TP 3 de familiarisation avec Unix

Redirections – Filtres – Archives

Vincent Berry - vberry@lirmm.fr

Préambule

L'application lisant ce pdf vous permet de prendre des notes : faites-le !

Rappel : si vous découvrez dans ce module l'environnement Unix, vous aurez tout intérêt à éviter les paragraphes dont le titre est précédé d'un symbole ℮ (pour **e**xpert). Ces paragraphes pourront être examinés en dehors de la présente séance. **Si vous connaissez déjà le système Unix, passez par toutes les sections.**

Table des matières

1 Redirections et filtres

La commande `ypcat` [1] permet d'avoir des informations sur le réseau : les login, les groupes d'utilisateurs, les machines... Malheureusement cette commande n'est permise ou ne marche pas bien sur vos machines. Pour palier ce problème nous allons travailler sur le fichier `ycpat_groups.txt` qui correspond au résultat de cette commande sur un autre système : j'ai créé ce fichier par la commande

```
ypcat groups > ypcat_groups.txt
```

Vous trouverez ce fichier dans une archive de fichier-TPs sur le site de votre enseignant. Au fait, comment désactiver ce fichier ? Quel répertoire est créé ?

Dans les questions suivantes, le but est de trouver la commande ou l'enchaînement de commandes qui permet de réaliser la tâche demandée.

1.1 Questions sur ypcat_groups.txt (20mn)

Comment connaître la liste des groupes auxquels appartient un certain utilisateur en passant par la commande `grep` ?	

1. catalogue des pages jaunes – *yellow pages*

Comment connaître le nombre de groupes connus ?	

Trouvez comment obtenir la liste des groupes définis sur ce réseau (sans leurs liste d'utilisateurs). Commande `cut`.	

Idem mais en faisant que cette liste de groupes aille s'inscrire dans un fichier `groupes.txt` trié par ordre alphabétique (un par ligne par exemple) – commande `sort` et redirection de la sortie standard.	

Trouvez maintenant comment indiquer pour un groupe qui nous intéresse le nombre de personnes membres de ce groupe (indication : chercher du côté de la commande awk).	

1.2 Questions sur ypcat passwd (20mn)

Dans les fichiers-TP téléchargés sur le site de l'enseignant, vous disposez aussi d'un fichier `ypcat_passwd.txt` :

Vérifiez le contenu de ce fichier (commande `less`) et repérez la façon dont les lignes sont composées. Est-ce que cette structure se retrouve sur toutes les lignes ?	

Comment utiliser la commande de filtre `egrep` pour localiser dans ce fichier la ligne qui concerne l'utilisateur Flo Mez ?	

Comment envoyer la ligne concernant cette personne dans un fichier `~/UNIX/mesInfos.txt` ?	

Comment ajouter en une commande la date d'aujourd'hui et l'heure à la fin du fichier `mesInfos.txt` (commande `date`) sans recourir à un copier/coller ou à un éditeur ?	

En analysant plus finement le fichier `ypcat_passwd`, comment savoir maintenant combien d'utilisateurs portent le même *prénom* Flo Mez ? (allez voir le manuel de `egrep` pour lui demander de **c**ompter le nombre de lignes du résultat)	

L'étudiant `Paul Riquet` veut savoir combien de personnes portent le même *nom* que lui. Il utilise pour cela l'expression régulière (le motif) suivante : "Riquet". Il obtient en réponse la ligne le concernant plus un certain nombre de lignes non désirées comme `Jean.Riquette` Comment cela se fait-il ?	

Pour parvenir à un résultat correct, il utilise maintenant le motif suivant : `^[A-Z]\.Riquet` Pouvez-vous indiquer pourquoi il utilise un symbole \ devant le caractère "." ?	

Quand il teste sa commande avec le motif indiqué ci-dessus, il n'obtient pas une seule ligne de résultat, même pas celle correspondant à son nom de login. Pouvez-vous indiquer quelle erreur il a faite ?	

Après avoir corrigé son erreur, il obtient bien les lignes souhaitées, mais toujours accompagnées de plusieurs lignes non correctes, comme celle du dénommé `Riquette`. Indiquez comment il doit compléter son motif actuel pour n'avoir que les lignes des utilisateurs ayant exactement le même nom que lui.	

Est-il possible d'obtenir le même résultat correct avec un motif plus court ? si oui, lequel ?	

1.3 Redirections et filtres avancés ☋ (10 mn)

Nous allons maintenant examiner quelques commandes permettant de retrouver des fichiers.

Consultez le manuel de la commande `find`. A quoi cette commande sert-elle ?

Faîtes un essai en vous positionnant dans votre répertoire d'accueil et en lui demandant de localiser le fichier `projet.txt` (utilisez l'option `-name`)

Sans changer de répertoire, demandez à cette commande de localiser dans l'ensemble de l'arborescence du SGF (c-a-d depuis `/`) le fichier exécutable correspondant à la commande `ls` .

Comment filtrer les lignes qui ne nous intéressent pas (correspondant à des répertoires dans lesquels vous n'avez pas le droit d'aller) en se servant de la commande `grep` ?

Comme alternative, il est possible (vérifier que c'est possible dans un shell bash) de rediriger la sortie des erreurs vers le faux fichier `/dev/null` en ajoutant 2>/dev/null ! à la fin de la commande `find`. Faîtes un essai pour trouver le syntaxe exacte et vérifiez que le résultat de la commande est maintenant plus lisible.

Question subsidiaire : pouvez-vous trouver l'option de `find` qui permet de lister tous les fichiers modifiés depuis hier ?

2 Introduction aux archives Unix (≈ 10 mn)

- Dans un Terminal, placez-vous dans votre répertoire d'accueil
- Sans bouger de là (je vous surveille ☺), créez un fichier vide nommé `vide` dans le répertoire `Unix` (commande `touch`).
- Quel est le chemin absolu du fichier `vide` ?
- Qulle commande permet de vérifier que vous avez raison ?
- Créez une **archive** `unix.tar` comprenant tous les fichiers de votre répertoire `UNIX` (commande `tar` avec l'option `c`)
- Vérifiez que le fichier a bien été créé dans votre répertoire d'accueil (`ls` avec le bon argument) et qu'il contient bien les bons répertoires et fichiers (commande `tar` avec l'option `t`)
- Essayez de détruire le répertoire `UNIX`. Que faut-il faire pour y parvenir ?
- Restaurez l'arborescence du répertoire `UNIX` en désarchivant le fichier `unix.tar` : `tar -xvf <nomarchive>`
- Explorez le répertoire restauré ainsi que ses sous-répertoires pour vérifier que tout est bien en place.
- Est-ce que l'arborescence recréée contient bien le fichier `vide` ?
- Quel est le chemin absolu de cet exemplaire du fichier `vide` ? Pouvez-vous justifier cela ?
- Faîtes maintenant une copie du fichier `unix.tar` sur une clef usb (sinon dans un nouveau répertoire nommé `tmp` que vous placerez sous la racine de votre arborescence personnelle).
- Essayez (et réussissez – c'est un ordre !) de restaurer cette copie de l'archive dans le dossier où vous venez de la mettre. Explorez rapidement l'arborescence qui a été créée. Pouvez-vous donner le chemin absolu du fichier `vide` qui devrait s'y trouver ? Pouvez-vous expliquer pourquoi il en est ainsi ?

3 Intégration des notions vues sur le SGF dans une application de taille réelle : l'assemblage d'un projet (50 mn)

Nous allons intégrer l'ensemble des notions vues ci-dessus dans une application type. Vous venez de rejoindre l'entreprise *IG+* où vous avez pour rôle d'assembler le travail des différentes équipes de développeurs sur le projet *Musiversal*.

Mise en place (10 mn)

Récupérez l'archive `Musiversal.tar` (depuis le répertoire commun) ou depuis la page web des TPs. Déplacez l'archive dans votre répertoire TP2. Décompressez l'archive à l'aide de l'explorateur de fichiers (dans le doute sur cette opération, un reflexe : demander le menu contextuel), mais sans explorer le résultat de cette opération.

Fermez juste après la fenêtre de l'explorateur de fichiers (pour la suite de cet exercice, vous n'y avez plus droit du tout). Pour réaliser votre mission sur ce projet, vous allez utiliser l'application `Terminal`, parfois pour vous déplacer dans la hiérarchie des répertoires du projet et parfois pour taper les commandes nécessaire à l'assemblage du projet. Pour distinguer ces deux utilisations, nous allons recourir à la possibilité qu'offre le Terminal d'utiliser différents **onglets** dans la même fenêtre graphique. Explorez cette possibilité en regardant les menus de cette application. Demandez à disposer de deux onglets et nommez-les `Assemblage` et `Exploration` respectivement. Le premier vous servira à effectuer les tâches décrites ci-dessous, le deuxième à vous promenez dans l'arborescence du projet quand vous le jugerez nécessaire.

Dans l'application Terminal, placez-vous dans le répertoire `Musiversal`. Explorez son contenu pendant quelques minutes (commandes `cd,ls,cat,more`).

Pourquoi est-il nécessaire de faire précéder par un symbole "\" l'espace qui apparaît dans le nom de certains répertoires du projet ?	

Le projet *Musiversal* est découpé en modules dont la réalisation a été confiée à trois équipes d'*IG+*, situées dans des villes différentes. Dans chaque centre de développement, une à deux personnes ont travaillé sur le projet, chacune réalisant un module de l'application finale. L'application que vous devez assembler est composée de 5 modules.

A l'aide du fichier `roadmap.lst` et des fichiers de sites `team.txt` déterminez quel développeur est en charge de quel module. Inscrivez la correspondance dans la case de droite.	

Assemblage du code des modules (20 mn)

Chaque module développé est composé de deux fichiers, dont le nom respecte une convention établie dans l'entreprise `IG+` : le `code` se nomme `module.pl` et sa `documentation` se nomme `readme.txt`

Créez un répertoire `bin` situé juste sous le répertoire `Musiversal`	

Tout en restant dans le répertoire `Musiversal`, copiez-y les différents modules de **code** développés dans les centres de la façon suivante : pour chacun utilisez la commande `cp` après avoir éventuellement utilisé la commande `ls` pour vous rappeler la structure de l'arborescence du projet.	

Vérifiez combien de fichiers contient maintenant le répertoire `bin` ? Pourquoi observez-vous ce résultat ?	

Recommencez donc la copie des fichiers de code en ajoutant dans le nom de chacun quelque chose qui le caractérise (le numéro du module, les initiales de son développeur, ...). Evitez les caractères spéciaux dans les noms choisis ainsi que les espaces.	

Pour enchaîner l'exécution des différents modules, le programme de code `running.pl` vous a été fourni, il se trouve dans le répertoire principal du projet. Placez-vous dans ce répertoire principal, puis demandez l'exécution de l'application en tapant `./running.pl` Que se passe-t-il ?	

Remédiez à cette situation en donnant les droits nécessaires pour que toute entité puisse exécuter ce fichier.	

Redemandez l'exécution de ce fichier. L'exécution doit maintenant démarrer, mais être vite arrêtée par l'absence d'un fichier de configuration que vous devez composer :

Avec l'éditeur (x)emacs, créez le fichier de configuration `modules.cfg` dans le même répertoire que `running.pl` : ce fichier de configuration devra contenir les noms que vous avez choisis pour les fichiers de code des modules que vous avez placé dans le répertoire `bin` . Chaque ligne du fichier de configuration ne doit contenir que le nom du fichier module (sans chemin) et doit être clôturée par un passage à la ligne (<`Entrée`>). Les modules doivent être listés dans le bon ordre. Inscrivez dans la case de droite le contenu du fichier de configuration obtenu.	

Demandez une exécution de l'application pour voir si tout est en place. Le programme est conçu pour vous dire si l'un des modules déclaré est manquant ou pas.

Même si le fichier de configuration est bien établi, et que l'application trouve bien le code des modules dans le répertoire `bin`, il manque encore un petit quelque chose pour que l'application fonctionne. Pouvez-vous deviner ce dont il s'agit et y remédier ?	

Bien, maintenant tout doit être en place pour que l'application s'exécute en utilisant le code des différents modules. Faites un essai. Si cela fonctionne correctement, 5 messages doivent apparaître à l'écran, chacun produit par l'exécution d'un des modules, et l'enchaînement des messages doit former une phrase intelligible. Si tel n'est pas le cas, faîtes les corrections nécessaires pour arriver à un résultat correct.

Documentation du projet (20mn)

Occupons-nous maintenant de la documentation du projet.

Créez un répertoire `doc` dans le répertoire `Musiversal`	
Utlisez le `man` sur la commande `ls` pour trouver comment obtenir en une seule commande la liste de tous les fichiers qui se situent dans le projet `Musiversal`	
Repérez où se situent les fichiers `readme.txt` qu'il va nous falloir assembler pour générer la documentation du projet. Quelles caractéristiques communes partagent les emplacements de ces fichiers dans la hiérarchie ?	
En profitant de l'emplacement symétrique de ces fichiers `readme.txt`, on veut maintenant utiliser la commande `ls` mais pour n'obtenir que la liste de ces fichiers. Quelle commande taper ?	
Bien, maintenant que nous savons désigner exactement ces fichiers (sans les autres), nous allons *concaténer* (assembler) leur contenu en un seul fichier `documentation.txt`. Pour cela, utilisez la commande `cat` avec comme argument la désignation calculée ci-dessus.	
Est-ce que les différentes parties de la documentation apparaissent dans le bon ordre ? Pourquoi ?	
Pour palier ce problème, et en vous servant de la correspondance entre numéro de modules et développeurs établie plus haut, vous allez utiliser la commande `cat` (une seule fois) en lui donnant 5 arguments dans l'ordre pour obtenir à l'écran la documentation dans le bon ordre (pour cela, bien-sûr il est recommandé d'utiliser autant que possible la touche de complétion automatique des noms de répertoires et fichiers (<`TAB`>) et d'agrandir la largeur de la fenêtre Terminal à tout l'écran.	
Utilisez l'éditeur de texte (x)emacs pour créer un fichier `documentation.txt` dans le répertoire `doc` dont le contenu est obtenu par un seul copier-coller de la documentation assemblée dans le terminal.	

Vous venez de remplir votre première mission au sein de la société *IG+* : bravo !

4 Manipulation d'archives dans le cadre de l'application (25mn)

La suite de votre contrat pour le projet `Musiversal`, implique d'intégrer de nouveaux modules dans l'application actuelle

- Si ce n'est pas déjà le cas, déplacez le projet `Musiversal` dans votre répertoire `Unix` puis déplacez-vous dans le répertoire `Unix`
- Créez une archive `musiversal-v1.tar` pour sauvegarder le contenu de ce projet et vérifiez ensuite qu'elle contient bien tout ce qu'elle doit (`tar` option `t`)
- Téléchargez maintenant le fichier `neoModule.tar.gz` et placez-le dans le répertoire `Unix`.
- D'après l'extension de ce fichier, sous quel format sont stockées les informations qu'il contient ?
- Décompressez le contenu de ce fichier (`gzip` avec la bonne option). Vérifiez le résultat de cette opération. Combien de fichiers ont été créés ? Le fichier `neoModule.tar.gz` existe-t-il toujours ?
- Demandez à consulter le contenu de l'archive `.tar` que vous avez obtenu (commande `tar` avec la bonne option)
- D'après le chemin indiqué pour les fichiers contenus dans cette archive, quel est le répertoire dans lequel vous devriez placer cette archive avant d'essayer de la désassembler ?
- Après avoir éventuellement déplacé l'archive, désassemblez les fichiers qu'elle contient de façon à ce qu'ils se situent au bon niveau en comparaison des autres modules
- Vérifiez en demandant le contenu des répertoires concernés.
- Donnez les droits d'exécution au fichier programme contenu dans ce nouveau module que vous venez de récupérer.
- Ajoutez le nom du module dans la liste des modules de l'application, placez une copie de son programme dans le répertoire `bin` et lancez une exécution de l'application pour voir si tout fonctionne bien.
- De même, ajoutez la documentation de ce nouveau module à la fin de la documentation actuelle, en utilisant une redirection (`>>`).

Note : si vous rencontrez un problème lors d'une des étapes précédentes, pensez à faire le ménage et à repartir de l'application qui fonctionnait, stockée dans l'archive `musiversal-v1.tar`

Il est question de déployer cette application chez le client, mais l'entreprise ne souhaite livrer que les exécutables et les fichiers de configuration nécessaires.

- Faîtes la liste de ce qu'il est nécessaire d'envoyer chez le client, puis dans le répertoire `UNIX` créez un répertoire de même nom que votre login et envoyez-y une copie des fichiers que vous avez repérés. Recréez exactement la sous-arborescence qui est nécessaire aux exécutables de l'application, c-a-d en omettant le répertoire `Musiversal` lui-même (mais pas tout son contenu) et les répertoires de développement.
- Testez l'exécutable dans ce nouvel endroit. Si cela ne fonctionne pas demandez de l'aide à votre enseignant après avoir examiné le contenu des scripts et essayé de repérer vous même pourquoi votre installation locale ne fonctionne pas.
- Quand cette installation fonctionne, créez une archive `exploitation.tar` qui contient votre installation de l'exécutable (c-a-d le répertoire de même nom que votre login et son contenu). Vous pouvez procéder en une seule exécution de la commande `tar` (en indiquant moults arguments) ou bien en complétant progressivement l'archive (voir le `man` de la commande `tar`), mais faîtes en sorte qu'au désassemblage de votre archive, l'ensemble aille bien dans un répertoire à votre nom.
- Vérifiez que cela va être le cas en demandant la liste et le chemin des fichiers et répertoires de l'archive (`tar` option `t`)
- Compressez maintenant l'archive pour passer pour un vrai pro avant de la transmettre (commande `gzip`).
- Pour vérifier que vous avez fait du bon travail, le patron vous demande de lui envoyer votre archive avant d'envoyer au client. Prudent, vous préférez d'abord tester avec un collègue pour éviter de prendre trop de risques avec le big boss : rédigez un fichier `README.txt` qui contient la commande pour désarchiver votre archive et la commande à lancer ensuite pour exécuter l'application [2], et envoyez un mél à votre voisin de gauche ou de droite avec ce fichier d'explication en attaché, ainsi bien-sûr que l'archive compressée.
- Demandez à votre collègue d'opérer une décompression et un désassemblage de l'archive dans un répertoire `tmp` qu'il aura créé dans son répertoire `Unix`, mais séparément de son répertoire `Musiversal` (afin que l'application que vous délivrez soit vraiment testée en solo, comme sur une machine ne contenant pas l'ensemble du projet

2. Soyez sûrs de bien exécuter la manoeuvre, car il vous faudra la répéter pour rendre un certain nombre de projets durant votre parcours IG

`Musiversal`).

— Après avoir effectué cette vérification Ne quittez pas la salle de TP avant d'avoir montré à votre enseignant (votre chef de service) qu'effectivement votre archive de l'exécutable est opérationnelle chez un collègue [3].

3. autrement dit, en environnement hostile ☺