OpenMP Cheat Sheet

Voici une liste complète des directives et fonctions les plus utiles en OpenMP.

---

1. Directives de parallélisation

Ces directives permettent de définir des régions parallèles, des boucles ou des sections spécifiques du code.

Directive	Description
`#pragma omp parallel`	Crée une région parallèle : le code à l'intérieur est exécuté par plusieurs threads.
`#pragma omp for`	Parallélise une boucle `for`. À combiner avec `parallel` pour exécuter la boucle sur plusieurs threads.
`#pragma omp sections`	Permet de paralléliser plusieurs blocs de code indépendants (chaque section est exécutée par un thread).
`#pragma omp single`	Exécute un bloc de code par un seul thread (utile dans une région parallèle).
`#pragma omp critical`	Protège une section de code pour qu'un seul thread à la fois l'exécute (section critique).
`#pragma omp atomic`	Rend une opération atomique (sécurisée pour les mises à jour concurrentes d'une variable simple).
`#pragma omp barrier`	Synchronise tous les threads à un point précis : tous attendent que les autres terminent avant de continuer.
`#pragma omp master`	Exécute le bloc de code seulement par le thread maître (thread 0).
`#pragma omp task`	Crée une tâche indépendante qui peut être exécutée par n'importe quel thread disponible.

---

2. Fonctions de gestion des threads

Ces fonctions permettent de contrôler le nombre de threads et d'obtenir des informations sur eux.

Fonction	Description
`omp_get_num_threads()`	Renvoie le nombre de threads dans la région parallèle actuelle.
`omp_get_thread_num()`	Renvoie l'identifiant du thread courant (0 pour le thread maître).
`omp_get_max_threads()`	Renvoie le nombre maximum de threads utilisables si une région parallèle est créée.
`omp_set_num_threads(int n)`	Définit le nombre de threads à utiliser pour les prochaines régions parallèles.
`omp_get_num_procs()`	Renvoie le nombre de processeurs disponibles sur la machine.
`omp_in_parallel()`	Indique si le code est actuellement exécuté dans une région parallèle (1 si oui, 0 si non).

3. Fonctions de synchronisation et timing

Utile pour mesurer les performances ou synchroniser les threads.

Fonction	Description
`omp_get_wtime()`	Renvoie le temps écoulé en secondes depuis une référence arbitraire (pratique pour mesurer la durée d'exécution).
`omp_get_wtick()`	Renvoie la résolution de `omp_get_wtime()` (temps minimal mesurable).

4. Fonctions de contrôle de l'environnement

Ces fonctions permettent de gérer le comportement d'OpenMP à runtime.

Fonction	Description
`omp_set_dynamic(int)`	Active/désactive l'ajustement dynamique du nombre de threads par OpenMP.
`omp_get_dynamic()`	Indique si l'ajustement dynamique est activé.
`omp_set_nested(int)`	Active/désactive les régions parallèles imbriquées.
`omp_get_nested()`	Indique si les régions parallèles imbriquées sont activées.

> Ce document peut servir de **cheat sheet** pour coder rapidement avec OpenMP et comprendre les directives et fonctions essentielles.