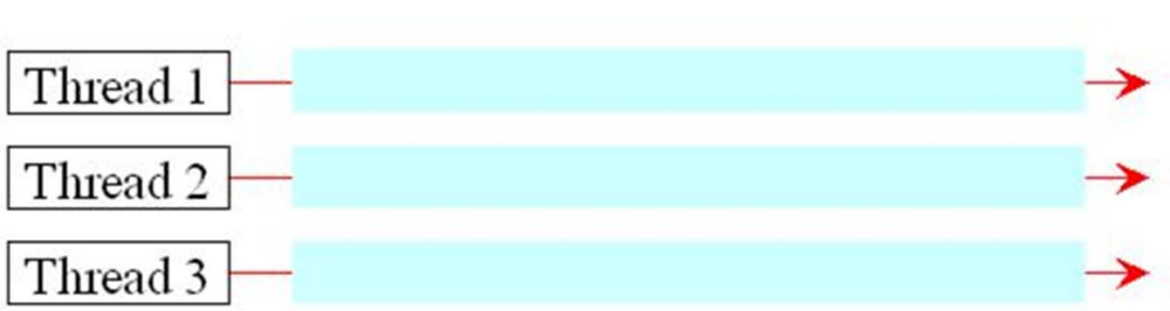
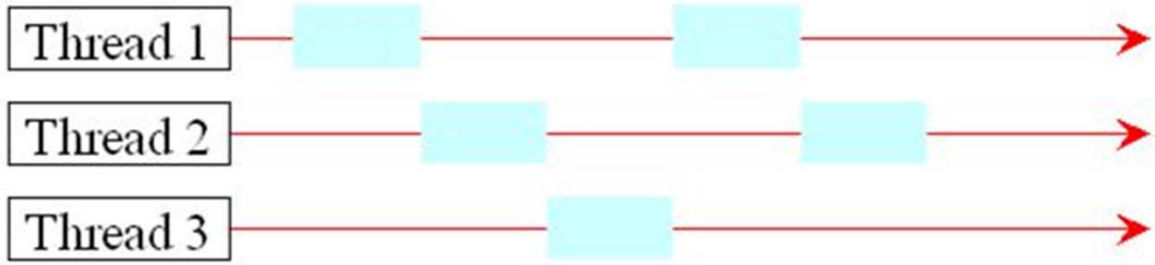
# Threads

# Concept



# Classe std::thread

- Permet de lancer un nouveau thread d'exécution
- Commence son exécution dès sa création
- Ex :
  - std::thread t1(''NOM FONCTION A EXÉCUTER'', ''ARGS'');
  - t1.join() //Permet d'attendre la fin de l'exécution

# Classe std::future

- Permet de récupérer la valeur d'une tâche asynchrone
- Nécessite de spécifier le type de valeur récupérer lors de la création
- Ex avec std::async :
  - std::future<int> f1 = std::async(std::launch::async, [](){ return 1; });

# Function std::async

- Constructeur : (‘’POLICE’’, ‘’FUNCTION’’, ‘’ARGS)
- Exécute la fonction passée en argument de façon asynchrone et retourne un objet de type std::future
- La majorité des cas nécessiteront de passer l’argument std::launch::async au premier argument

# Aller plus loin :

- std::promise : stocker une valeur pour la partager entre différents threads
- std::mutex (et compagnie) : bloquer l'accès à une variable pour éviter des collisions
- std::condition_variable : bloquer l'exécution de un ou plusieurs threads jusqu'à ce que la condition variable soit vraie