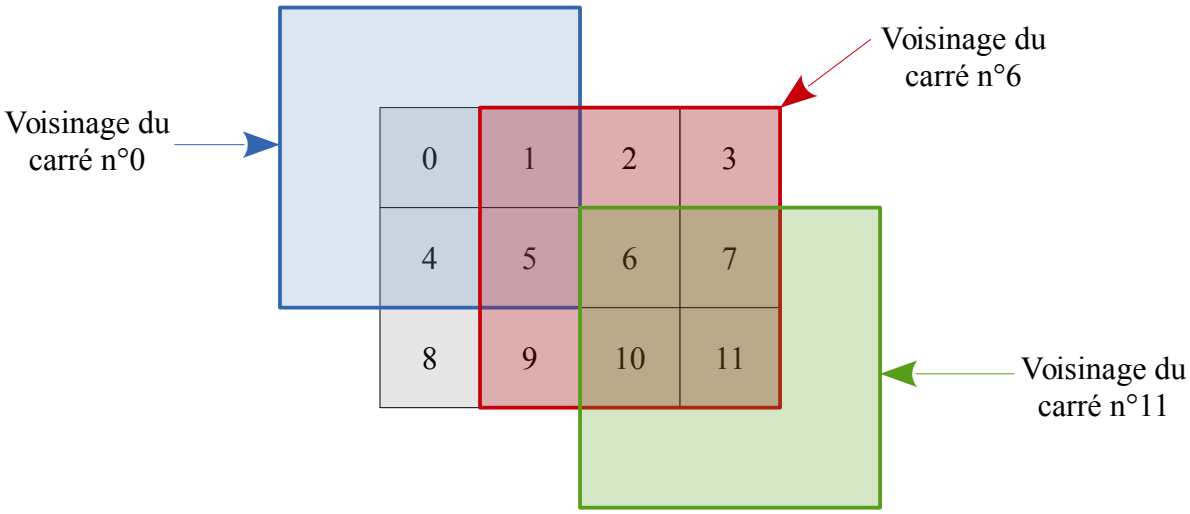
1. Bloquer le programme sur la réception de la température ambiante de la part du maître
2. Afficher un message du type « Le coordinateur a recu la temperature ambiante »
3. **Bloquer le programme sur la réception de la longueur de la plaque de la part du maître**
4. **Afficher un message du type « Le coordinateur a recu la longueur de la plaque »**
5. Effectuer 10 fois les traitements suivants :
    a. Envoyer la température ambiante vers les esclaves (point de départ du pas de simulation)
    b. Attendre la réception de la température de la part des esclaves
    c. Afficher les températures reçues sous forme d'un tableau 2D pour faciliter la visualisation (fin du pas de simulation)
4. Envoyer un char vers le maître pour lui signifier la fin de la simulation
5. Arrêter le programme « coordinateur ».

Modifier le programme « esclave » de manière à ce qu'il effectue les actions suivantes :

1. Bloquer le programme sur la réception de la température de la part du maître
2. Afficher un message du type « L'esclave a recu la temperature xx de la part du maitre »
3. **Bloquer le programme sur la réception de la longueur de la plaque de la part du maître**
4. **Afficher un message du type « Le coordinateur a recu la longueur de la plaque »**
5. Effectuer 10 fois les traitements suivants :
    a. Attendre la réception de la température ambiante de la part du coordinateur (point de départ du pas de simulation)
    b. Envoyer sa température à tous ses voisins en mode asynchrone
    c. Attendre la réception des températures des voisins en mode synchrone (si le voisin n'existe pas, on doit considérer la température ambiante)
    d. Calculer la nouvelle température du carré comme étant la moyenne des 9 températures (l'ancienne température du carré et les températures des 8 voisins)
    e. Afficher un message du type « L'esclave a mis à jour sa température »
    f. Envoyer la température ambiante vers le coordinateur (fin du pas de simulation)
6. Arrêter le programme « esclave »


Maître
Coordinateur
Esclave 0
Esclave 1
...
Esclave N-1
Temp ambiante
Temp n°0
Temp n°1
Temp n°N-1
Affiche
Affiche
Affiche
Affiche
Loop [1..10]
Temp ambiante
Temp ambiante
Temp ambiante
Temp n°0
Temp n°0
Temp n°1
Temp n°1
Temp n°N-1
Temp n°N-1
Calul nouvelle temp
Calul nouvelle temp
Calul nouvelle temp
Nouvelle temp
Nouvelle temp
Nouvelle temp
Affiche
Fin

Pour chaque carré, il faut déterminer les carrés voisins ce qui revient à déterminer pour chaque processus esclaves, les autres processus esclaves avec lesquels il devra communiquer (envoyer sa température et recevoir leur température).



Jalon 5 :

Mettre en place un système d'affichage graphique qui sera utilisée par le **coordinateur** à chaque pas de simulation. Pour ce faire, deux solutions s'offre à vous :

- utiliser SVG ;
- utiliser OpenGL.

La solution basée sur SVG :

Cette solution consiste à écrire, à chaque pas de simulation, un fichier HTML contenant des balises SVG. Le squelette de ce fichier est le suivant :

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Refroidissement d'une plaque de métal</title>
 </head>
 <body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1"
       width="200px" height="200px">

   <rect x="0" y="0" width="100%" height="100%" fill="none" stroke="black"/>

  </svg>
 </body>
</html>
```

La partie, écrite en rouge, correspond à « l'entête » du fichier alors que la partie, écrite en vert, correspond à la fin de ce même fichier.

L'objectif est donc d'écrire les balises `rect` avec les bonnes valeurs de manière à obtenir le résultat souhaité. Les paramètre de la balise `rect` sont les suivants :

- `x` et `y` sont les coordonnées du coin supérieur gauche ;
- `width` et `height` correspondent aux dimensions de ce rectangle (qui peuvent être données en nombre de pixels ou encore en pourcentages par rapport à la zone d'affichage de l'image SVG) ;
- `fill` permet de spécifier la couleur de remplissage ;
- `stroke` correspond à la couleur du trait délimitant le rectangle.

La couleur peut être spécifiée :

- en utilisant des symboles prédéfinis comme `black`, `red` ou `green` ;
- en utilisant le codage RVB qui peut être lui-même spécifié selon différents formats :
  - `#rvb` (exemple : `#F56`)
  - `#rrvvbb` (exemple : `#56FCA6`)
  - `rvb(r,v,b)` (exemple : `rvb(123,67,34)`)
  - `rvb(r%, v%, b%)` (exemple : `rvb(24%, 33%, 12%)`)

La solution basée sur OpenGL :

Cette solution est plus complexe mais elle présente l'avantage d'être plus interactive. Dans cette solution, le coordinateur doit :

- créer une fenêtre graphique ;
- dessiner la plaque de métal grâce aux instructions OpenGL ;
- refermer la fenêtre graphique.

Ces différents actions correspondront à des fonctions à ajouter dans le code du coordinateur.

Pour cela, il sera d'abord nécessaire d'ajouter les bibliothèque GL et GLU à votre système en exécutant la commande ci-dessous :

```
sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

Il faudra ensuite compiler la bibliothèque SDL dont les sources de la version 2 sont disponibles à l'URL ci-dessous :

https://www.libsdl.org/download-2.0.php

Vous pouvez enfin vous inspirer du programme ci-dessous qui dessine un simple carré à l'écran.

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

#include <SDL2/SDL.h>
#include <SDL2/SDL_opengl.h>

#include <GL/gl.h>
#include <GL/glu.h>

SDL_Renderer* FenetreGraphique_creer (int    largeurDeLaFenetre,
                                      int    hauteurDeLaFenetre,
                                      double xMin,
                                      double xMax,
                                      double yMin,
                                      double yMax)
 {
  SDL_Init (SDL_INIT_VIDEO);

  SDL_Window*      displayWindow;
  SDL_RendererInfo displayRendererInfo;
  SDL_Renderer*    displayRenderer;

       SDL_CreateWindowAndRenderer    (largeurDeLaFenetre,    hauteurDeLaFenetre,    SDL_WINDOW_OPENGL,
&displayWindow, &displayRenderer);
  SDL_GetRendererInfo(displayRenderer, &displayRendererInfo);

  glViewport (0, 0, (GLsizei) largeurDeLaFenetre, (GLsizei) hauteurDeLaFenetre);

  glMatrixMode( GL_PROJECTION );
  glLoadIdentity( );

  glOrtho (xMin, xMax, yMin, yMax, 1, -1);

  glMatrixMode( GL_MODELVIEW );
  glLoadIdentity( );

  return displayRenderer;
 }
```

```
void FenetreGraphique_initialiser ()
 {
  glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 }

void FenetreGraphique_rendre (SDL_Renderer* displayRenderer)
 {
  SDL_RenderPresent (displayRenderer);
  SDL_Delay(5000);
 }

void FenetreGraphique_fermer ()
 {
  SDL_Quit();
 }

void FenetreGrahique_dessinerRectangle (double x1, double y1,
                                        double x2, double y2,
                                        float  fondR, float fondV, float fondB,
                                        float  formeR, float formeV, float formeB)
 {
  glColor3f (fondR, fondV ,fondB);

  glBegin (GL_QUADS);
   glVertex2f (x1, y1);
   glVertex2f (x2, y1);
   glVertex2f (x2, y2);
   glVertex2f (x1, y2);
  glEnd ();

  glColor3f (formeR, formeV ,formeB);

  glBegin (GL_LINE_LOOP);
   glVertex2f (x1, y1);
   glVertex2f (x2, y1);
   glVertex2f (x2, y2);
   glVertex2f (x1, y2);
  glEnd ();
 }


int main(int argc, char *argv[])
 {
  SDL_Renderer* displayRenderer = FenetreGraphique_creer (800, 600, -2, 2, -2, 2);

  FenetreGraphique_initialiser ();

  FenetreGrahique_dessinerRectangle (-0.8, -0.8, 0.8, 0.8,
                                     1, 0, 1,
                                     0, 1, 0);

  FenetreGraphique_rendre (displayRenderer);
  FenetreGraphique_fermer ();

  return 0;
 }
```

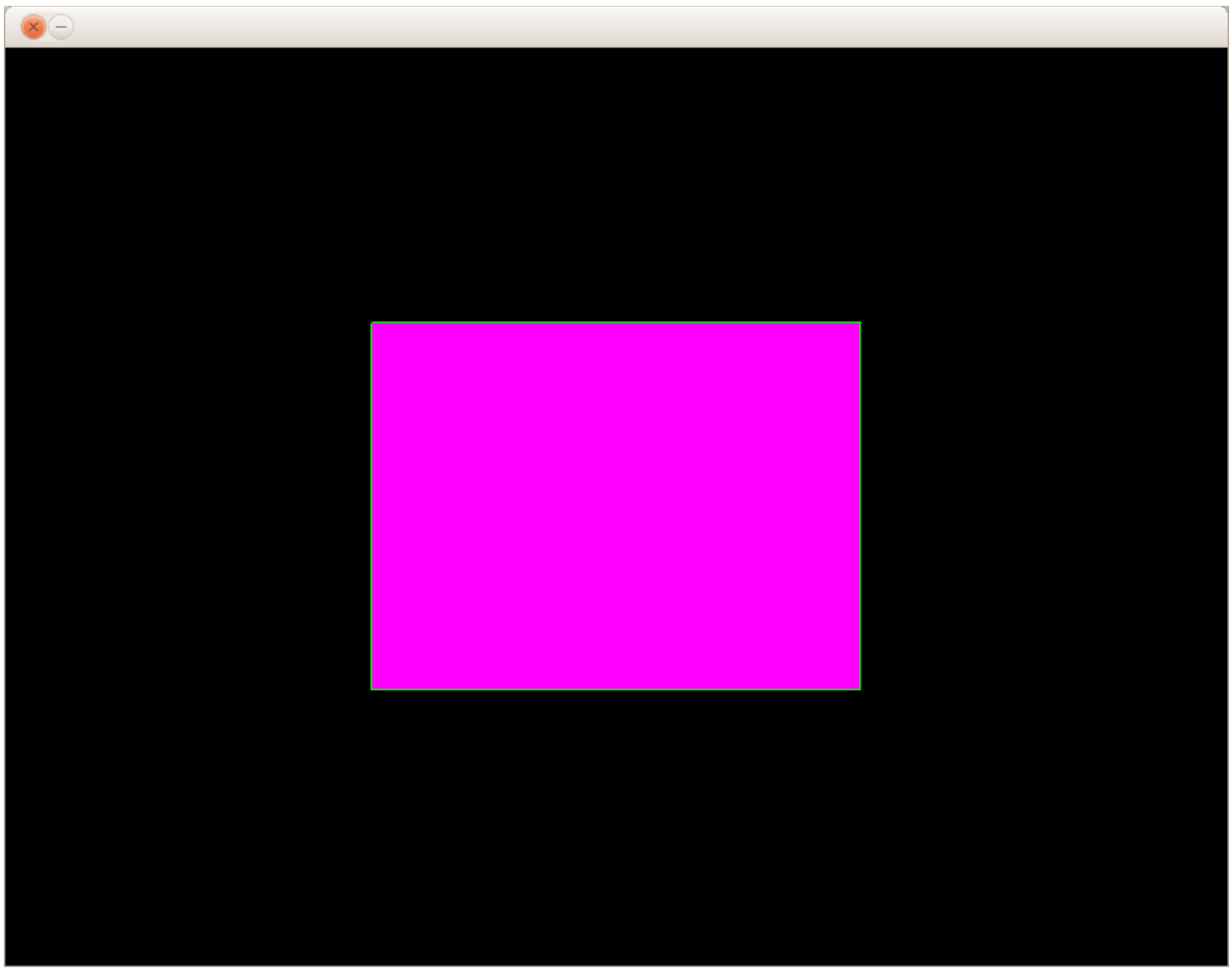
Vous pouvez compiler ce programme en utilisant la ligne de commande ci-dessous :

```
gcc fenetreGraphique.c -o fenetreGraphique -lDSL2 -lGL -lGLU  -I/rep/SDL2/include -L/rep/SDL2/lib
```

Vous devrez également ajouter le répertoire `lib` à votre variable `LD_LIBRARY_PATH` en exécutant la commande ci-dessous :

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:repertoireInstall/lib
```

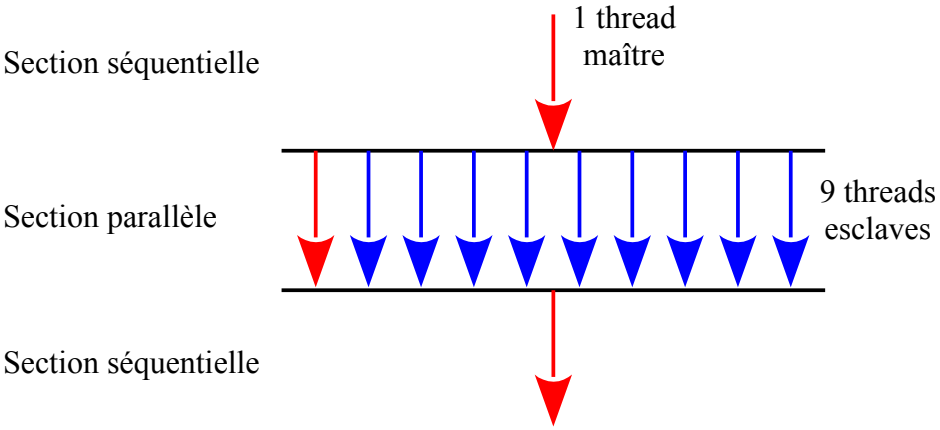
Normalement, si vous compilez cet exemple, vous devez obtenir la fenêtre de la page suivante.



Jalon 6 :

Faire un **autre code indépendant du programme des jalons 1 à 5**, comportant des directives OpenMP, qui va simuler le refroidissement d'une plaque de 3x3 où un thread esclave gère une case.

Les threads esclaves sont créés automatiquement au sein d'une section parallèle OpenMP alors que le thread maître correspond au « main » du programme (il est présent dans les sections parallèles et séquentielles).



Pour un pas de simulation, le scénario de fonctionnement est le suivant :

- Le thread maître lit un tableau « départ » contenant la température des 9 cases ainsi que la température ambiante.
- Le thread maître détermine un ensemble de températures à fournir à chaque thread esclave (la température correspondant à la case et les températures des cases voisines)

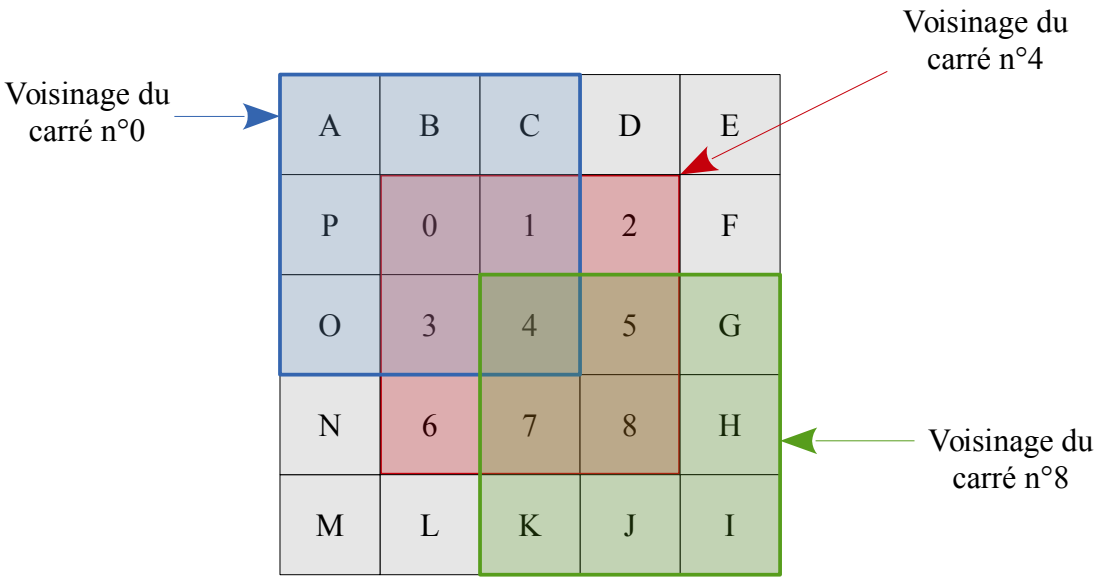
- Le thread esclave récupère les températures de ses voisins et de lui-même
- Le thread esclave calcule sa nouvelle températures (comme une moyenne des températures reçues précédemment)
- Le thread écrit sa nouvelle température **dans un autre tableau « résultat »**.

- Le thread maître recopie le contenu de « résultat » dans « départ » pour l'itération suivante.

Au niveau du codage, il n'est pas nécessaire d'installer une bibliothèque quelconque car les compilateurs modernes intègrent déjà OpenMP. Dans le cas de gcc (et par voie de conséquence de mpicc), il suffit d'ajouter la directive `-fopenmp` (dans le code, il faudra cependant faire un include du fichier omp.h.

Le codage doit d'abord est fait en séquentiel. Vous écrivez un programme qui calcule l'évolution de la température de ces 9 cases sur une dizaine d'itérations.

Lorsque ce code séquentiel fonctionne, vous pouvez alors ajouter des directives OpenMP pour paralléliser le programme.


Voisinage du carré n°4
Voisinage du carré n°0
A
B
C
D
E
P
0
1
2
F
O
3
4
5
G
N
6
7
8
H
M
L
K
J
I
Voisinage du carré n°8

Jalon 7 :

L'objectif est de créer un programme qui fusionne les codes des jalons 5 et 6.

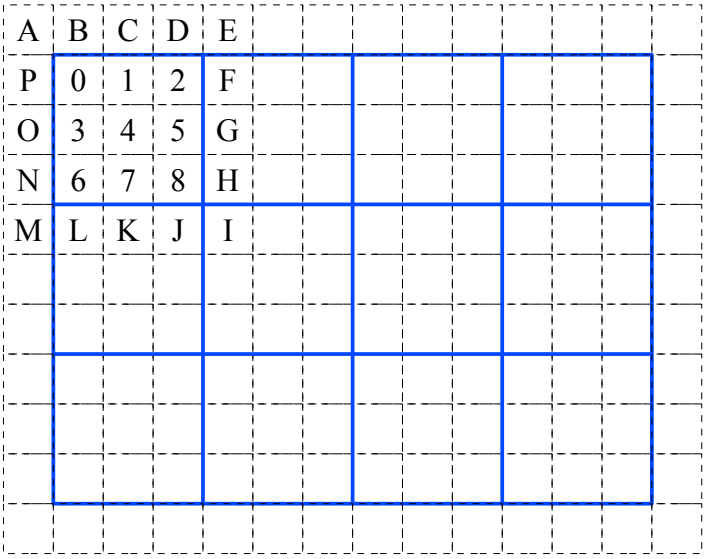
D'un point de vue du modèle, cela veut dire que chaque carré géré par un processus MPI sera subdivisé en petits carrés gérés par des threads via OpenMP.

On suppose le carré géré par le premier esclave MPI.

Si on considère par exemple, le carré n°4, pris en charge par le thread esclave n°4, le thread maître doit lui fournir les températures des cases 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

Pour la case n°0, le thread maître doit envoyer des cases 0, 1, 3 et 4 mais également les températures des cases A, B, C, P et O. Ces cases correspondent à des cases des blocs « voisins » à l'air ambiant comme le montre le schéma ci-dessous.

Pour la case n°8, le thread maître doit envoyer des cases 4, 5, 7 et 8 mais également les températures des cases G, H, I, J et K. Ces cases correspondent cette fois-ci à des cases des blocs « voisins »comme le montre le schéma ci-dessous.



Pour obtenir un tel résultat, les processus MPI ne doivent plus échanger des températures « simples » mais des des tableaux de températures (un processus MPI va par exemple envoyer, les 9 températures des 9 mini-cases qui composent son carré.

Jalon 8 :

L'objectif est simplement de modifier le programme de manière à passer d'un nombre d'itérations fixée à un nombre d'itérations calculées de manière à arrêter le calcul lorsque le système cesse d'évoluer.

Autrement dit, la différence entre la température moyenne de la plaque à l'instant précédent et la température moyenne à l'instant courant est inférieur à un seuil spécifié dans le programme ou passé en paramètre de ce même programme.