TP 1 : File de tâches

travail à réaliser sur 2 séances (6h)

Les fichiers (pas d'exe, code source uniquement : .c, .h) sont à déposer sur Moodle à la fin de la séance dans la zone de dépôt de votre groupe. Les .zip sont acceptés mais pas les autres types de compressions (pas de .rar ou .tar par exemple).

Vous pouvez travailler en binôme si vous le souhaitez, dans ce cas n'oubliez pas d'indiquer vos 2 noms (vos2noms.zip).

L'objectif est de simuler le fonctionnement d'une file de tâches d'impression, gérées à l'aide d'une liste chaînée double. On utilise les types suivants :

```
typedef char Chaine[15];

typedef struct _Tache {
        Chaine ip; // l'adresse IP de l'expediteur
        Chaine doc; // le document a imprimer represente de facon simplifie
        int taille ; // taille du document en Ko
        struct _Tache * suiv, * prec;
        } Tache, * List ;

typedef struct {
        List debut, fin ;
        } File ;
```

Ecrire les fonctions suivantes, ainsi qu'une fonction `main` pour les tester au fur-et-à-mesure. Rem. : pour les tests, il est plus efficace d'entrer les données à partir d'un fichier qu'au clavier. Cela implique de créer une fonction supplémentaire de lecture dans un fichier (par exemple, une tâche par ligne).

1. Une fonction `creerTache` qui alloue l'espace mémoire pour une `Tache`, y stocke les données passées en paramètre et retourne l'adresse correspondante.
2. Une fonction `ajouterTacheFile` qui permet d'ajouter une tâche en fin de file.
3. Une fonction `afficherIpAttenteFile` d'affichage dans l'ordre (du début vers la fin de la liste) les ip des tâches qui sont dans la file.
4. Une fonction `imprimer` qui simule l'impression du document qui est en début de file : affiche "début d'impression", simule le temps d'impression qui dépend de la taille du document (vous pouvez par exemple demander à l'exécution de "dormir" pendant un certain nombre de secondes...), puis affiche le texte du document et le supprime de la file.
5. Une fonction `afficherEtatFile` qui affiche le nombre de tâches en attente dans la file.
6. Une fonction `supprimerTachesUser` qui supprime de la file toutes les tâches envoyées par l'utilisateur d'une ip donnée.
7. L'adresse ip "192168.144.111" devient prioritaire par rapport aux autres.
   Ecrire une fonction `passerUserPrioritaire` qui fait "remonter" vers le début de la file toutes les tâches provenant de l'ip prioritaire, tout en conservant l'ordre entre les tâches de cet IP.