Fonctionnement

Le code est répartit en trois fichiers : fonctions.c qui contient le code des fonctions utilisées, fonction.h qui contient les déclarations de structures et de fonctions et analyseur.c qui contient le main et qui donne le fichier exécutable.

Les trames, ainsi que les entêtes sont représentées par des structures. La structure Trame contient quatre champs, qui correspondent aux différentes entêtes (Ethernet, IP, TCP et HTTP). Les structures qui représentent les entêtes contiennent autant de champs que les différentes entêtes. (Par exemple, il y a trois champs pour la structure Ethernet car il y a trois champs dans l'entête Ethernet)
Les options du protocole IP et du protocole TCP sont contenues dans une liste de structure option.
La structure option contient le type, la taille et un pointeur vers l'option suivante.

Http n'est pas représentée par une structure mais par une chaîne de caractères.

Comme il peut y avoir plusieurs trames les unes après les autres, il existe aussi une structure Listrame qui contient un numéro, un pointeur sur trame, la ligne du fichier qui correspond au début de la trame, un entier qui indique la ligne ou se situe une erreur dans la trame, ou 0 s'il n'y en a pas, et un pointeur vers l'élément suivant de Listrame.

Le programme se fait en plusieurs étapes. D'abord on cherche à savoir sur combien de caractères est codé l'offset, grâce à la fonction int nb_car_offset(FILE *f, int *cmp_ligne). Puis on cherche à connaître la valeur de l'offset pour savoir le nombre d'octet sur chaque ligne.

On créé la liste de trame contenues dans le fichier Trame.txt grâce à la fonction Listrame *total_liste(FILE *f, int ofst, int nb_car_ofst, int *cmp_ligne). Puis on rempli les éléments de la liste avec rempli_listrame(FILE *f, Listrame *liste, int ofst, int nb_car_ofst, int *curseur, int *offset_ligne, int *cmp_ligne).

Puis on affiche chaque analyse de trame dans le fichier Analyse.txt grâce à la fonction void affiche_listrame(FILE *g, Listrame *liste).

Enfin, on libère la Listrame avec void libere_listrame(Listrame * liste).
Cette fonction libère l'ensemble de la mémoire allouée pour cette Listrame, contenant toutes les structures crées.
Et on affiche « Fin sans soucis » dans le terminal pour indiquer que le programme n'a pas crashé pendant l'exécution.

Pour remplir les éléments de Listrame, on utilise une boucle qui rempli les éléments un à un. A chaque tour de boucle, on se place au début du fichier Trame.txt, puis on saute des lignes jusqu'à la ligne de départ de la trame que l'on veut analyser. Puis on utilise la fonction cree_trame(FILE *f, int ofst, int *curseur, int nb_car_ofst, int *offset_ligne, int * cmp_ligne). Grâce à cette fonction le champ trame de la structure

Listrame ne pointe plus sur NULL mais sur la nouvelle trame. Cette fonction va elle-même appeler des fonctions de remplissage pour chacun de ses champs, en fonction des différentes entêtes de la trame.

Par exemple, pour remplir le champ IP de la trame, on vérifie d'abord si le type du champs Ethernet est bien 0x0800. Puis on appelle IP* lecture_ip (FILE* f, int ofst, int* curseur, int nb_car_ofst, int *offset_ligne, int * cmp_ligne). Cette fonction va récuperer toutes les informations de l'entête IP (options comprises s'il y en a) et va les mettre dans la structure IP de cette trame. Grâce au champ protocol de IP, on peut savoir si le paquet est de type TCP, et donc si le champ TCP de la trame pointera sur NULL.

Avec cette manière de faire on peut facilement ajouter d'autres entêtes analysables.

De même, pour l'affichage des éléments de la trame, on vérifie dans chaque entête quelle entête on doit ensuite afficher.

Pour naviguer dans le fichier, on utilise un curseur global qui permet de savoir où on se situe sur chaque ligne d'octets, et qui nous permet de savoir quand passer à la ligne.
Nous utilisons aussi int saute_ligne(FILE *f, int offset_gen, int *offset_ligne, int nb_car_ofst, int * cmp_ligne) à la fin de chaque ligne. Elle nous permet d'ignorer les potentiels caractères après les octets, les lignes de caractères entre deux lignes d'octets. Cette fonction vérifie aussi que l'offset de la nouvelle ligne d'octets est bien l'offset qui suit celui de la ligne précédente.

Pour l'analyse et l'affichage, nous utilisons aussi des fonctions de traduction, comme int hexa_en_int(char * s), qui prend une chaîne de caractères en hexadécimal et qui renvoie sa valeur en décimal, ou la fonction int traduc (int c) qui prend un caractère, représentant un chiffre hexadécimal et qui renvoie sa valeur en entier décimal.

Pour détecter les erreurs, nous utilisons principalement deux fonctions : int end_of_file(FILE *f) et int end_of_line(FILE *f, int curseur, int ofst), qui détectent si le caractère suivant est la fin du fichier, ou la fin d'une ligne. Pour savoir à quelle ligne a lieu l'erreur, on utilise un compteur de ligne global.