TP C++ n°4 - Analyse de logs Apache : Compte-Rendu :

Le web est basé sur un échange de données entre un client (le navigateur) et un serveur. Le client effectue une requête par l'intermédiaire du protocole HTTP. Cette requête est transmise au serveur qui, en retour, renvoie un code retour et un document.

Toutes ces opérations sont consignées au niveau du serveur web sous la forme d'un fichier journal (*log*) contenant entre autres l'URL cible, l'URL source (*referer*), le type de document renvoyé, la date et l'heure de la requête. Nous avons donc, dans le cadre de ce TP, implémenté un programme permettant d'analyser ces fichiers pour en tirer de l'information.

I. Spécifications :

A. Spécifications générales de l'application :

Notre programme (*main*) est un outil en ligne de commande conçu et réalisé pour analyser un fichier log Apache. Sa principale fonctionnalité est le tri des URL cibles par popularité (nombre de *hits* décroissant) et l'affichage des 10 premiers dans le terminal. Il possède également d'autres options d'analyse comme exclure les documents ayant une extension image, css ou javascript (option *-e*), prendre compte d'un intervalle temporel dans l'analyse du log (option *-t*) ou générer un fichier au format .dot pour générer un GraphViz par la suite (option *-g*) où l'on modélise les différentes URL sources et cibles (noeuds) ainsi que les "passages" des unes aux autres (arcs).

Pour bien fonctionner, notre programme doit obligatoirement connaître le chemin relatif vers le fichier log à analyser. Pour simplifier l'utilisation, nous avons créer un dossier *logs/* où il faut placer les fichiers à analyser. De plus, il faut indiquer si une ou plusieurs option doit être activée. Ainsi, la syntaxe à suivre est la suivante :

./main [options] logName.log

Les options sont celles évoquées ci-dessus :

- *[-g graphName.dot]* : génére le fichier GraphViz avec le nom donné dans l'argument qui suit (ici *graphName.dot*).
- *[-e]* : seules les requêtes dont l'URL cible ne référence pas un fichier css, javascript ou une image sont analysées.
- *[-t heure]* : seules les requêtes dont l'heure de traitement est compris dans l'intervalle [*heure; heure+1*[sont analysées. *heure* ici est un entier dans l'intervalle [0;23].

Ces options peuvent être combinées lors de l'analyse des logs et l'ordre dans lequel elles sont demandées dans la ligne de commande n'importe pas. Cependant, le nom du fichier log doit forcément être en dernière position.

L'analyse du journal Apache peut donc se dérouler de deux façons :

→ *main* est lancé sans options : toutes les requêtes du fichier log seront analysées et le résultat en sortie sera la liste des 10 cibles les plus fréquentes dans le log et le nombre de hits correspondant affichée sur la sortie standard.

→ *main* est utilisé avec au moins une option : seules les requêtes respectant les conditions de sélection seront analysées. La liste des cibles les plus populaires ne contiendra cette fois-ci que les cibles respectant les conditions de sélection avec le nombre de hits correspondant. Enfin, si l'option *-g* est activée, le graphe ne contiendra que les requêtes analysées.

B. Spécifications détaillées :

I.B.1. Passage d'arguments :

Les noms de fichiers ne peuvent contenir que des caractères alphanumériques ou les caractères *'/', '.', '-'* ou *'_'* . De plus, ils ne doivent pas contenir d'espaces.

Cas testé	Résultat attendu	Test(s) correspondant
Nom du fichier log contient un caractère licite non-alphanumérique	"Lancement du traitement" + résultat de l'analyse sur la sortie standard	1
Options passées dans n'importe quel ordre	"Lancement du traitement" + résultat de l'analyse sur la sortie standard	2, 3, 4
Créneau horaire spécifié erroné (n'est pas compris entre 0 et 23)	"Erreur heure indiquée" sur la sortie d'erreurs	5, 6
Créneau horaire spécifié erroné (caractères invalides, ex : lettres)	"Erreur heure indiquée" sur la sortie d'erreurs	7
Pas d'arguments passés au programme	"Arguments manquants" sur la sortie d'erreurs	8
Lancement du programme avec une option inexistante (*-a*) avec un fichier log licite	"Lancement du traitement" + résultat de l'analyse sur la sortie standard	9
Fichier log vide	"Erreur fichier log" sur la sortie d'erreurs	10

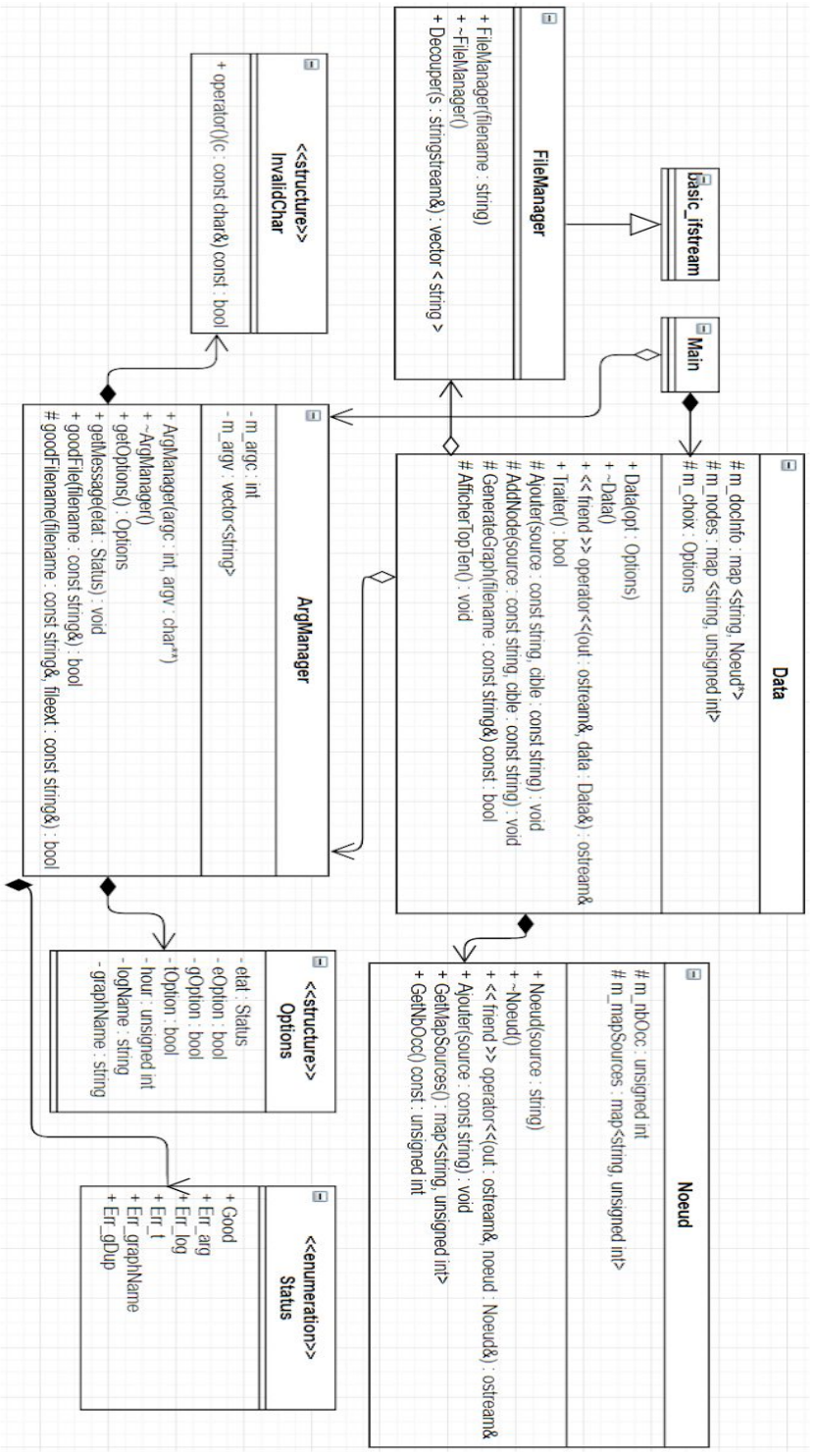
I.B.2. Traitement des URL :

Pour alléger les URL dans les différents résultats, l'outil *main* a été conçu pour supprimer la base de l'URL quand celle-ci est locale, c'est-à-dire lorsque le *referer* contient "*http://intranet-if.insa-lyon.fr/*". Nous avons donc mis en place un test pour vérifier que c'est bien le cas.

Cas testé	Résultat attendu	Test(s) correspondant
Suppression de la base locale du *referer* lorsqu'il est local.	"Lancement du traitement" + résultat de l'analyse sur la sortie standard	11

II. Architecture globale de l'application :

A. Diagramme UML et explications générales :



Sur ce diagramme, il est plus ou moins visible que nous avons décidé d'implémenter une solution qui respecte la notion Modèle-Vue-Contrôleur (MVC) qui répartit assez bien les charges de travail sur les différentes classes de l'application.

B. Classes réutilisables :

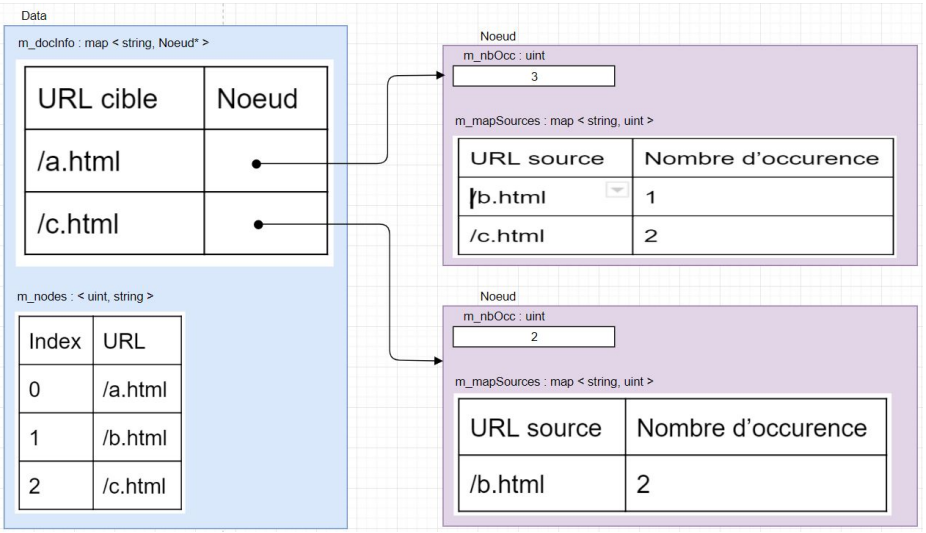
Comme demandé, nous avons essayé de réaliser des classes réutilisables dans la mesure du possible. Ainsi, la classe *FileManager* est entièrement réutilisable pour récupérer les différents éléments d'une requête d'un log Apache étant donné que la majorité de ses méthodes proviennent de son parent : ifstream. De plus, la classe *Noeud* peut également être utilisée pour une autre modélisation. Enfin, la classe *ArgManager* pourrait, avec quelques modifications, être employée pour une autre utilisation. Néanmoins, il semble plus raisonnable d'implémenter une nouvelle classe plus adaptée dans ce dernier cas.

III. Structures de données :

Nous avons décidé d'utiliser des maps et des multimaps pour leur propriété qui étaient parfaites pour notre utilisation et pour sa complexité compétitive pour l'insertion. En effet nous avons une première map Data::m_docInfo qui a pour clé des strings, les sites cibles des requêtes, et pour valeur une instance de la classe Noeud.
La classe Noeud, elle, a pour attribut principal une map Noeud::m_mapSources qui a pour clé des strings et pour valeur un nombre.
Ainsi, avec l'utilisation de la classe Noeud dans la classe Data, il est possible de stocker pour chaque site cible des requêtes, les sites referer ainsi que leur nombre de hit propre à chacun. De plus, il est possible d'obtenir facilement le nombre de hits totals sur un site cible grâce à l'attribut m_nbOcc qui s'incrémente à chaque modification d'un noeud.

Voici ci-dessous un schéma mémoire réalisé à partir du fichier log suivant :

```
1  192.168.0.0 - - [08/Sep/2012:11:16:06 +0200] "GET a.html HTTP/1.1" 200 5192 "b.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1"
2  192.168.0.0 - - [08/Sep/2012:11:16:02 +0200] "GET a.html HTTP/1.1" 200 12106 "c.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1"
3  192.168.0.0 - - [08/Sep/2012:11:16:02 +0200] "GET a.html HTTP/1.1" 200 12106 "c.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1"
4  192.168.0.0 - - [08/Sep/2012:11:16:02 +0200] "GET c.html HTTP/1.1" 200 12106 "b.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1"
5  192.168.0.0 - - [08/Sep/2012:11:16:02 +0200] "GET c.html HTTP/1.1" 200 12106 "b.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1"
```



Pour l'affichage des dix sites les plus populaires, une multimap temporaire est créée à partir de la map m_docInfo. Elle a pour clé le nombre de hit sur chaque site et pour valeur le site associé à ce nombre. La multimap étant triée de façon croissante, il faut la parcourir avec un itérateur inverse pour récupérer les dix derniers éléments qui correspondent alors aux dix sites les plus populaires.

Enfin, la création du fichier dot (pour obtenir un graphique) se fait en utilisant toutes les maps qui ont été précédemment créées. C'est uniquement pour ce but que la map m_nodes a été construite. Elle permet d'associer à tous les sites un index différent grâce à un nombre static et d'ainsi obtenir un fichier dot cohérent. Voici le résultat du fichier dot après l'exécution ./main -g ex.dot ex.log à partir du fichier log présenté plus haut.

```
digraph {
    node0 [label="a.html"];
    node1 [label="b.html"];
    node2 [label="c.html"];
    node1 -> node0 [label="1"];
    node2 -> node0 [label="2"];
    node1 -> node2 [label="2"];
}
```

IV. Performance :

Grâce à la commande time, l'exécution de la commande ./main -g anonyme.dot anonyme.log avec le fichier anonyme.log fourni prend seulement 0,7 secondes pour parcourir tous le fichier, stocker les informations nécessaires et créer un fichier .dot associé, alors qu'il comporte 100 000 lignes.