Programmation système P01, rapport individuel David Baschung

Approche du problème

Dans notre approche, nous avons traité les triplets SPO comme un ensemble d'hyperliens dans une base de données. Chaque triplet désigne un lien (Objet) redirigeant vers un certain type de ressource (Prédicat) et étiquetté d'un titre (Sujet). Nous avons donc d'abord créé cette base sous forme d'un tableau de *char* à deux dimensions et avons implémenté les méthodes *insert* et *match*. La fonction *insert* s'assure de la complétion des données fournies. Elle fournit un nouveau triplet et indique par sa valeur de retour si l'insertion est un succès.
La fonction *match* est conçue pour traiter des hyperliens. Elle devait donc être en mesure de trouver les liens en objet d'abord, et de céder ensuite la priorité au type de ressource puis au sujet, justifiant ainsi l'ordre de détection particulier.

Problèmes rencontrés

L'implémentation de base fonctionnait correctement avec un tableau, néanmoins la taille prédéfinie du tableau définissait une limite diminuant la flexibilité de la liste et des chaînes de caractères. Un autre inconvénient lié au tableau s'est également présenté lors du développement de la fonctionnalité supplémentaire *erase*, car supprimer un élément créerait une zone de vide. Il aurait donc fallu décaler tous les éléments en arrière pour ne pas perdre de place en ajoutant un nouvel élément. C'est pour ces raisons que nous avons décidé de transformer notre tableau en linked list.

Dans la version finale de notre projet, nous avons donc implémenté une queue enchaînant une quantité libre de nœuds et parcourable de manière séquentielle. Seuls le premier et dernier élément sont enregistrés. Chaque nœud représente alors une structure de données possédant un contenu (triplet SPO) et la référence vers le prochain nœud. De plus, les triplets sont définis sous forme d'une structure avec des chaînes de caractères illimtées. Cette méthode est donc plus économe et maniable.

Un problème standard supplémentaire a pu être résolu grâce à *malloc*. En créant un triplet, une nouvelle structure est déclarée localement. En utilisant uniquement l'assignation des variables, celle-ci serait détruite à la fin de la fonction. *Malloc* permet de réserver de la mémoire dans le tas et garantit la pérennité de nos données, ainsi que la variabilité des longueurs de chaînes de caractères.

Expérience acquise

La difficulté principale consistait probablement à utiliser *malloc*, les pointeurs et à gérer la taille de l'allocation. La notation est en effet parfois confuse avec le déréférencement et la transmission d'adresses comme paramètres. Ce projet nous a donc permis d'acquérir une maîtrise de base de cet aspect lié au language C.

Un autre élément appris était l'utilisation de git, compliqué à prendre en main mais correctement organisé.