TP n°2

Dans ce TP, il est demandé de créer un Tiddler intitulée TP2 dans votre Wiki et de rédiger rapidement certaines réponses, en mettant un paragraphe qui décrit le problème : « le but est de faire ceci : ... » et rajoutez ensuite une copie écran de vos commandes {{{ la copie }}}, relire le TD1 à ce sujet.

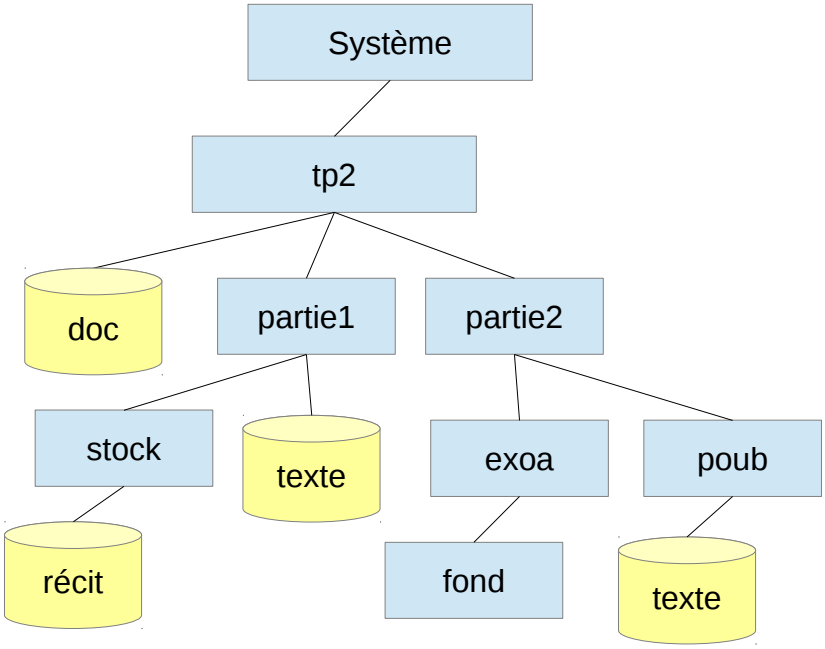
1) Fichiers et répertoires

a) Préparation et premières commandes sur les dossiers

- Créez deux dossiers dans votre compte Unix, l'un appelé `algo`, l'autre `systeme` (ou sys), en faisant `mkdir algo` puis `mkdir systeme`. Affichez la liste des fichiers (et des dossiers) par la commande `ls`. Faites un dessin des fichiers et des dossiers de votre compte. Vérifiez à l'aide de l'interface graphique que ces dossiers sont bien créés.
- Dans le terminal, descendez dans le répertoire système en faisant `cd systeme`. Constatez que la commande `ls` n'affiche rien. C'est normal il n'y a aucun fichier à cet endroit. Utilisez l'éditeur pour créer un fichier appelé `essai` dans ce répertoire. Vérifiez avec `ls` l'existence et le contenu de ce fichier et mettez à jour le dessin de votre arbre de fichiers.
- Remontez dans le répertoire de votre compte (*home directory*) en faisant `cd ..` Vérifiez que `ls` n'affiche rien du répertoire `systeme` mais les fichiers de votre compte. Faites `ls systeme` pour voir le répertoire en question.
- Redescendez dedans puis faites `ls ..` pour voir le répertoire du dessus. Remontez à nouveau dans votre compte et faites par curiosité `ls ..` pour voir ce qui se trouve au dessus : l'ensemble des comptes des étudiants de première année.
- Prendre note des observations : `ls` sans paramètre affiche le répertoire courant, `ls` *`rép`* affiche le contenu du répertoire rep (qui doit se trouver dans le répertoire courant) et `ls ..` affiche le contenu du répertoire englobant le répertoire courant. Faire un petit schéma sur papier avec 3 répertoires imbriqués contenant chacun quelques fichiers, se placer dans le répertoire central et écrire ce qu'affichent `ls`, `ls ..` et `ls` *`bas`*.
- Dans le répertoire `systeme`, créez un sous répertoire appelé `tp1`. Il faudra prendre l'habitude de bien ranger les fichiers dans les bons répertoires, afin de rapidement retrouver le travail en cours et de faire place nette pour chaque activité (algo, sdd, bdd, ppp…). Quand vous commencez un TP, créez systématiquement un répertoire pour lui et faites-en votre répertoire de travail, tout ce que vous ferez sera mis à cet emplacement.
- On peut déplacer un fichier d'un endroit à l'autre : il suffit de faire `mv fichier endroit`. Déplacez les fichiers créés aujourd'hui et lors du TP1 vers ce répertoire `tp1` situé dans `systeme`, éventuellement en plusieurs étapes.

b) Arborescence, commandes

- On repart du dossier appelé `systeme` (ou Système ou sys) créé précédemment. À l'aide des commandes `mkdir`, `cd` et un éditeur de texte, créer l'arbre des fichiers dessiné ci-dessous. Les ovales représentent des fichiers, les rectangles sont les répertoires. Dans chacun des 4 fichiers, vous placerez une ligne indiquant le nom du fichier et son emplacement, p. ex. : « voici doc du répertoire tp2 », c'est parce qu'il faudra afficher le contenu et être sûr que c'est bien le bon fichier :



Vous vérifierez que l'arbre est correctement créé en utilisant la commande `tree` ou `ls -R` à partir du dossier `tp2`. Ces deux commandes pourraient être inscrites dans la liste des commandes à connaître dans votre Wiki.

- On trouve deux fichiers appelés `texte` dans cet arbre, pourquoi est-ce possible ? Tentez de créer un autre fichier appelé `doc` dans le répertoire `tp2`, tentez de créer un autre sous-répertoire appelé `partie2` dans `tp2` et enfin tentez de créer un sous-répertoire appelé `doc` dans `tp2`.
- Vérifiez que la commande `rmdir` refuse de supprimer le répertoire `partie1`. Quelle est la signification du message d'erreur et que faudrait-il faire pour parvenir à supprimer le répertoire `partie1` (mais ne pas le faire maintenant) ? Écrire la liste des commandes que cela implique si on ne veut pas utiliser l'option `-r` de `rm`
- Exercice : quel est le plus petit jeu de commandes à taper pour créer tout cet arbre, sachant qu'une commande comme `mkdir` peut recevoir plusieurs paramètres et qu'on peut faire : `mkdir titi titi/toto titi/toto/tutu`. Saurez-vous recréer cet arbre en 5 commandes en tout ? Ceci est à rédiger dans le wiki : ces commandes et la sortie de `tree` qui montre le résultat obtenu.

c) *Chemins, noms complets*

- Placez-vous dans le répertoire `partie1` et n'en bougez pas. Que devrait afficher la commande `ls` ? Vérifier ce qu'elle affiche effectivement en lançant cette commande. Vous pouvez comparer en ouvrant une fenêtre sur les fichiers et en naviguant vers le bon dossier.
- Quelles sont les commandes qui permettent d'aller du répertoire `partie1` au répertoire `partie2` ? Vous pouvez faire `ls` pour vérifier que vous y êtes arrivé(e). Quelles sont ensuite les commandes pour aller de `partie2` à `fond` ? Quelles sont les commandes pour aller de `fond` à `tp2` ? Pour finir, revenez dans `partie1`.
- Quel est le nom relatif du fichier récit par rapport à partie1 ? Pour vérifier, il vous suffit de taper la commande `more` adéquate
- Quels sont les noms complets relatifs à `partie1` de chacun des deux fichiers appelés `texte`. Vérifier que ces noms sont les bons en tapant les commandes `more` adéquates sans changer de répertoire (ex : `more bidule/truc/texte`).
- Les dossiers aussi peuvent être désignés. Alors quel est le nom relatif du répertoire `tp2` vu de `partie1` ? Quel est le nom relatif de `exoa` vu de `tp2`? Idem avec `fond` vu de exoa. Idem avec `partie2` vu de `fond`. Enfin quel est le nom complet relatif de `partie1` vu de `partie2` ? Cette manipulation est un peu compliquée si vous débutez, alors ne la faites que si ça vous tente.
- En vous plaçant dans `tp2`, faites une copie du fichier doc dans le répertoire `partie1`. Copiez le fichier `recit` dans `fond`. A chaque fois, vérifiez les opérations avec la commande `ls -R ~/sys/tp2`. Placez-vous dans `partie2` et copiez le fichier `doc` dans `poub`, copiez le fichier `texte` de `poub` dans `fond`. Copiez le fichier `recit` dans `exoa`. Copiez le fichier `texte` de `partie1` dans `partie2`.

- Un fichier appelé `bravo` a été placé dans le répertoire `tp2` situé dans `1A` lui-même dans `SYS` du répertoire `anonymous` qu'on trouve dans `local` sous `usr` de la racine. Quel est son nom complet absolu, quel est son nom complet relatif vu de votre compte ? Copiez ce fichier chez vous comme vous voulez et surtout notez cette commande dans le wiki, car elle est assez complexe. D'autre part, vous pouvez noter dans le wiki toutes les erreurs que vous avez eu : fichier non trouvé, chemin inexistant et commande cp incorrecte. Toutes ces erreurs constituent votre expérience professionnelle et ne devraient plus être refaites à l'avenir.

2) Exercices sur les jokers

Créez maintenant un nouveau dossier ~/système/tp2/jokers (à vous d'adapter), placez-vous dedans. Continuez en créant les fichiers suivants à l'aide de la commande suivante :

```
touch prog.c prog.o projet.c projet.o projet.out presentation scene
```

Donner les commandes les plus courtes possibles pour effectuer les opérations demandées ci-dessous. « Montrer » signifie qu'on utilise la commande ls et qu'il faut lui donner un paramètre qui ne lui fait afficher que les fichiers concernés. Par exemple, ls tot*o. Notez les questions, les réponses et les résultats dans le Wiki.

1. montrer prog.c et prog.o
2. montrer prog.c et projet.c
3. montrer projet.o et projet.out, (mais pas projet.c ni les autres)
4. montrer projet.c, projet.o, projet.out et presentation
5. montrer tous les fichiers sauf presentation
6. montrer projet.c et projet.o mais pas projet.out
7. montrer presentation et scene mais aucun autre.
8. montrer scene seulement
9. montrer projet.out et presentation

3) Lecture de la documentation

Le but de cette partie est d'apprendre à se servir de la documentation en ligne, c'est à dire disponible sur l'ordinateur. Presque toutes les commandes Unix sont bien expliquées en ligne, il manque seulement des exemples d'utilisation plus détaillés, mais on les trouve sur les forums internet. On y accède par la commande man, p. ex. :

✍ `man who`

Cette documentation est en plusieurs parties : le résumé, la liste détaillée des options, des exemples, les cas d'erreurs et la liste des commandes similaires.

Il faut savoir également que la documentation est en plusieurs volumes. Le volume 1 concerne les commandes Unix (ex : ls), le volume 3 concerne les fonctions standard du langage C (ex : printf), le volume 2 regroupe les fonctions du système d'exploitation accessibles en C (ex : open). Les autres volumes concernent l'administration du système. Quand on trouve une référence à une commande, il faut vérifier qu'elle est bien dans le volume 1. Par exemple, `man 1 mkdir` donne la doc de la commande shell mkdir tandis que `man 2 mkdir` donne la doc de la fonction mkdir du langage C (on peut la placer dans un programme et elle crée un dossier).

- Afficher la documentation de la commande ls et chercher l'option qui permet d'afficher *récursivement* tous les sous-répertoires – testez la, que fait-elle ? La doc est tellement longue qu'il faut utiliser la recherche de mot-clé.
- Quelles options de ls faut-il pour afficher les fichiers triés selon les tailles, avec les plus petits en premier ?
- Afficher la documentation de la commande who pour savoir comment afficher seulement deux lignes : le nombre d'utilisateurs connectés et la liste des noms.
- Afficher la documentation de la commande rm et chercher les options qui permettent de supprimer récursivement tout un répertoire et son contenu, sans jamais demander de confirmation. Cette option de commande serait à noter dans la liste des commandes à connaître.
- Afficher la doc de la commande date et cherchez comment faire pour afficher au format nom du jour, numéro du jour et mois dans cet ordre, comme par exemple « dimanche 14 juillet ». Vous pourriez noter cette commande et son résultat dans votre wiki (page du TP2).

4) Étude de quelques commandes utiles

Cette partie propose de découvrir diverses commandes. Il est normal d'avoir un peu de mal à les faire fonctionner correctement au début. Notez bien les essais que vous faites. Lisez la doc en ligne systématiquement. Rédigez une phrase dans votre wiki avec la copie d'écran pour vous souvenir de vos manipulations.

a) find (très utile)

Cette commande permet de trouver des fichiers dans une arborescence (recherche *récursive*). On donne des critères de recherche, p. ex. tel nom, telle date, telle taille… et cette commande affiche les noms complets des fichiers qui correspondent. Elle est similaire à l'assistant recherche de Windows (le chien qui frétille).

Cette commande possède de nombreuses options qui sont décrites dans la documentation. Le principe de base est de donner d'abord le chemin de l'arbre à parcourir, ex `.` pour le répertoire courant, `~` pour le compte, ~/algo pour le répertoire algo de votre compte, etc. les fichiers seront cherchés dans ce répertoire et ses sous-répertoires. Ensuite, on donne les critères de recherche, par exemple `find ~ –name 'toto'` recherchera tous les fichiers appelés toto dans votre compte. -name '*nom*' est l'un des critères de recherche, il y en a beaucoup d'autres : sur la date de création, de modification, sur la taille, etc. Consulter la documentation en ligne.

Par exemple, `find ~ –name '*~'` cherche tous les fichiers temporaires créés par les éditeurs de texte : dans ~ (le compte), on recherche ce qui porte un ~ à la fin du nom. Essayez de chercher ce qui porte un # au début.

Vous remarquerez qu'elle farfouille dans tous les répertoires cachés de votre compte (faites ls –a pour les voir). Ce sont des répertoires de configuration de certains logiciels.

Pour éviter ça : `find ~ –name '\.*' -prune -o ...critères... -print`

Le mot anglais *prune* signifie élaguer : on ne va pas dans les répertoires dont le nom est `.*` (joker).

Employez cette commande pour rechercher l'emplacement de votre fichier soleil (TP n°1). Affichez les noms de tous vos fichiers source C (*.c).

Dans le même style, il y a la commande `locate`. À vous de voir si elle est installée et ce qu'elle permet de faire...

b) encodage base64 (peu utile en pratique)

Cette technique permet d'envoyer un fichier quelconque (image, musique...) par e-mail. En interne, le courrier électronique ne tolère que des fichiers texte ; on ne peut pas envoyer un fichier binaire, c'est à dire contenant autre chose que des lettres, des chiffres et des signes de ponctuation. Le système de codage base64 consiste à transformer les octets d'un fichier binaire en séquences de symboles choisis parmi les lettres, chiffres, +, / et =. Le fichier obtenu est 33% plus gros mais il peut être envoyé par e-mail sur internet car c'est une sorte de texte.

Voici le mode d'emploi :

✍ `openssl enc –base64 –in` *`monfichier`* `–out` *`monfichier.b64`*

transforme le fichier indiqué dans un sens ou dans l'autre :

✍ `openssl enc –d –base64 –in` *`monfichier.b64`* `–out` *`monfichier`*

extrait les données du fichier encodé.

Tester ce procédé sur un de vos programmes exécutable C : faites-en une copie dans le répertoire courant, encodez-le, regardez ce qui se trouve dans le fichier encodé. Effacez le fichier initial (la copie du programme). Décodez le fichier encodé et vérifier que le programme fonctionne encore.

NB: le droit d'exécution doit être remis en place, faites : `chmod u+x ceprogramme`

c) gzip et gunzip, bzip2 et bunzip2 (assez utiles)

Ces commandes permettent de compresser des fichiers : après opération, ils prennent moins de place mais ils ne sont plus utilisables, il faut les décompresser. gzip et bzip2 sont deux compresseurs, gunzip et bunzip2 sont les décompresseurs correspondant. Ils sont très simples à utiliser :

✍ `gzip` *`fichier`*

transforme le fichier indiqué de manière à ce qu'il occupe moins de place. Son nom est changé en nom.gz.

✍ `gunzip` *`fichier`*

décompresse le fichier compressé. Il n'est pas nécessaire de fournir l'extension .gz.

Les commandes bzip2 et bunzip2 fonctionnent exactement pareil.

- Faire l'essai de compresser un source C (copié) puis de le décompresser.
- La commande `ls –l` affiche la taille des fichiers, c'est le nombre qui se trouve juste avant la date de création. La commande `wc -c *` affiche également la taille des fichiers. Comparer les tailles des fichiers avant et après compression, et déterminez le taux de compression obtenu. Faire la même comparaison avec un programme exécutable, en déduire que le compresseur n'est vraiment efficace qu'avec les textes.

d) tar (assez utile)

Cette commande permet de transformer tout le contenu d'un répertoire en un seul fichier qu'on appelle une *archive*. L'avantage est de pouvoir garder ensemble tout un lot de fichiers, par exemple un projet et on peut le copier sur une clé USB sans risquer d'en perdre un. Les installateurs de logiciels windows (.msi), linux (.deb) ou android (.apk) sont des sortes d'archives autoextractibles. Si on veut, ce fichier d'archive peut être à son tour compressé ou encodé en base64.

La manipulation d'une archive se fait par la commande tar (*tape archive*). Cette commande prend au moins deux paramètres : une directive et le nom du fichier d'archive. Les directives sont les triplets de lettres `cfv`, `tfv` ou `xfv` :

✍ `tar cfv fichier.tar répertoire`

crée le fichier.tar avec ce qui se trouve dans le répertoire. Il est prudent de supprimer le fichier.tar s'il existe déjà afin qu'il ne rentre pas en conflit avec ce qu'on veut y mettre. Il est **très important** de donner un nom relatif au répertoire et non pas son nom absolu afin de pouvoir désarchiver ailleurs.

✍ `tar tfv fichier.tar`

affiche ce qui se trouve dans le fichier d'archive. Le f indique qu'on fournit un fichier, le v signifie *verbose*, bavard.

✍ `tar xfv fichier.tar`

extrait tous les fichiers de l'archive, recrée le répertoire qu'elle contient. Si le répertoire avait été spécifié par un chemin absolu, il est remis à son exacte ancienne place !

- Pour faire des manipulations, repartez de l'arbre des fichiers du début de ce tp et faites-en une archive. Vérifiez que l'archive est correcte (tfv). Déplacez l'archive au niveau du répertoire stock et extrayez les fichiers (ça va recréer toute une arborescence à ce niveau). Au pire, si vous ratez, vous devrez reconstruire l'arbre.

En général, on comprime l'archive. Il y a deux moyens, soit on comprime l'archive en utilisant gzip, soit on rajoute la lettre z à toutes les directives tar : tar cfvz…, tar tfvz…, tar xfvz… et on nomme l'archive archive.tgz au lieu de archive.tar. Faites quelques essais.

- Manip de synthèse : choisissez un répertoire (ex : algo) faites-en une archive compressée, encodez-la en base64 et compressez le fichier encodé. Ensuite, créez un répertoire ~/tmp dans lequel vous mettrez le fichier encodé et compressé. Décodez-le et extrayez l'archive à cet endroit. Vérifiez que vous avez tous vos fichiers. Pour finir, effacez ~/tmp.

La commande tar est gérée par l'interface Unix, un outil qui s'appelle gestionnaire d'archive.

e) split et cat (peu utiles)

Quand on dispose d'un fichier trop gros pour tenir sur une clé USB, ou pour être envoyé par mail, il est nécessaire de le couper en morceaux. La commande split fait cela :

✍ `split –l nb fichier préfixe`

Sépare en plusieurs fichiers de nb lignes chacun le fichier indiqué. Les nouveaux fichiers s'appelleront prefixeaa, préfixeab, préfixeac…préfixeba, préfixebb… et ainsi de suite. Comme préfixe, on peut choisir le nom du fichier suivi d'un _. Vérifiez ces informations dans la doc en ligne.

- Faire l'essai avec un fichier texte assez long, faites-en des morceaux de 5 lignes.
- Combinez les joies de tar pour faire une archive, bzip pour la compresser, base64 pour l'encoder et split pour en faire des tranches de saucisson.

Réciproquement, pour réunir les morceaux :

✍ `cat préfixe* > fichier`

Cette commande rassemble tous les *préfixe*… en un seul morceau appelé *fichier*. L'essayer sur les fragments que vous avez obtenus avec split. Alors, il marche encore ?