Les éditeurs de texte standards

Le programme `vi` est un éditeur de texte *visuel* se basant sur l'éditeur de texte `ed` . Il est apparu dans les premières versions des implémentations d'Unix faites par l'Université de Californie à Berkeley [1].

L'éditeur de texte `vi` est loin d'être un éditeur très évolué et encore moins convivial [2]. Néanmoins, comme `ed` , c'est désormais un standard sur toutes les versions d'UNIX, c'est-à-dire que quelque soit la version d'UNIX que vous utiliserez la commande `vi` sera présente. Il est très puissant et ne nécessite que très peu de ressources.

La majorité des directives de `vi` et de `ed` sont utilisables dans d'autres commandes UNIX comme par exemple `more` , `sed` , `man` , etc. Toutes ces commandes sont elles-aussi basées sur l'éditeur en ligne `ed` .

Il existe aujourd'hui un grand nombre d'implémentation de `vi` toutes différentes, cependant elles respectent un fonctionnement minimum commun que nous allons étudier dans ce TP et qui correspond aux premières versions historiques de `vi` . Pour plus de détails sur `vi` vous pourrez vous reporter à la *Vi Lovers Home Page* à l'adresse suivante : `http://www.thomer.com/vi/vi.html`

L'éditeur de texte en ligne `ed`

`ed` [3] est un programme rudimentaire permettant de créer, voir ou éditer des fichiers textes, c'est-à-dire des fichiers contenant des lignes de caractères terminées par un passage à la ligne. `ed` ne sait manipuler que les lignes de tels fichiers. Il fonctionne sur tous les terminaux possédant un canal d'entrée et un canal de sortie sans aucune autres contraintes. C'est un des plus anciens éditeurs de textes.

`ed` peut être appelé avec ou sans argument. Si un argument est passé à `ed` il considèrera que cet argument est un chemin valide vers un fichier texte. Dans ce cas il lira le contenu du fichier et le copiera en mémoire afin de pouvoir l'utiliser. Si le fichier n'existe pas un message d'erreur sera envoyé mais le chemin sera conservé pour les opérations suivantes. Aucun enregistrement des modifications faites sur le contenu en mémoire ne sera effectué sans une demande explicite de l'utilisateur.

`ed` possède 2 modes de fonctionnement :
– le mode **commande**, les caractères arrivant à `ed` par le canal d'entrée du terminal sont considérés comme des ordres de manipulation du contenu en cours d'édition ;
– le mode **saisie**, les caractères arrivant à `ed` par le canal d'entrée du terminal sont considérés comme devant être ajoutés au contenu qui est en mémoire.

Lors de son démarrage `ed` est en mode **commande**. Le passage en mode de **saisie** se fait après réception de certaines commande (`a` , `i` , `c` par exemple). Pour quitter le mode de **saisie** il suffit d'envoyer une ligne ne contenant que le caractère « `.` ».

Une commande `ed` peut être préfixée par des caractères désignant une ligne, ou un bloc de lignes, sur laquelle la commande doit être exécutée. On appelle cela l'**adresse** sur laquelle la commande doit s'appliquer. Certains caractères ou mots désignent des lignes spécifiques :
– un nombre *n* correspond à la *n*ième ligne en mémoire. La première ligne en mémoire est la ligne 1.
– « `.` » désigne la ligne courante (c'est-à-dire la dernière ligne sur laquelle a eu lieu une opération).
– « `\$` » désigne la dernière ligne du contenu en mémoire.
– `/motif/` correspond à la prochaine ligne contenant le mot *motif* (à partir de la ligne courante).
– `?motif?` correspond à la précédente ligne contenant le mot *motif* (à partir de la ligne courante).

Un bloc de lignes est représenté par la ligne de départ et la ligne de fin séparée par le caractère « `,` ». Le bloc `1,\$` représente par exemple l'ensemble du contenu en mémoire.

Quand `ed` reçoit une commande qu'il ne comprends pas il envoie sur le canal de sortie du terminal le caractère « `?` ».

Le tableau 1 rassemble un certain nombre des commandes `ed` importantes. La page du manuel de `ed` vous en dira plus.

1. Ces implémentations ont données naissance à la famille des Unix BSD. Pour un historique complet des implémentations d'UNIX reportez-vous à l'adresse `http://www.levenez.com/unix`
2. ...pour les utilisateurs habitués à utiliser plus souvent une souris que leur cerveau. Mais tout ça n'est qu'une question d'habitude.
3. Souvent remplacé par `ex` dans les implémentations récentes d'UNIX

Exercice 1 : Utilisation de base de `ed`

Q 1. Connectez-vous, en mode graphique puis ouvrez un émulateur de terminal.

Q 2. Assurez-vous d'être que votre catalogue de travail soit le dossier `asr3/unix02` (créez le si nécessaire).

Q 3. Lisez[4] complètement la page du manuel de `ed` en passant les sections `OPTIONS` et `REGULAR EXPRESSIONS`.

Q 4. Créez maintenant un fichier nommé `information` contenant 3 lignes puis quittez `ed`. La première ligne du fichier doit contenir votre nom, la seconde votre prénom, la dernière votre groupe.

Q 5. Depuis le shell vérifiez que le contenu de votre fichier correspond bien à ce que vous avez saisi.

Q 6. Ouvrez le fichier `information` avec `ed` et faites afficher le contenu du fichier avec une numérotation des lignes.

L'éditeur de texte visuel `vi`

Contrairement aux outils en mode ligne qui n'utilisent que le minimum des capacités offertes par les terminaux (la ligne), `vi` est dit *visuel* car il représente le contenu d'un fichier texte en utilisant toute la zone d'affichage offerte par certains terminaux, comme les écrans, dans lequel il est utilisé. Une partie de la mémoire, correspondant au contenu du fichier, est donc constamment visible à l'écran. La partie visible dépend de la taille de l'écran du terminal.

Comme `ed`, `vi` est un éditeur de texte fonctionnant grâce à des modes. Il existe deux modes d'utilisation sous `vi` :

1. le mode **commande**[5] : au démarrage de `vi` vous êtes toujours dans ce mode. Une commande c'est un ou plusieurs caractères qui permettent de demander un traitement particulier (déplacement, recherche, remplacement, etc.). Il y a 2 types de commandes :
   – les *commandes* `vi` sont utilisées en appuyant sur la (ou les) touche(s) leur correspondant
   – les *commandes* `ed` sont données sur la dernière ligne de votre terminal. Pour cela vous devez :
     (a) utiliser la commande `vi` « `:` ». Le curseur se positionne alors sur la dernière ligne de votre terminal juste après le caractère « `:` ».
     (b) saisir votre commande `ed`.
     (c) valider la requête pour la faire s'exécuter en appuyant sur la touche [entrée].
2. le mode **saisie** : il existe plusieurs commandes `vi` permettant de passer dans ce mode. Le seul moyen d'en sortir (de s'en *échapper*) est d'appuyer sur la touche d'échappement : [échap]. Comme son nom l'indique ce mode permet de saisir du texte. La différence avec `ed` est que le texte saisi apparaît directement à l'écran à l'endroit où il est ajouté.

La *position d'édition*, ou *position courante*, dans le fichier est repérée par le *curseur* (un caractère représentant un bloc noir ou un trait clignotant). Le caractère se trouvant sous le curseur est appelé le *caractère courant*, il correspond à un caractère du fichier. Pour déplacer le curseur et donc la position d'édition il existe 4 commandes `vi` particulières :

1. « `h` » déplace le curseur d'un caractère vers la gauche
2. « `l` » déplace le curseur d'un caractère vers la droite
3. « `j` » déplace le curseur d'une ligne vers le bas
4. « `k` » déplace le curseur d'une ligne vers le haut

Ces commandes `vi` fonctionnent quelque soit le terminal utilisé. Cependant sur les micro-ordinateurs des salles de TP vous pouvez de manière plus simple utiliser les flèches de votre clavier pour effectuer ces déplacements, respectivement [←], [→], [↓] et [↑].

Dans les descriptions de commandes et exemples du reste de ce TP les mots en *emphases* représentent des informations que vous devez fournir.

– Pour démarrer l'édition d'un fichier il suffit de taper la commande : `vi fichier`.
– Pour sauvegarder un fichier une fois dans `vi` il vous faut utiliser la commande `:w`
– Pour quitter l'éditeur :
  – si vous n'avez pas modifié le fichier il suffit d'utiliser la requête `:q`
  – si vous avez modifié le fichier mais que vous ne désiriez pas sauvegarder les modifications il vous faut utiliser la commande `:q!`
  – si vous avez modifié le fichier et que vous voulez sauvegarder ces modifications il vous faut utiliser l'une des requêtes `:x`, `:wq` ou simplement `ZZ`.

4. Ici, lire veut dire lire **et** comprendre
5. On dira aussi **directive**

Le tableau 2 rassemble un certain nombre des commandes `vi` importantes. La page du manuel de `vi` vous en dira plus.

Il est possible de répéter l'éxécution d'une commande `vi` en la préfixant par le nombre de fois qu'elle doit être éxécutée. Par exemple pour effacer 10 lignes il suffit de taper `10dd`.

Lorsqu'il vous semble que vous ne vous souvenez pas dans quelle mode vous êtes et que vous êtes perdu un bon réflexe est d'appuyer plusieurs fois sur la touche échap jusqu'à ce que vous entendiez un bip sonore signalant que vous êtes de nouveau en mode commande (`vi` ne connaissant pas de directive liée à la touche échap vous signale une erreur en émettant un bip sonore).

Commandes `ed`	
`u`	Annuler la dernière commande.
n	Fixer la ligne courante comme étant la ligne *n*
`(.,.)p`	Afficher les lignes spécifiées.
`(.,.)n`	Afficher les lignes spécifiées en les numérotant.
`(.)a`	Ajouter le texte saisi après la ligne spécifiée. *fait entrer en mode saisie*
`(.,.)c`	Supprimer les lignes spécifiées et remplace les par le texte saisi. *fait entrer en mode saisie*
`(.,.)d`	Supprimer les lignes spécifiées.
`(.)i`	Ajouter le texte saisi avant la ligne spécifiée. *fait entrer en mode saisie*
`(.,.)m(.)`	Déplacer les lignes spécifiées après la ligne dont la spécification suit le caractère `m`.
`(.,.)s/`*motif1*`/`*motif2*`/g`	Remplacer toutes les occurences de *motif1* par *motif2* sur les lignes spécifiées.
`(.,.)t(.)`	Copier les lignes spécifiées après la ligne dont la spécification suit le caractère `t`.
`(.,.)y`	Copier dans un tampon les lignes spécifiées.
`(.)x`	Coller le contenu du tampon après la ligne spécifiée.
`e` *chemin*	Lire le fichier *chemin* et copier son contenu en mémoire.
`($)r` *chemin*	Lire le fichier *chemin* et insérer son contenu après la ligne spécifiée.
`(1,$)w`	Enregistrer les lignes spécifiées dans le fichier courant.
`(1,$)w` *chemin*	Enregistrer les lignes spécifiées dans le fichier *chemin*.
`!` *commande*	Exécute *commande* dans un shell.
`q`	Quitter `ed`

Table 1 – **Commandes** `ed`. Les commandes sont montrées ici avec leur adresse d'application par défaut. Ces adresses par défaut sont celles que la commande considère si aucune adresse n'est spécifiée. Ces valeurs par défaut sont placées entre parenthèses uniquement pour simplifier leur identification. Les parenthèses ne doivent pas être saisies si une adresse est spécifiée. Les mots en *emphases* doivent être remplacés par les valeurs voulues.

Commandes `vi`	
`0`	Positionner le curseur en début de ligne
`$`	Positionner le curseur en fin de ligne
`:0`	Positionner le curseur sur la première ligne du fichier
`:$`	Positionner le curseur sur la dernière ligne du fichier
`u`	Annuler la dernière commande
`.`	Répéter la dernière commande
`i`	Passer en mode de saisie (insertion avant le curseur)
`I`	Passer en mode de saisie (insertion avant la ligne courante)
`a`	Passer en mode de saisie (ajout après le curseur)
`A`	Passer en mode de saisie (ajout après la ligne courante)
`yy`	Copier la ligne courante dans un tampon
`P`	Coller le contenu du tampon avant la ligne courante
`p`	Coller le contenu du tampon après la ligne courante
`x`	Supprimer le caractère courant
`dw`	Supprimer le mot courant
`dd`	Supprimer la ligne courante
`ZZ`	Quitter après avoir sauvegarder les modifications
`/`*motif*	Rechercher *chaine*
`n`	Répéter la dernière recherche effectué à partir de la position courante
`:`*commande*	Effectuer la *commande* `ed` spécifiée
`:set` *option*	Fixer l'*option* de l'éditeur (cf table 3)
`:set no`*option*	Enlever l'*option* de l'éditeur (cf table 3)

Table 2 – **Commandes** `vi`. Les mots en *emphases* doivent être remplacés par les valeurs voulues. La plupart des commandes peuvent être préfixées par le nombre de fois qu'elles doivent être exécutées.

Options de `vi`	
`number`	Afficher les numéros de lignes devant chaque ligne.
`ignorecase`	Ne pas faire de distinction entre les majuscules et les miniscules pour les recherches.
`autoindent`	Indenter automatiquement les programmes en fonction du langage de programmation utilisée.

Table 3 – **Options de l'éditeur** `vi`

Exercice 2 : Entraînement forcé

Q 1. Démarrez l'éditeur `ed`.

Q 2. Insérez le texte suivant, **en respectant les longueurs des lignes** et **en ne saisissant pas les numéros de lignes** :

1905.txt

```
Extraits de la loi de 1905 concernant la séparation des Églises et de l'État :

Art. 26.- Il est interdit de tenir des réunions politiques dans les locaux servant habituellement à
l'exercice d'un culte.

Article premier. - La République assure la liberté de conscience. Elle
garantit le libre exercice des cultes sous les seules restrictions
édictées ci-après dans l'intérêt de l'ordre public.

Art. 28.- Il est interdit, à l'avenir, d'élever ou d'apposer aucun
signe ou emblème religieux sur les monuments publics ou en quelque
emplacement public que ce soit, à l'exception des édifices servant au
culte, des terrains de sépulture dans les cimetières, des monuments
funéraires, ainsi que des musées ou expositions.

Art. 35.- Si un discours prononcé ou un écrit affiché ou distribué
publiquement dans les lieux où s'exerce le culte, contient une
provocation directe à résister à l'exécution des lois ou aux actes
légaux de l'autorité publique, ou s'il tend à soulever ou à armer une
partie des citoyens contre les autres, le ministre du culte qui s'en
sera rendu coupable sera puni d'un emprisonnement [...].

Extraits d'un éditorial du dessinateur Siné :

Bienvenue dans les lieux de culte, dans les usines à décerveler, à
enconner, églises, temples, pagodes, mosquées, synagogues...  Portes
ouvertes à la bêtise, à l'ignorance, à l'obscurantisme, à l'ineptie, à
l'abrutissement...

Comme sur les paquets de clopes où figure maintenant en gros "FUMER
TUE", on devrait inscrire, en énorme, sur les couvertures de la bible,
de la thora et du coran : "LA RELIGION REND CON".
```

Q 3. Sans quitter `ed`, vérifiez en le faisant afficher que le texte que vous avez saisi est bien en mémoire

Q 4. Sans quitter `ed`, sauvegardez le contenu dans un fichier de nom `1905.txt`.

Q 5. Sans quitter `ed`, insérez le contenu du fichier `/home/public/asr3/1905`, **après** la ligne 15.

Q 6. Sans quitter `ed`, déplacez les extraits de l'éditorial de Siné (de la ligne 34 à la dernière ligne) en début de fichier.

Q 7. Sans quitter `ed`, insérez le texte suivant entre l'avant dernière et la dernière ligne :
`L'anarchiste est celui qui a un tel besoin d'ordre qu'il n'en admet aucune parodie. Antonin Artaud.`

Q 8. Sans quitter `ed`, vérifiez que toutes les modifications que vous avez faites sont bien en mémoire en faisant afficher le contenu de la mémoire.

Q 9. Quitter `ed` en enregistrant vos modifications.

Q 10. Ouvrez votre fichier en utilisant `vi`.

Q 11. Insérez vos noms et prénoms en tête du fichier sur une seule ligne.

Q 12. Copiez l'article 26 en début de fichier 20 fois.

Q 13. Utilisez la commande de substitution pour remplacer partout dans le texte `culte` par `cuculte`.

Q 14. Ajoutez en fin de fichier le contenu du fichier `information` que vous avez créé dans le premier exercice.

Q 15. Utilisez la commande de substitution pour remplacer partout dans le texte `religi` par `/piege/ /a/ /`.

Q 16. Comptez en utilisant les commandes de recherche de chaîne le nombre de lignes contenant une ou plusieurs occurrence du mot `les`.

Q 17. Appelez la commande `man vi` sans quitter `vi`.

Q 18. Quittez l'éditeur en sauvegardant le contenu.

Commandes	Utilité
`pwd`	afficher le nom du répertoire de travail
`cd`	changer de répertoire de travail
`ls`	lister des fichiers
`rm`	supprimer un fichier
`mkdir`	créer un répertoire
`rmdir`	supprimer un répertoire
`cp`	copier un fichier
`mv`	déplacer un fichier
`ln`	surnommer (créer un *lien* vers) un fichier
`cat`	afficher le contenu de fichiers
`stat`	afficher les caractéristiques de fichiers
`file`	deviner la nature du contenu de fichiers

Table 4 – Commandes de manipulations du système de fichiers

Système de fichiers: Principes généraux

Sous UNIX quasiment tout est accessible grâce à la notion de fichier : des données stockées sur le disque dur jusqu'aux périphériques de l'ordinateur. La notion de fichier est donc centrale dans le fonctionnement du système d'exploitation.

Du point de vue du cœur du système, un fichier est simplement une suite d'informations (ses *caractéristiques*) et de données non structurée (son *contenu*) stockée dans une table du système. Il est donc repérable de manière unique par son index dans cette table (son *inode*).

Le sous-système de gestion des fichiers d'UNIX permet à l'utilisateur d'avoir une vue arborescente du rangement de ceux-ci. La base de la hiérarchie de tout système UNIX est notée : « `/` ». On appelle cela la *racine* (*root* en anglais) de la hiérarchie. Afin de pouvoir gérer cette hiérarchie il existe un type de fichier particulier : *les répertoires*.

Le contenu d'un répertoire est simplement une liste d'inode repéré par un nom. Plus simplement on dira une liste de fichiers. Un répertoire est aussi appelé un *catalogue*, ou un *dossier*. En anglais on dira plus souvent *directory*. On considère donc que ces fichiers ont la propriété de pouvoir *contenir* autant de fichiers que voulu, **y compris des répertoires**. Un fichier est donc vu par l'utilisateur comme une entrée dans un catalogue, ou encore un lien vers une inode.

Les noms de fichiers peuvent généralement comporter entre 1 et 1024 caractères. UNIX fait une différence entre les lettres en majuscules et les lettres en minuscules, de telle sorte que `toto` et `Toto` représentent deux fichiers différents.

Les fichiers dont le premier caractère du nom est le caractère « `.` » sont considérés comme des *fichiers cachés* : la commande `ls` ne les montrera qu'avec une option particulière.

Tous les fichiers sont identifiables par leur nom et leur emplacement dans la hiérarchie. On doit pour cela fournir la liste des répertoires qu'il faut traverser avant de parvenir dans le répertoire contenant le fichier. On appelle cette information le *chemin* (*path* en anglais) du fichier. Chaque répertoire d'un chemin est séparé du répertoire suivant par le caractère « `/` ». Il existe deux manières de donner un chemin :
– en spécifiant la liste **la plus courte** des répertoires à traverser **depuis la racine de la hiérarchie**, on dit alors que c'est un chemin **absolu** ;
– en spécifiant **une** liste des répertoires à traverser à partir d'un répertoire particulier de la hiérarchie, on dit dans ce cas que c'est un chemin **relatif à ce répertoire de départ**.
Il existe des commandes, accessibles via le shell, permettant de se déplacer ou de modifier la hiérarchie. Le tableau 4 donne une liste de certaines de ces commandes. Les pages du manuel en ligne vont donneront des détails sur l'utilisation de ces commandes.

Utilisation

La commmande `cp chemin1 chemin2` :
– crée le fichier `chemin2` si celui-ci n'existe pas,
– sinon elle écrase les données existantes dans `chemin2` par celles de `chemin1`.
Dans le premier cas, il s'agit d'une création de fichier, dans le deuxième cas, il s'agit d'une modification.

La commmande `mv chemin1 chemin2` :

1. supprime l'entrée `chemin1` du répertoire dans lequel ce fichier se trouve,
2. – crée l'entrée `chemin2` avec la même inode que l'entrée `chemin1` si elle n'existe pas,
   – sinon elle écrase les données existantes dans `chemin2` en les remplaçant par celles de `chemin1`.

La commmande `ln chemin1 chemin2` :
– crée l'entrée `chemin2` avec la même inode que l'entrée `chemin1` si elle n'existe pas.

Les droits

Chaque fichier appartient à un utilisateur, son *propriétaire*, et est rattaché à un ensemble d'utilisateur, son *groupe*. À chaque fichier sont attachés des droits d'accès qui spécifient pour chacune des catégories d'utilisateur (le propriétaire, les membres du groupe du fichier ou les autres utilisateurs) les manipulations possibles sur le fichier.

Pour chacune de ces catégories d'utilisateurs ce mode d'accès spécifie trois droits :

1. le droit de *lecture* du contenu du fichier (droit `r`)
2. le droit de *modification* du contenu du fichier (droit `w`)
3. pour les répertoires, le droit de *franchissement* du répertoire, c'est-à-dire le droit d'utiliser le répertoire dans un chemin ; pour les autres fichiers le droit d'*exécuter* le contenu du fichier, c'est-à-dire le droit de demander au système de considérer le contenu du fichier comme des instructions à exécuter (droit `x`).

Il est à noter que :

1. Pour modifier les droits d'un fichier, il faut en être le propriétaire ou être l'administrateur du système (utilisateur nommé `root`, d'UID 0).
2. Pour accéder à un fichier, il faut avoir le droit de franchissement de (*passage dans*) chacun des répertoires qui constituent son chemin.
3. Pour écrire dans un fichier, il faut avoir l'autorisation d'écriture sur ce fichier.
4. Pour créer, détruire, déplacer, renommer, ou surnommer un fichier, il faut avoir le droit d'écriture dans le **répertoire** contenant ce fichier, puisqu'il s'agit d'ajouter ou de supprimer un lien dans un répertoire (une entrée dans le catalogue).

Modification

La commande `chmod` permet de modifier les droits d'accès d'un fichier :

`chmod` *mode_d'accès chemins*

En version symbolique le mode est composé de la manière suivante :

`<personne><action><accès>`

Personne	
`u`	propriétaire
`g`	groupe
`o`	autres
`a`	tous

Action	
`+`	ajouter
`-`	enlever
`=`	initialiser

Accès	
`r`	lecture
`w`	écriture
`x`	éxécution/franchissement

En version numérique il est composé de 3 chiffres octaux (*i.e.* en base 8, donc des chiffres entre 0 et 7) représentant les droits pour :

1. le propriétaire du fichier
2. les membres du groupe du fichier
3. les autres utilisateurs

Les chiffres octaux correspondent aux droits à attribuer pour chacune de ces trois catégories d'attribution :

Nombre octal	Nombre binaire	Droits équivalents
0	000	`---`
1	001	`--x`
2	010	`-w-`
3	011	`-wx`
4	100	`r--`
5	101	`r-x`
6	110	`rw-`
7	111	`rwx`

Exercices

Exercice 3 : Chemins absolus ou relatifs/Caractéristiques

Soit la hiérarchie UNIX représentée par la figure 1.

Q 1. Donnez le nom absolu du premier fichier de chaque répertoire de la hiérarchie.

Q 2. Remplissez la colonne `Type` de la figure 1, en spécifiant pour chaque fichier s'il s'agit d'un répertoire, d'un lien symbolique ou d'un fichier régulier.

Dans la suite de l'exercice on considère que tous les fichiers qui ne sont pas, avec certitude, des catalogues sont des fichiers réguliers. Pour simplifier on considère également que le contenu des fichiers réguliers est simplement le nom de l'animal représenté, par exemple le fichier `chacal` contient le mot `chacal`.

Q 3. Pour chacun des noms suivants spécifiez s'il correspond à un chemin relatif correct par rapport au répertoire spécifié dans la colonne.

nom	**relatif à** `/reptiles/avec_pattes`	**relatif à** `/oiseaux`
`../sans_pattes/cobra`		
`../../mammiferes`		
`/mammiferes/cetaces/dauphin`		
`../reptiles/avec_pattes/caiman`		
`../oiseaux/canides/chacal`		
`./caiman/../crocodile`		
`../sans_pattes/../../oiseaux/Autruche`		
`../.`		
`/oiseaux/avec_pattes/autruche`		
`../oiseaux/autruche`		

Q 4. Donnez le type, l'inode et le contenu de chacun des fichiers suivants :
- `chien`
- `reptiles`
- `avec_pattes`

Exercice 4 : Manipulation du système de fichiers

Q 1. En considérant que **le répertoire courant est la racine de l'arbre**, que vous **n'avez pas le droit d'utiliser la commande** `cd` et que **vous ne devez utiliser que des chemins relatifs**, donnez pour chacune des phrases suivantes la (ou les) commande(s) à exécuter pour effectuer l'action demandée :

1. faire en sorte que le fichier nommé `chacal` s'appelle également `hyene` dans le répertoire `mammiferes`
2. créer un répertoire nommé `a_plumes` dans le répertoire `oiseaux`
3. effacer le fichier nommé `chacal`
4. déplacer le fichier nommé `pingouin` à la racine de l'arbre
5. copier le fichier nommé `tortue` dans le répertoire nommé `mammiferes`
6. déplacer le fichier créé par la commande précédente dans le répertoire racine en le nommant `pingouin`
7. déplacer le fichier nommé `pingouin` de la racine vers le répertoire nommé `avec_pattes` en le nommant `tortue`
8. copier le contenu du fichier nommé `loup` dans le fichier nommé `homme`.

Q 2. Donnez les numéros d'inodes de tous les fichiers de la hiérarchie après exécution de toutes les commandes de la question précédente.

Q 3. Donnez le contenu des fichiers suivants :
- `mammiferes/hyene`
- `mammiferes/singes/homme`
- `reptiles/avec_pattes/tortue`

Exercice 5 : Mode symbolique - mode octal

Soit le fichier suivant :

```
-rw-rw-r-- 1 toto iut_z Jul 7 Fich1
```

Le but de l'exercice est de déterminer les commandes permettant à l'utilisateur `toto` de modifier les droits d'accès sur le fichier `Fich1` pour avoir :

```
-rwxr-xr-x 1 toto iut_z Jul 7 Fich1
```

Q 1. Donnez une solution numérique.

Q 2. Donnez une solution avec les actions sous forme symbolique.

Q 3. La réponse à la question précédente est-elle unique ?

Q 4. Peut-on y répondre avec une seule commande en utilisant le mode symbolique ?

Exercice 6 : Validation des droits

Pour cet exercice on fait les considérations suivantes :
– les utilisateurs `darwin` et `lamarque` appartiennent au groupe `serieux` ;
– l'utilisateur `dieu` appartient au groupe `blagues` ;
– on se trouve dans le répertoire `singes` de la hiérarchie de la figure 1 ;
– on considère que le fichier `hommes` est un répertoire.

Q 1. Pour chacune des commandes suivantes quelles sont les droits qu'un utilisateur quelconque doit posséder pour pouvoir les exécuter ?
– `cp /mammiferes/canides/loup hommes`
– `cp /mammiferes/singes/gorille chimpanze`
– `rm gorille`
– `ln /mammiferes/canides/loup hommes`
– `chmod 666 /mammiferes/singes/hommes`

Q 2. Pour chacun des utilisateurs `darwin`, `lamarque` et `dieu` spécifiez s'ils ont le droit d'exécuter chacune des commandes précédentes.

```
Hiérarchie               Inode    Type   Propriétaire   Groupe     Mode d'accès
----------               -----    ----   ------------   ------     ------------

/                            2               darwin  serieux        r-xr-xr-x
|-- mammiferes               3               darwin  serieux        rwxr-x---
|   |-- canides              4               darwin  serieux        rwx-wx---
|   |   |-- chacal           5
|   |   |-- chien            6
|   |   |-- coyote           7
|   |   `-- loup             8               darwin  serieux        r-x-w-rw-
|   |-- cetaces              9
|   |   |-- baleine         10
|   |   |-- dauphin         11
|   |   `-- orque           12
|   `-- singes              13               darwin  serieux        r-x--x-wx
|       |-- chimpanze       14               darwin  serieux        rwxrw-r-x
|       |-- gorille         15                 dieu  blagues        -wx---rwx
|       |-- homme           16               darwin  blagues        rwxr-x-w-
|       `-- orang-outan     17             lamarque  serieux        ---rwx---
|-- oiseaux                 18
|   |-- autruche            19
|   |-- colibri             20
|   |-- moineau             21
|   `-- pingouin            22
`-- reptiles                23
    |-- avec_pattes         24
    |   |-- caiman          25
    |   |-- crocodile       26
    |   `-- tortue          27
    `-- sans_pattes         28
        |-- boa             29
        `-- cobra           30
```

Figure 1 – **Hiérarchie animale.** Des numéros d'inodes fictifs ont été placés entre parenthèses sur la représentation pour simplifier les exercices. Le contenu de chacun des fichiers **réguliers** correspond simplement au nom du fichier en caractères minuscules.