Module ITC313 - Informatique

TPs UNIX / Shell

Benoît Darties - benoit.darties@u-bourgogne.fr
Univ. Bourgogne Franche-Comté

Année universitaire 2015-2016

Avant-propos : Support de Travaux Pratiques (TP) dispensés à l'Université Bourgogne Franche-Comté, enseignements en Informatique partie UNIX. La totalité de ce document a été rédigée uniquement à partir des connaissances de son auteur, et en utilisant un matériel personnel. L'utilisation / réutilisation partielle ou complète d'éléments de ce document est soumise à l'approbation de son auteur. Les corrections détaillées de chaque exercice sont réservées au personnel académique et disponibles par mail sur simple demande.

Consignes :

Les exercices sont à réaliser en monôme exclusivement. Le travail à réaliser se doit d'être votre propriété exclusivement. Les enseignants de TP sont là pour vous aiguiller durant les séances prévues à cet effet. Les exercices pourront être complétés en dehors de ces séances en cas de manque de temps.

Au terme des séances de TP du module Informatique, il vous sera demandé de remettre un compte rendu reprenant les réponses aux différents exercices proposés et dans lequel vous détaillez la démarche que vous avez adoptée. Ce compte rendu se présente plus sur la forme d'un rapport permettant de suivre le cheminement de votre démarche en la repositionnant dans le contexte de l'exercice, plutôt qu'une simple liste de réponses aux exercices. Il devra contenir une introduction, une table des matières, une liste des figures, une conclusion, et le cas échéant une bibliographie. Chaque figure doit être numérotée et référencée dans le texte à l'endroit adéquat. Le document pourra être rédigé en Word, LaTeX, ou tout autre éditeur de votre choix. Mais il devra être soumis en un seul fichier `pdf` dont la taille, images comprises, ne devra pas excéder *30 Mo*

La date limite de remise des comptes-rendus vous sera communiquée par la suite. Le compte-rendu global sera à déposer sur le serveur de dépôt à l'adresse suivante (qui sera opérationnelle une semaine avant la date limite de remise du rapport) :

`http://depot.infotro.fr/ITC313/`

Des éléments complémentaires pourront, le cas échéant, vous être communiqués par mail.

En cas de fortes similitudes avec un travail réalisé par un autre étudiant de votre promotion ou d'une promotion précédente, une note de 0 non négociable sanctionnera votre travail, ainsi que celui de la personne ayant communiqué son travail à un tiers si ce dernier fait partie de la même promotion. Il vous appartient donc de protéger l'intégrité et la confidentialité de votre travail.

1 Manipulations de l'environnement et des fichiers sous UNIX

Les exercices de cette section évaluent vos capacités à manier quelques commandes classiques de l'environnement UNIX, et à comprendre les mécanismes de base des systèmes de fichiers.

Exercice 1 : *Decouverte de quelques commandes d'archivage*

L'objectif de cet exercice est de vous faire découvrir et manipuler quelques commandes UNIX, et développer vos capacités à vous approprier des commandes qui pourraient vous être inconnues en lisant la documentation associée. Entre autres, vous allez manipuler les commandes suivantes :

- La commande `wget` : commande puissante qui permet de télécharger des fichiers d'internet en ligne de commande, d'aspirer le contenu d'un site web, et bien d'autres
- La commande `tar` : permet de rassembler des fichiers sous forme d'archive, et de décompresser une archive pour en récupérer les fichiers qu'elle contient.
- Les commande `gzip` et `gunzip` : permet de compresser (`gzip`) un fichier en un fichier zippé au format `.gz`, ou de décompresser un fichier au format `.gz` en un fichier non compressé.

1. Récupération et décompression d'une archive
   (a) Positionnez vous dans un répertoire vide avec accès en écriture, et exécutez la commande `wget https://cloud.infotro.fr/ITC313/archive.tar` Quel est le résultat de cette commande ?
   (b) En vous aidant du manuel, à quoi servent respectivement les options `-x`, `-c` et `-f` de la commande `tar` ?
   (c) Dans le même répertoire, utilisez la commande `tar -x -f archive.tar` pour décompresser le fichier archive que vous venez de télécharger. Combien de fichiers étaient présents dans cette archive ?
2. Manipulation de fichiers
   (a) L'un des fichiers avec l'extension `jpg` n'est pas une image au format jpeg, mais au format jpg2000. En utilisant la commande `file`, essayez de retrouver lequel, puis recherchez l'extension exacte de ce format de fichier sur internet. Quelle commande exécutez-vous pour renommer l'image avec le format correct ?
   (b) En utilisant la commande `ls` avec les options adéquates, quelle est la taille du fichier `script.txt` ?
   (c) Utilisez la commande `gzip` sur le fichier `script.txt`. Que s'est-il passé ? Quelle est la taille du nouveau fichier créé ? Exprimez-le pour pourcentage de compression par rapport au fichier original. Utilisez ensuite la commande `gunzip` sur ce nouveau fichier. Que s'est-il passé ?
3. Création d'une nouvelle archive :
   (a) En utilisant la commande `tar -c -f nouvelleArchive.tar *.jpg *.txt *.jp2`, créez une nouvelle archive au format `tar` contenant l'ensemble des fichiers image et texte de l'archive originale. Quelle est la taille de cette archive ? Quelle est la différence de taille par rapport à la somme des tailles des fichiers qu'elle contient ?
   (b) Utilisez la commande `gzip` sur le fichier `nouvelleArchive.tar` et observez le résultat. Quelle est la différence de taille par rapport à la somme des tailles des fichiers qu'elle contient ?
   (c) Utilisez la commande `tar -c -z -f nouvelleArchive2.tar.gz *.jpg *.txt *.jp2` pour créer une nouvelle archive au format `tar.gz` contenant l'ensemble des fichiers image et texte de l'archive originale. Quelle est la taille de cette archive ? Quelle est la différence de taille par rapport à la somme des tailles des fichiers qu'elle contient ? Quelle est la différence de taille par rapport au fichier `nouvelleArchive2.tar.gz` résultant de la commande précédente ? Finalement, que fait l'option `-z` ?
   (d) Utilisez la commande `tar -c -z *.jpg *.txt *.jp2` sans préciser de nom d'archive à créer avec l'option `-f`. Que remarquez-vous ? Utilisez ensuite la commande `tar -c -z *.jpg *.txt *.jp2 > nouvelleArchive3.tar.gz`, et comparez sa taille avec les fichiers `nouvelleArchive2.tar.gz` et `nouvelleArchive.tar.gz`. Concluez.

Exercice 2 : *Utilisation des masques de creation de fichiers*

Cet exercice a pour objectif de vous familiariser avec les droits initiaux avec lesquels sont créés les nouveaux fichiers sur un système UNIX

1. Créez quatre fichiers réguliers vides nommés `Raphael.txt`, `Donatello.txt`, `Michelangelo.txt` et `Leonardo.txt` avec la commande `touch`, ayant respectivement les droits `rw-rw-rw-`, `---------`, `r--r--rw-`, et `------rw-`, mais sans utiliser la commande `chmod` : simplement en positionnant le masque de droit avant création des fichiers au moyen de la commande `umask` vue en cours avant de créer le fichier.
2. Est-il possible de créer un cinquième fichier régulier nommé `TortueGeniale.txt` avec les droits `rwx-wx-w-` sur le même principe (sans utiliser `chmod`, mais avec `umask`) ? Si oui, donnez la suite de commande permettant de créer ce fichier, sinon expliquez pourquoi.
3. Même question avec un fichier de type répertoire nommé `Franklin` avec les droits `rwx-wx-w-`.

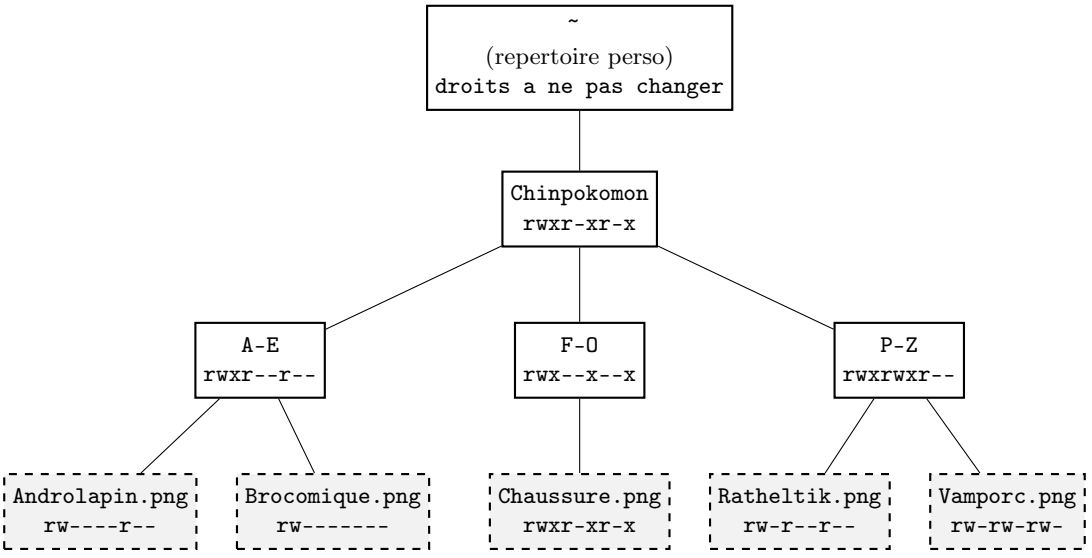
Exercice 3 : *Manipulation du systeme de fichier et des droits et navigation*

Dans cet exercice, nous manipulerons quelques commandes de navigation dans l'arborescence et de création de fichiers. Cet exercice requiert les bases qui auront été acquises dans l'exercice *Découverte de quelques commandes d'archivage*.

1. En utilisant les compétences désormais acquises dans la récupération et la gestion des archives, récupérez, décompressez et extrayez l'archive présente à l'adresse

   `https://cloud.infotro.fr/ITC313/chinpokomon.tar.gz`

   en utilisant uniquement des instructions de la ligne de commande. Que contient cette archive ?
2. En vous appuyant sur les fichiers récupérés dans l'archive précédente, reproduisez l'arborescence suivante dans votre répertoire personnel en utilisant *uniquement* les commandes `mkdir`, `mv` et `chmod`, et en appliquant les droits mentionnés pour chaque répertoire.



3. Quel est le nom absolu du fichier `Vamporc.png` ?
4. Quel est le nom relatif du fichier `Vamporc.png` depuis le répertoire `A-E` ?
5. Chaque chinpokomon possède également un nom en anglais, qui est mentionné dans chaque image. Sans modifier l'arborescence précédemment créée, on vous demande de reproduire une nouvelle arborescence similaire, avec les contraintes suivantes :

- le nom `Chinpokomon` est remplacé par `Chinpokomon (us)`,
- la commande `chmod` ne doit pas être utilisée. Aussi, on utilisera la commande de masques `umask` permettant de déterminer les droits d'un fichier régulier / répertoire *avant* création du fichier.
- Chaque fichier image doit porter le nom du chinpokomon en anglais présent sur l'image (par exemple : `Vamporc.png` devra s'appeler `Vamporko.png`). On ne souhaite pas se retrouver avec une copie de fichier. Aussi on utilisera uniquement des liens phyisiques (commande `ln`) sur les fichiers images.

6. Créez une archive au format `.tar.gz` contenant les deux arborescences créées, nommée `ITC313_TP_Shell_`*`nom.prenom`*`.tar.gz` . A quoi sert l'option -z ? Quelle commande permettra de décompresser cette archive ?
7. Copiez cette archive sur un support ou utilisez le mail pour la transférer sur un autre ordinateur, puis décompressez cette archive. Les permissions des fichiers ont-elles été conservées ? existe-t'il une option dans `tar` permettant de conserver les droits des fichiers ?

Exercice 4 : *Manipulation d'expressions regulieres*

Un très bref aperçu des expressions régulières est présenté dans cet exercice. Les expressions régulières sont un outil extrêmement puissant mais qui nécessite une certaine pratique. Cette pratique est très brièvement initiée au sein de cet exercice via différentes illustrations d'expressions régulières, afin que les étudiants puissent posséder les bases leur permettant de se former par la suite sur la confection de modèles pour expressions régulières. La syntaxe que nous utiliserons sera toujours :

- *`grep -E "expressionReguliere"` pour sélectionner une ligne contenant un motif,*
- *ou `grep -o -E "expressionReguliere"` pour sélectionner seulement le motif*

1. Récupérez le fichier `https://cloud.infotro.fr/ITC313/unGrosBordel.txt` et affichez son contenu avec la commande `cat`. Remarquez les différents éléments, tels que des mots, des adresses mails, des lignes avec un ou plusieurs mots, des majuscules et minuscules, etc. Regardez entre autres les mots au début ou en fin de phrase.
2. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "ette"`
3. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "T"`
4. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "^T"`
5. Comparez les résultats des questions 3 et 4. Que représente le caractère spécial `"^"`
6. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "te$"` .
7. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "c.r"`
8. Qu'ont en commun les lignes affichées par `cat unGrosBordel.txt |grep -E "(oui|non)"`
9. Déduisez-en ce que représentent les caractères spéciaux `"$"`, `"|"`, et `"."`
10. que fait l'option `-o` lorsqu'on l'ajoute à la question 7, c'est à dire que l'on exécute la commande `cat unGrosBordel.txt |grep -o -E "c.r"`

Dans la suite de l'exercice, on utilise l'option `-o` pour se limiter au motif et non à la ligne.

11. Qu'ont en commun les motifs affichées par `cat unGrosBordel.txt |grep -o -E "[A-Z]{4}"`
12. Qu'ont en commun les motifs affichés par `cat unGrosBordel.txt |grep -o -E "[A-Z][a-z]+"`
13. Qu'ont en commun les motifs affichés par `cat unGrosBordel.txt |grep -o -E "[A-Z][a-z]*"`
14. Comparez les résultats des questions 12 et 13 , et déduisez-en ce que représentent les caractères spéciaux `"+"` et `"*"`

Commentez enfin le motif suivant et essayez d'en comprendre la signification ;

15. `cat unGrosBordel.txt |grep -o -E "[A-Za-z0-9\.\_]+@([A-Za-z0-9\-]+\.)*[a-zA-Z]{2,4}"`
16. `cat unGrosBordel.txt |grep -o -E "(\+33|0)(\.| )?[0-9]((\.| )?[0-9]{2}){4}"`
17. Trouvez l'expression qui sélectionne uniquement les mots en minuscule, sans accent ni espace et encadrés par une double parenthèse de part et d'autre, et affichez la phrase secrète. Les caractères spéciaux comme (doivent être précédé de \ pour être considérés comme caractères.

2 Édition de scripts

Les exercices de cette section évaluent vos capacités à définir des scripts UNIX, et exécuter des successions de commandes en utilisant les outils orientés programmation du shell. Vous serez emmenés à rédiger des fichiers scripts dans ces exercices. Vous pouvez utiliser les éditeurs de texte que vous souhaitez, dans la mesure ou ces derniers peuvent produire des fichiers au format texte simple (pas de texte enrichi. Ex : word est à bannir), qu'ils soient en interface graphique (par exemple `gedit`, `notepad++`, `codeblocs`, *...) ou en ligne de commande (*`nano` *conseillé pour les débutants,* `emacs` *pour les avancés,* `vi` *pour les désespérés,...).*

Exercice 5 : *Un premier script*

L'objectif de cet exercice est de vérifier vos bases en matière de création de script : instructions élémentaire et exécution.

1. Créez un fichier script *hello.sh* affichera le texte "~~hello world~~" "hello Kitty"
2. Ajoutez un commentaire dans ce fichier qui indique le nom du ou des auteur(s) (vous) et la date de création du script. Vous pouvez garder cette habitude pour tous les scripts que vous créez ou modifiez.
3. La commande `whoami` permet de connaître le nom de connexion de l'utilisateur qui la lance. Rajoutez une ligne pour que le message "Hello *nom de login*" s'affiche.
4. Modifiez votre script pour qu'il affiche comme message à l'utilisateur "Do you like fishsticks(y/n)?", récupérez la réponse dans une variable nommée `reponse`, puis affichez la réponse "then you're a gayfish!" si la réponse saisie est `y`.

Exercice 6 : *Comptage des parametres*

Cet exercice vous permet de vous familiariser avec le passage de paramètres au sein d'un script, leur comptage, et la récupération de leur valeur sans utiliser les variables `$1, $2, ....`

Ecrivez deux scripts `parametres.sh` et `parametres2.sh` qui comptent le nombre de leurs paramètres lorsqu'ils sont exécutés, et affiche leur valeur ainsi que leur nombre.

1. le script `parametres.sh`, affiche les paramètres à l'aide d'une boucle `for` et de la variable $*.
2. le script `parametres2.sh` affiche les paramètres à l'aide de la commande `shift`.

Quelques exemples d'exécution et de résultat attendu sont présentés ci-après :

```
$ ./parametres.sh

- liste des parametres entres :
- nombre de paramètres : 0

$ ./parametres.sh  Ginyu Reacum Butta Jeese Ghourd
- liste des parametres entres :  . Ginyu Reacum Butta Jeese Ghourd
- nombre de paramètres : 5

$ ./parametres2.sh  mu aldebaran saga-kanon  MasqueDeMort Aiolia Shaka Dohko Milo
    Aiolos Shura Camus Aphrodite
- liste des parametres entres :   mu aldebaran saga-kanon MasqueDeMort Aiolia Shaka
   Dohko Milo Aiolos Shura Camus Aphrodite
- nombre de paramètres : 12
```

Exercice 7 : *Portee des variables*

Cet exercice vous permet de comprendre la notion de portée de variable, de variable locale et héritée, ainsi que les mécanismes de transfert de valeurs entre shell. Nous parlerons de terminal, mais ce dernier fait également référence à un shell.

1. Portée des variables locales créées :
   (a) Dans un terminal, créez une variable `midichloriens` et initialisez sa valeur à `50000`.
   (b) Affichez cette variable à l'écran avec l'instruction `echo "la valeur de midichloriens est $midichloriens "`
   (c) Créez un script nommé `yoda.sh`, recopiez-y l'instruction de la question précédente, et exécutez ce script. Que se passe-t-il ? Qu'en concluez-vous sur la portée de la variable `midichloriens`, c'est à dire la "zone" depuis laquelle la valeur de cette variable est accessible ?
   (d) Copiez le script `yoda.sh` dans un nouveau fichier `vader.sh`. Ajoutez au début du fichier `vader.sh` une instruction qui initialise la variable `midichloriens` avec une valeur de `20000` et exécutez le script `vader.sh`, puis à nouveau le script `yoda.sh`. Exécutez enfin l'instruction `echo "la valeur de midichloriens est $midichloriens "` dans le shell. Que concluez vous sur les variables créées et leur portée ?
2. Portée limitée au shell :
   (a) Lancez un second terminal, et affichez directement la valeur de la variable `midichloriens` en utilisant la même instruction que précédemment. Que se passe-t-il ? Fermez ce second terminal et revenez dans le terminal de départ.
   (b) Lancez un sous-shell dans le teminal actuel en appelant la commande `bash`, et affichez à nouveau la variable `midichloriens`. Que se passe-t'il ?
   (c) Executez une fois seulement la commande `exit` pour fermer le sous-shell et revenir dans le terminal de départ. Affichez à nouveau la variable `midichloriens` et vérifiez que cette dernière contient bien la valeur `50000`. Que concluez-vous ?
3. Etendre la portée de la valeur d'une variable locale :
   (a) Exportez la variable `midichloriens` en tapant la commande `export midichloriens`. Refaites alors les manipulations de la partie "Portée limitée au shell". Quelles sont les différences constatées ? Quelles sont les similitudes ?
   (b) Dans une fenêtre shell, créez une variable `force` et initialisez sa valeur à `100`. Exportez cette variable avec la commande `export force`, exécutez un sous-shell dans ce shell en lançant la commande `bash`, et affichez dans ce sous-shell la valeur de `force`. Vérifiez que cette valeur est bien de `100`. Modifiez cette valeur à `150`, fermez le sous-shell avec la commande `exit` (une fois seulement) et affichez à nouveau la valeur de la variable `force`. La valeur a t-elle changé de `100` à `150` ? Que concluez-vous : la commande `export` permet elle de partager entièrement une variable, ou se contente-t'elle de cloner la variable en créant une variable totalement indépendante ayant le même nom et la même valeur ?