Extraits du fichier `mots-cles.c`

```
// Rappel (pour bien comprendre la reduction) du type TableMotsCles,
// defini au debut du fichier.

typedef struct {
  // Nombre effectif de mots cles presents dans la table.
  int nbMotsCles;

  // Les differents mots cles.
  // Valeurs significatives pour [0..nbMotsCles-1].
  MotCle motsCles[MAX_MOTS_CLES];

  // Le nombre d'occurrences des divers mots cles.  Valeurs
  // significatives pour [0..nbMotsCles-1].
  int occurrences[MAX_MOTS_CLES];
} TableMotsCles;
```

```
void coordonnateur( char* fichNomsFichiers, int nbTravailleurs )
{
  NomFichier nomsFichiers[MAX_FICHIERS];  // Les noms des fichiers lus dans le f
  int nbFichiers;                         // Le nombre de fichiers lus, donc a t

  // On charge la liste des noms de fichiers dans le tableau.
  chargerNomsDeFichiers( fichNomsFichiers, nomsFichiers, &nbFichiers );

  // Tant que tous les travailleurs n'ont pas termine...
  int nbTravailleursActifs = nbTravailleurs; // Travailleurs n'ayant pas encore
  int nbNomsFichiersEnvoyes = 0;             // Nombre deja envoyes.

  while ( nbTravailleursActifs > 0 ) {
    MPI_Status statut;
    MPI_Recv( NULL, 0, MPI_BYTE, MPI_ANY_SOURCE, MSG_REQUETE_TACHE,
              MPI_COMM_WORLD, &statut );
    if ( nbNomsFichiersEnvoyes < nbFichiers ) {
      // Il reste des taches : on la transmet.
      MPI_Send( &nomsFichiers[nbNomsFichiersEnvoyes], sizeof(NomFichier), MPI_BY
                statut.MPI_SOURCE, MSG_NOM_FICHIER, MPI_COMM_WORLD );
      nbNomsFichiersEnvoyes += 1;
    } else {
      // Il ne reste plus de taches : on signale au travailleur de terminer.
      MPI_Send( NULL, 0, MPI_BYTE, statut.MPI_SOURCE, MSG_TERMINER, MPI_COMM_WOR
      nbTravailleursActifs -= 1;  // Un de moins est maintenant actif.
    }
  }

  // On recoit la table produite par les travailleurs -- un seul
  // d'entre eux transmet la table resultante.
  TableMotsCles tab;
  MPI_Recv( &tab, sizeof(TableMotsCles), MPI_BYTE,
            MPI_ANY_SOURCE, MSG_TABLE, MPI_COMM_WORLD, MPI_STATUS_IGNORE );

  // On imprime la table des occurrences.
  printf( "*** Contenu de la table des mots cles apres execution ***\n" );
  imprimerMotsCles( &tab );
  printf( "*************************************************************\n" );
  ...
}
```

```
void travailleur( char* fichMotsCles, MPI_Comm comm )
{
  int ID_travailleur;
  MPI_Comm_rank( comm, &ID_travailleur );

  // On charge les mots-cles dans la table et on la diffuse a tous les processus
  TableMotsCles tab;
  if ( ID_travailleur == 0 ) {
    // C'est le premier travailleur qui lit le fichier, puis
    // qui diffuse le contenu de la table aux autres travailleurs.
    chargerMotsCles( fichMotsCles, &tab );
  }
  MPI_Bcast( &tab, sizeof(TableMotsCles), MPI_BYTE, 0, comm );

  // On indique qu'on veut une tache.
  MPI_Send( NULL, 0, MPI_BYTE, 0, MSG_REQUETE_TACHE, MPI_COMM_WORLD );

  for (;;) {
    NomFichier fich;
    MPI_Status statut;
    MPI_Recv( fich, sizeof(NomFichier), MPI_BYTE, 0, MPI_ANY_TAG, MPI_COMM_WORLD

    if ( statut.MPI_TAG == MSG_TERMINER ) break;

    // On envoie une nouvelle requete avant de debuter l'analyse => concurrence!
    MPI_Send( NULL, 0, MPI_BYTE, 0, MSG_REQUETE_TACHE, MPI_COMM_WORLD );

    analyserFichier( ID_travailleur, fich, &tab );
  }

  // On effectue la somme des occurrences pour chaque mot-cle.
  // Solution avec table auxiliaire pour destination du resultat.
  TableMotsCles tabRes = tab;
  MPI_Reduce( tab.occurrences, tabRes.occurrences,
              tab.nbMotsCles, MPI_INT, MPI_SUM, 0, comm );

  // Le travailleur 0 transmet la table finale au coordonnateur.
  if ( ID_travailleur == 0 ) {
    MPI_Send( &tabRes, sizeof(TableMotsCles), MPI_BYTE, 0,
              MSG_TABLE, MPI_COMM_WORLD );
  }
}

//
```

```
////////////////////////////////////////////////////////////////////
// Autre solution possible pour la reduction et envoi du resulat
// au coordonnateur.
////////////////////////////////////////////////////////////////////

  // On effectue la somme des occurrences pour chaque mot-cle.
  MPI_Reduce( ID_travailleur == 0 ? MPI_IN_PLACE : tab.occurrences,
              tab.occurrences,
              tab.nbMotsCles, MPI_INT, MPI_SUM, 0, comm );

  // Le travailleur 0 transmet la table finale au coordonnateur.
  if ( ID_travailleur == 0 ) {
    MPI_Send( &tab, sizeof(TableMotsCles), MPI_BYTE, 0,
              MSG_TABLE, MPI_COMM_WORLD );
  }
}
```