Baptiste Lotigier
Téo Bouvard

TP C++ : Analyse de logs Apache

1. Introduction

Analog est un programme en ligne de commande permettant d'analyser des logs Apache. Les différentes options disponibles permettent d'affiner l'analyse des logs en filtrant les requêtes. Ce programme permet de synthétiser l'analyse des fichiers log grâce à la création de graphes de parcours et l'affichage des pages les plus consultées.

2. Manuel d'utilisation

Commande :

```
./analog [options] fichier.log
```

Options :

-h
Affiche un résumé du manuel d'utilisation.

-e [extension]
Analyse uniquement les requêtes contenant la chaîne de caractères **extension**. La valeur par défaut de cette option est ".html"

-t heure
Analyse uniquement les requêtes reçues au cours de l'heure **heure**

-x
Analyse uniquement les requêtes dont le referer est différent de "-"

-u host_URL
Modifie l'adresse locale du log. La valeur par défaut est "http://intranet-if.insa-lyon.fr"

-g [dot_file]
Crée un fichier **dot_file** synthétisant l'analyse du log. Par défaut, ce graphe représente les 10 liens les plus suivis (arcs), ainsi que les URL source et destination correspondantes (noeuds)

-n nombre_arcs
À n'utiliser qu'avec l'option -g ! Modifie le nombre d'arcs présents dans les graphe généré par la valeur **nombre_arcs**.

-d [graph_file]
À n'utiliser qu'avec l'option -g ! Permet de convertir le fichier dot_file en fichier SVG **graph_file**

3. Spécifications

a. Spécifications générales

- L'utilisateur peut passer n'importe quel fichier au programme. Ce dernier le traitera ligne par ligne et ignorera toutes celles qui ne correspondent pas au prototype suivant :

  n.n.n.n str str [n/str/n:n:n:n n] "str str str" n n "str" "str"

  (n = nombre et str = chaîne de caractère sans espace)

  Toute les lignes différentes de ce pattern (espaces compris) sont exclues de l'analyse, car le programme n'est pas conçu pour les traiter.

- Avant le début de l'analyse, la présence d'un fichier log dans la ligne de commande est vérifiée, ainsi que la possibilité de son ouverture.

- À chaque appel du programme, les 10 documents les plus consultés du log sont affichés sur la sortie standard. Ils sont triés par nombre de hits puis par ordre alphabétique décroissant en cas d'égalité. Si le fichier log comprend moins de 10 requêtes valides, tous les hits seront affichés.

- Les options utilisées affichent leur appel avant le début de l'analyse.

- Les différentes options peuvent être combinées dans la limite des spécifications des options. Si une option est appelée plusieurs fois avec des paramètres différents, c'est l'appel le plus proche du fichier log qui est pris en compte.

- Les options qui nécessitent un paramètre vérifient la présence et la validité de celui ci, et renvoient un code d'erreur associé si l'une des deux n'est pas vérifiée.

- Dans le cas où l'adresse du referer n'est pas connue ("-" dans le fichier log), le nom utilisé dans le dotfile sera "Undefined".

- Les noms des fichiers générés (options -g et -d) ne peuvent pas contenir d'espace.

- Les différents codes d'erreurs pouvant être renvoyés sont les suivants

  EXIT_SUCCESS : 0
  MISSING_LOG_ARG : 1
  BAD_ARG_LOG_NOT_FOUND : 2
  BAD_ARG_HOUR : 3
  BAD_ARG_HOST : 4
  BAD_ARG_ARCS : 5
  NO_DOT_FILE : 6
  NO_ARG_HOUR : 7
  INVALID_COMMAND : 100

b. Spécifications des options

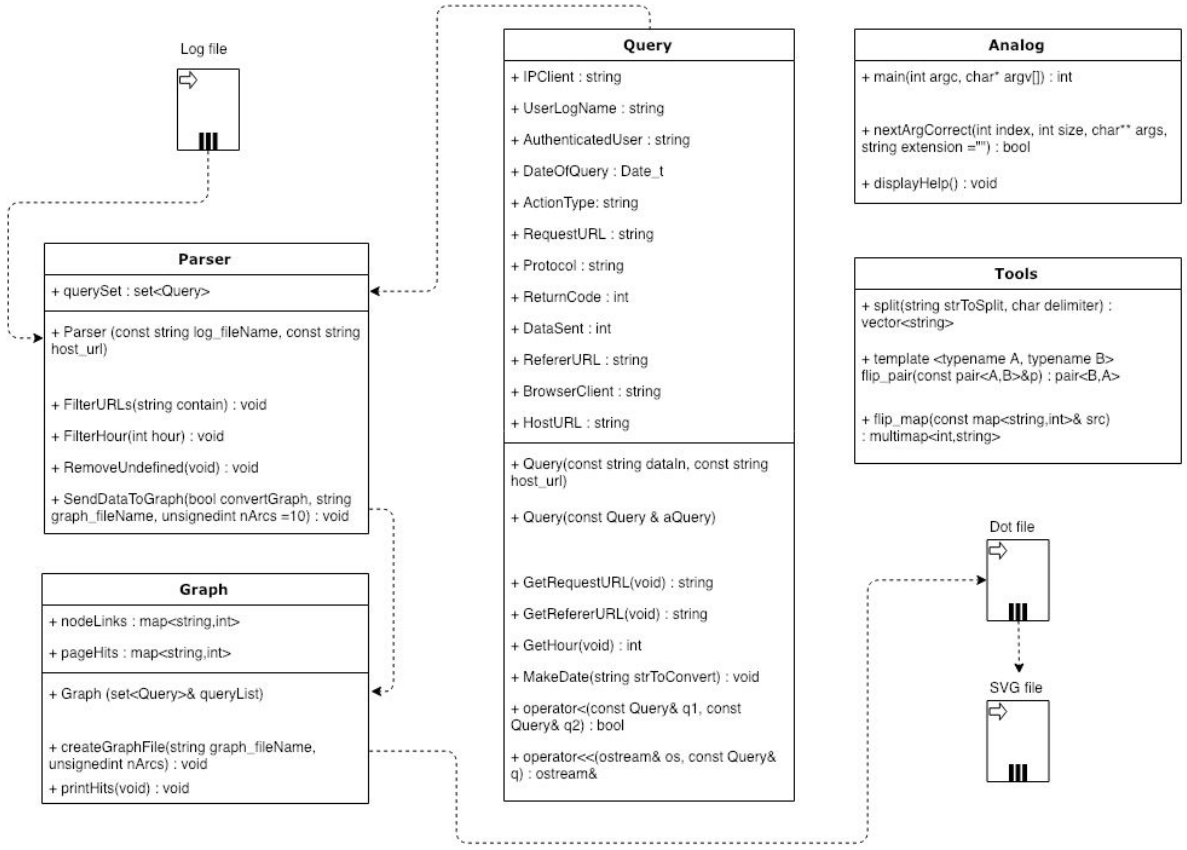
Option	Spécifications	Tests
aucune	L'existence et l'ouverture du fichier sont vérifiées. Seules les lignes formatées selon notre pattern de log sont traitées.	1, 2, 3
-h	Pas d'analyse du fichier log. Si d'autres options sont appelées au préalable, elles affichent leur appel mais ne sont pas exécutées.	-
-e	S'utilise avec 0 ou 1 paramètre. Utilisation avec 0 paramètre : ".html" est utilisé comme filtre. Utilisation avec 1 paramètre : la chaîne de caractères suivant l'appel de l'option (jusqu'au prochain espace) est utilisé comme filtre.	7, 8, 19
-t	S'utilise avec 1 paramètre : entier positif compris entre 0 et 23 inclus. Tout paramètre ne respectant pas cette spécification entraîne le retour du code d'erreur BAD_ARG_HOUR (3) et l'affichage d'un message d'erreur.	9, 10, 19, 20
-x	S'utilise sans paramètre. Filtre les adresses référentes inconnues de la forme "-".	13, 19
-u	S'utilise avec un paramètre : la chaîne de caractères correspondant à l'adresse locale du fichier log qui sera analysé.	17, 18
-g	S'utilise avec 0 ou 1 paramètre : le nom du fichier dotfile de sortie qui sera généré à partir de l'analyse du fichier log. Si aucun paramètre n'est utilisé, le fichier généré sera nommé "out.dot". Si un nom de fichier est utilisé en paramètre, l'extension fournie est vérifiée et, le cas échéant, ajoutée.	4, 5, 6, 12, 19,
-n	S'utilise uniquement avec l'option -g ! Utilise un paramètre : un entier positif correspondant au nombre d'arcs qui seront affichés dans le graphe. Renvoie le code d'erreur BAD_ARG_ARCS (5) en cas de mauvaise utilisation.	14, 15, 16, 19, 21
-d	S'utilise uniquement avec l'option -g ! Le nom du fichier SVG de sortie sera identique au nom du fichier dotfile créé grâce à l'option -d.	11, 12, 19
inexistante	Retour du code d'erreur INVALID_COMMAND(100)	22

4. Conception

a. Fonctionnement du programme

Après avoir vérifié l'existence et l'ouverture du fichier log, la fonction main crée une instance de l'objet Parser. Lors de sa construction, cet objet lit le fichier log et crée une instance de Query pour chaque ligne qui correspond à notre pattern de requête. Tous les objets Query créés sont ajoutés au querySet, attribut de Parser.
Le querySet est ensuite épuré selon les options choisies par l'utilisateur.
Une fois cette liste de requêtes filtrée, elle est utilisée pour construire une instance de Graph, qui réalise le comptage des hits et la construction du graphe de parcours.

b. Diagramme de classe



c. Évolution

Ce programme est conçu pour que l'implémentation de nouvelles fonctionnalités soit facilitée. En effet, les instances de Query sont des objets contenant toutes les informations que possède une ligne d'un fichier log. Nous n'exploitons pas toutes les données stockées dans notre réalisation, mais elles sont accessibles pour de futures mises à jour.

5. Données

Pour notre structure de données, nous avons d'abord réfléchi à comment nous voulions répartir les informations contenues dans chaque ligne du fichier log. Nous en sommes venu à la conclusion qu'il nous fallait créer un objet représentant une requête (élément Query).
Chaque instance de Query pourra alors être insérée dans une liste qui sera un attribut du Parser. De plus, à cette étape de filtrage, nous souhaitons stocker uniquement les Query, sans information supplémentaire. Nous avions pensé utiliser un unordered_set pour implémenter cette liste, mais des tests de vitesse nous on montré qu'il était plus rapide d'utiliser un set avec une fonction de comparaison simple.

En ce qui concerne le tri des requêtes pour l'affichage du top 10 des hits sur les documents, nous avons décidé d'utiliser une structure de données différente. Nous souhaitons ici stocker des attributs de Query (URL) auxquels sont associés un entier, correspondant au nombre de hits ou de parcours du lien. L'implémentation par l'utilisation d'une map permet des tests d'existence rapide des clés (URL) afin de mettre à jour les valeurs (nombre de hits).

Pour réaliser l'affichage sur la sortie standard et dans le fichier graphe, nous inversons les clés et les valeurs de ces maps afin de les ordonner par nombre de hits et plus par URL.
Il nous suffit alors de récupérer les 10 derniers éléments d'une multimap afin d'afficher les documents les plus consultés. Cette implémentation est réalisée dans la classe Graph, et utilise les méthodes génériques du module Tools.