Compilation

TP 1

Dans le fichier `/etc/network/interfaces`, on définit...des interfaces !

Une interface va consister en une adresse IP, un masque de sous-réseau, une adresse de diffusion et une adresse de réseau. Ces informations vont être à spécifier par l'utilisateur ou détectées automatiquement.

— L'interface **loopback**
Le premier interlocuteur de la machine, c'est elle-même. Et pour se parler à elle-même, elle peut utiliser tout simplement le réseau. On déclare l'interface comme suit `iface lo inet loopback`

— L'interface **dhcp**
Lorsque l'interface obtient sa configuration grâce à un serveur dhcp, on déclare l'interface comme ça : `iface mon_interface inet dhcp`
Par défaut, quand l'interface se montera, elle va envoyer une requête dhcp, le serveur lui répondra en lui donnant une IP, un masque de sous-réseau, une adresse de diffusion, une passerelle et des serveurs DNS.

— L'interface **static**
Parfois il n'est pas pertinent ou pas possible de se servir d'un serveur DHCP. Soit parce que c'est trop lent, soit parce qu'il n'y en a pas, soit parce qu'il ne sait pas donner des adresses fixes à ses clients, ...
Voici un exemple de configuration pour un réseau local

```
iface maison inet static
  address 192.168.0.1
  netmask 255.255.255.0
  gateway 192.168.0.254
```

S'il n'est pas forcément souhaitable de démarrer automatiquement une interface filaire au démarrage quand celle-ci ne sert que rarement, il est en revanche nécessaire de démarrer certaines interfaces avec le système. Le démarage automatique se fait de la façon suivante : `auto mon_interface`

Un fichier `/etc/network/interfaces` respecte donc une grammaire de la forme :

1. $IS \rightarrow I\ IS$
2. $IS \rightarrow \Lambda$
3. $I \rightarrow auto\ id$
4. $I \rightarrow iface\ id\ inet\ T$
5. $T \rightarrow loopback$
6. $T \rightarrow dhcp$
7. $T \rightarrow static\ address\ ip\ netmask\ ip\ gateway\ ip$

Un exemple de fichier est donc :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

iface maison inet static
  address 192.168.0.1
  netmask 255.255.255.0
  gateway 192.168.0.254
```

Exercice 1 : Analyse lexicale

1. Donner les expressions régulières pour les noms d'interface et les adresses IP ;
2. Réaliser un analyseur lexical pour les fichiers `/etc/network/interfaces` avec l'outil `jflex` (`http://jflex.de/`) ;
3. Ecrire un programme de test qui affiche le nom de l'unité lexicale et le texte reconnu ;
4. Ecrire différents fichiers tests pour l'analyseur ainsi réalisé.

Exercice 2 : Analyse syntaxique Nous souhaitons maintenant réaliser un analyseur syntaxique en utilisant le langage Java. Pour ce faire, nous aurons besoin (au minimum) :

— d'une `Lexer` qui permettra, entre autre d'accepter un non terminal ;
— d'une classe abstraite `NonTerminal` ;
— pour chaque non terminal X, une classe NT_X contenant une méthode analyser() qui sélectionne, selon le symbole courant du flux d'entrée et les symboles directeurs calculés pour cette grammaire, la bonne règle d'analyse (méthodes regleNN).

1. Calculer les symboles directeurs des règles de la grammaire ;
2. Réaliser un analyseur syntaxique, respectant le principe de l'ADR, et les contraintes ci-dessus, en utilisant le langage Java ;
3. Ecrire un programme de test qui affiche le nom de l'unité lexicale et le texte reconnu ;
4. Ecrire différents fichiers tests pour l'analyseur ainsi réalisé.