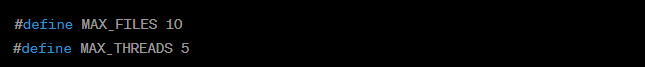


Ces instructions sont des directives de préprocesseur pour inclure les fichiers d'en-tête nécessaires au programme. **stdio.h** est inclus pour pouvoir utiliser les fonctions d'entrée/sortie standard, **windows.h** est inclus pour pouvoir utiliser les fonctions de la bibliothèque Windows, et **time.h** est inclus pour pouvoir utiliser des fonctions de mesure du temps.

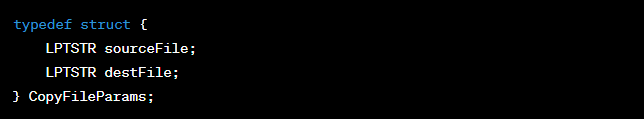


Ces directives définissent deux constantes, **MAX\_FILES** et **MAX\_THREADS**, qui indiquent respectivement le nombre maximal de fichiers à copier et le nombre maximal de threads à créer.



Ceci est le prototype de fonction pour le thread qui effectue la copie de fichier. Il s'agit d'une fonction qui renvoie un type **DWORD** et prend un paramètre de type **LPVOID**. **LPVOID** est un pointeur générique vers n'importe quel type d'objet. Dans ce cas, il sera utilisé pour passer des paramètres à la fonction du thread.

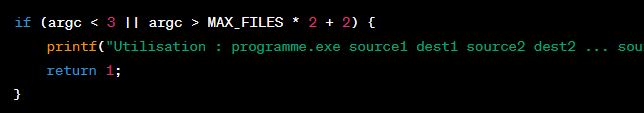
"**WINAPI**" est une convention de déclaration de fonction utilisée dans les applications Windows. Elle spécifie que la fonction suit la convention d'appel standard de l'API Windows.



Ceci est une définition de structure nommée **CopyFileParams** qui contient deux membres, **sourceFile** et **destFile**, qui sont tous les deux de type **LPTSTR**. **LPTSTR** est un alias pour **TCHAR**\*, qui est un pointeur vers un caractère (**char** ou **wchar\_t**) en fonction de si vous compilez le programme en mode Unicode ou ANSI.



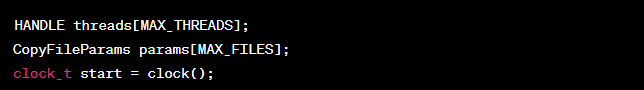
Ceci est la fonction principale du programme. Elle prend deux paramètres, **argc** et **argv**[], qui sont utilisés pour passer des arguments de ligne de commande au programme. **argc** contient le nombre d'arguments passés, et **argv**[] est un tableau de chaînes de caractères contenant les arguments eux-mêmes.



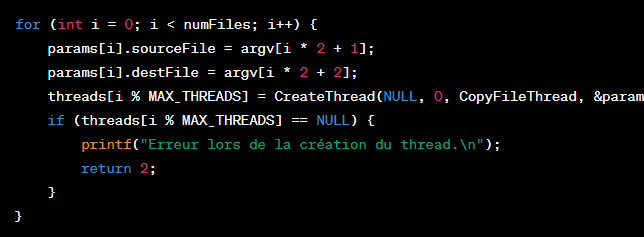
Cette condition vérifie si le nombre d'arguments est inférieur à 3 ou supérieur à deux fois la valeur maximale de fichiers plus deux. Si la condition est vraie, cela signifie que les arguments de ligne de commande sont incorrects. Dans ce cas, un message d'utilisation est affiché et le programme se termine avec un code de retour de 1, indiquant une erreur.



Cette ligne de code calcule le nombre de fichiers à copier en divisant le nombre d'arguments (à l'exception du nom du programme lui-même) par 2. Chaque paire d'arguments consécutifs est considérée comme un nom de fichier source et un nom de fichier de destination.



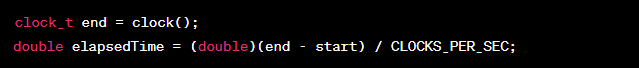
Ces lignes de code déclarent trois variables : un tableau de poignées de thread appelé threads avec une taille maximale égale à **MAX\_THREADS**, un tableau de structures **CopyFileParams** appelé params avec une taille maximale égale à **MAX\_FILES**, et une variable start de type **clock\_t** utilisée pour mesurer le temps d'exécution du programme.



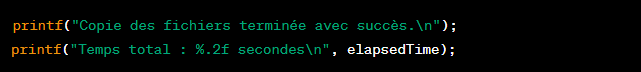
Cette boucle for itère sur chaque paire de fichiers à copier. Pour chaque paire, elle assigne les noms de fichiers source et destination aux membres correspondants de la structure **CopyFileParams**. Ensuite, elle crée un thread en appelant **CreateThread** et passe l'adresse de la structure **CopyFileParams** à la fonction du thread. La poignée du thread est stockée dans le tableau threads. Si la création du thread échoue, un message d'erreur est affiché et le programme se termine avec un code de retour de 2, indiquant une erreur.



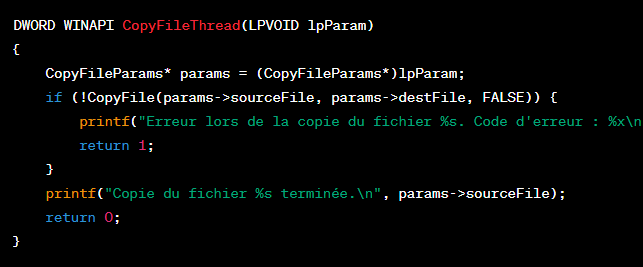
Cette ligne de code attend que tous les threads se terminent. La fonction **WaitForMultipleObjects** bloque l'exécution du programme jusqu'à ce que tous les objets spécifiés (dans ce cas, les poignées de thread) soient dans un état de signalisation. L'argument **numFiles** **< MAX\_THREADS ? numFiles : MAX\_THREADS** est utilisé pour spécifier le nombre d'objets à attendre. Si le nombre de fichiers à copier est inférieur au nombre maximal de threads, il attendra le nombre de fichiers, sinon il attendra le nombre maximal de threads.



Ces lignes de code mesurent le temps écoulé depuis le début de l'exécution du programme jusqu'à ce point. La fonction **clock** renvoie le nombre de pulsations d'horloge écoulées depuis le lancement du programme. En soustrayant la valeur de start de la valeur de end, on obtient le nombre de pulsations d'horloge écoulées. Cette valeur est ensuite divisée par **CLOCKS\_PER\_SEC** pour obtenir le temps écoulé en secondes.



Ces lignes de code affichent un message indiquant que la copie des fichiers est terminée avec succès, suivi du temps total écoulé.



Cette fonction **CopyFileThread** est exécutée par chaque thread. Elle prend le paramètre **lpParam**, qui est un pointeur générique vers une structure **CopyFileParams**. À l'aide d'une conversion de type, le paramètre est casté en un pointeur de type **CopyFileParams**. Ensuite, la fonction utilise la fonction **CopyFile** de la bibliothèque Windows pour copier le fichier source vers le fichier de destination. Si la copie échoue, un message d'erreur est affiché, et la fonction retourne 1. Sinon, un message de succès est affiché et la fonction retourne 0.