

TP de familiarisation avec Unix

Filtres – Réseau – Variables – Scripts

Vincent Berry - vberry@lirmm.fr

Table des matières

1	Commandes utilisant les possibilités réseau	1
1.1	Connexion par ftp à un serveur distant (10 mn)	1
1.2	Connexion à une machine Linux (15-20 mn)	1
1.3	Copie de fichiers entre machines distantes (5 mn)	3
2	Variables	3
2.1	Configuration de l'invite de commande (15 mn)	3
3	Retour sur les filtres	4
3.1	Partie extension dans le nom d'un fichier (5-10 mn)	4
3.2	Analyse d'un fichier html par la commande <code>egrep</code> (20 - 30 mn)	5
4	Scripts (20 mn) @	6
4.1	Principe	6
4.2	Initialisation de variable depuis une commande	6
4.3	Gestion d'un paramètre	6

1 Commandes utilisant les possibilités réseau

1.1 Connexion par ftp à un serveur distant (10 mn)

Comme application pour un cours qui démarre bientôt, nous allons chercher de la documentation sur le compilateur du langage Ada que vous utiliserez bientôt dans le cours d'algorithmique.

Dans un Terminal, connectez vous au serveur ftp `cs.nyu.edu`

Une fois connecté, naviguez (commandes `ls` et `cd`) pour vous rendre dans le répertoire `/pub/gnat/papers/` puis récupérez le fichier `gnat-slides.ps.gz`

Déconnectez-vous et décompressez le fichier récupéré. Visualiser son contenu avec l'outil Linux prévu pour les documents postscripts.

1.2 Connexion à une machine Linux (15-20 mn)

La commande `ssh` vue en cours permet de se connecter sur une machine différente de celle en face de laquelle vous êtes assis. Nous allons explorer les possibilités de cette commande.

Dans un Terminal, placez-vous dans votre répertoire UNIX. Créez-y un sous-répertoire Reseaux (en évitant les accents!!!). Placez-vous maintenant dans ce nouveau répertoire.	
---	--

Demandez à voir tous les processus qui vous appartiennent (et seulement ceux-là, pas ceux des autres utilisateurs) avec la commande <code>ps -u votreLogin</code> . repérez le numéro de processus d'une application graphique qui tourne sous votre identité (un navigateur internet, un éditeur emacs par exemple).	
---	--

Repérez le nom de la machine sur laquelle êtes-vous connectez (commande <code>hostname</code>).	
--	--

Déduisez-en le nom d'une des machines à côté de vous. Demandez un nouvel onglet dans le Terminal en cours, nommez-le **Machine Distant**, tandis que l'onglet initial peut être renommé **Machine Locale**. Dans l'onglet **Machine Distant**, connectez-vous sur une machine voisine par la commande `ssh`. Acceptez ensuite la clef identifiant la machine que vous connectez (**yes** à la question) car nous sommes sur un réseau sécurisé par vos administrateurs systèmes, en qui vous avez confiance. La machine contactée vous demande ensuite un mot de passe, que devez-vous indiquer :

- le nom de la machine contactée ?
- le mot de passe de votre dresse `@etud.univ-montp2.fr` ?
- votre nom de login ?
- le mot de passe de votre compte Unix (c-a-d celui utilisé pour vous connecter sur un ordinateur en salle de TP) ?
- le nom de votre machine d'origine ?

Utilisez la logique pour résoudre cette question et la pratique pour disculper tous vos doutes.

Bon, une fois identifié sur la nouvelle machine, sur quel répertoire êtes-vous situé ? Pouvez-vous vous rendre à la racine de votre arborescence personnelle ? Comment cela est-il donc possible ?	
--	--

Demandez la liste des processus qui vous appartiennent sur cette machine distante (même commande <code>ps</code> qu'essayée précédemment). Comment se fait-il que vous ne voyiez pas le processus graphique que vous aviez repéré il y a 5 minutes ?	
--	--

Toujours dans l'onglet Machine Distant Descendez maintenant dans le répertoire Reseaux (que vous avez créé quand vous étiez sur la machine de départ). Créez un fichier <code>distant.txt</code> contenant le nom de la machine distante sur laquelle vous être connecté (commande <code>hostname</code> et redirection de la sortie standard dans un fichier dont le nom est précisé).	
---	--

Repassez dans l'onglet Machine Locale Est-ce qu'il va être possible de voir le fichier <code>distant.txt</code> ?	
--	--

Dans l'onglet Machine Distant demandez qui est connecté sur la machine que vous avez rejoint par le réseau (commande <code>who</code>).	
---	--

Qu'est-ce qui permet que vous soyez en train de taper des commandes sur la machine de votre voisin : <ul style="list-style-type: none">– parce qu'il a abandonné sa machine en se connectant lui aussi sur une autre machine distante ?– parce que le système est multitâches et multi utilisateurs ?	
--	--

1.3 Copie de fichiers entre machines distantes (5 mn)

Dans l'onglet Machine Locale de votre Terminal, placez-vous au dessus du projet Musiversal et créez une archive compressée du projet ne contenant que les fichiers de programmes (sans la documentation ni les autres fichiers) en utilisant la commande <code>tar</code> ; revoquez éventuellement vos notes du TP précédent pour la syntaxe de cette commande.	
--	--

Utilisez la commande <code>scp</code> pour envoyer une copie de cette archive sur un compte dont vous disposez sur une machine distante. Si vous n'en possédez pas en dehors de Polytech, alors vous vous contenterez d'envoyer l'archive à la racine de votre arborescence personnelle mais par l'intermédiaire d'une autre machine de la salle (qui jouera donc le rôle de la machine distante).	
--	--

Vérifiez que la copie a bien eu lieu.

2 Variables

2.1 Configuration de l'invite de commande (15 mn)

Les questions ci-dessous considèrent que vous êtes dans un shell `bash` (la syntaxe d'un autre shell comme `tcsh` peut en effet varier quelque peu).

Affichez le type de shell dans lequel vous êtes (commande `echo $SHELL`). Si vous n'êtes pas en `bash`, vous pouvez-y passer simplement en tapant son nom dans le terminal. Vérifiez ensuite comment la variable `SHELL` a changé.

Dans le Terminal, placez-vous à la racine de votre arborescence personnelle. Remarquez la chaîne de caractères qui s'affiche (le *prompt* ou *invite de commandes*). Déplacez-vous maintenant dans le répertoire `UNIX`. Comment a changé l'invite de commande par rapport à la situation précédente ?

La variable d'environnement qui correspond à l'invite de commande s'appelle `PS1`. Demandez à voir la valeur de cette variable (commande `echo` sans oublier le symbole qui doit précéder le nom d'une variable).

Essayez de modifier cette variable dans le terminal, suivant la syntaxe <code>nomVariable=valeur</code> en indiquant par exemple <code>"coucou>"</code> comme valeur. Changez ensuite de répertoire pour voir comment l'invite s'adapte ou ne s'adapte plus à un tel changement.	
---	--

Demandez un nouvel onglet. Comment est l'invite de commandes ? Pourquoi (attention, il faut indiquer deux raisons qui se combinent) ?	
---	--

Lancer **(x)emacs** grâce au chargeur d'application (appuyez simultanément sur **<ALT>** et **<F2>**)

Ouvrez avec l'éditeur le fichier `.bashrc` s'il existe ou créez le sinon (attention à ne pas oublier le point au début du nom).

Dans ce fichier, trouvez la ligne positionnant la variable `PS1` si elle existe. Positionnez la variable `PS1` de la façon suivante : `PS1='Que faire, Maître ? \W > '`

Sauvez le fichier.

Ouvrez un nouveau terminal (ou un nouvel onglet dans le terminal courant. Que remarquez-vous ? Pourquoi un tel changement ?	
---	--

Nous allons repositionner la variable `PS1` à une valeur plus utile. Demandez l'ensemble des informations qui peuvent être indiquées dans le prompt : commande `man bash` puis cherchez la section `PROMPTING` (respectez la casse) en utilisant les facilités de recherche de l'aide (touche `/`).

Revenez dans l'éditeur **(x)emacs** et indiquez la valeur que vous avez choisi pour la variable `PS1` (attention, l'invite de commandes ne doit pas être trop longue car sinon les commandes que vous saisissez par la suite ne pourront pas s'afficher en entier ce qui sera plutôt pénible).

Faites quelques essais jusqu'à être content de votre invite de commande (essayez de ne pas y passer plus de 10mn).

3 Retour sur les filtres

3.1 Partie extension dans le nom d'un fichier (5-10 mn)

Nous avons brièvement vu dans le TP précédent comment retenir un certain nombre de lignes d'un fichier texte par la commande `egrep`. Nous allons l'utiliser ici dans le cadre plus réel d'un informaticien devant faire l'étude d'une page HTML dans un projet informatique.

Votre enseignant a mis un document intitulé `www.univ-montp2.fr.htm` sur sa page consacrée aux enseignements.

Que se passe-t-il quand vous cliquez gauche sur le lien représentant ce document ? Quelle est l'adresse de la page alors visitée ? Comment expliquer la présence de <code>www</code> à la fois au début et à la fin de l'URL (c-a-d l'adresse) indiquée en haut dans le navigateur ?	
--	--

Revenez sur la page précédente et utilisez cette fois un clic droit sur le lien `www.univ-montp2.fr.htm` pour télécharger le fichier correspondant dans votre répertoire `UNIX`.

Vérifiez dans un terminal quel est le nom du fichier téléchargé sur votre système. Pouvez-vous facilement identifier l'extension dans le nom de ce fichier ? Quelle est-elle ?	
--	--

Visualisez les premières lignes de ce document (commande `head`) dans le terminal pour vous faire une idée de son contenu.

De quel type de fichier s'agit-il ? Après avoir réfléchi, utilisez la commande <code>file</code> pour vérifier votre intuition.	
---	--

Changez maintenant le nom de ce fichier (par une commande dans le terminal) en `exemple.html`¹

¹Les fichiers html conçus sur Windows ont tendance à avoir une extension en `.htm` car ce système d'exploitation utilise traditionnellement trois caractères pour coder le type d'un fichier. Sur Unix, ce n'est pas la norme et on peut trouver des extensions d'à peu près n'importe quelle longueur.

3.2 Analyse d'un fichier html par la commande egrep (20 - 30 mn)

Pour répondre aux questions suivantes, il est conseillé d'ouvrir un 2ème onglet dans la fenêtre Terminal afin d'avoir en permanence à portée de main l'aide du manuel sur cette commande (`man egrep`). On aura aussi intérêt à avoir sous les yeux les diapos du cours donnant les principaux métacaractères utilisables dans un motif recherché par la commande `egrep`.

La structure d'un document html est donnée par la présence de *balises*, sortes de mot-clefs, entre symboles "<" et ">". Par exemple `<head>` ou bien `<tr>`. Comme vous le voyez, le nom de ces balises peut être explicite (*head* signifiant *entête* du document) ou bien abrégé (*tr* signifiant *table row*, c-a-d une ligne d'une table).

Appelons *simple* une balise dont le format respecte exactement la syntaxe suivante : `<nomBalise>` où le nom de la balise est composé de n'importe quelle suite de lettres (mais pas d'autres types de caractères).

Comment affichez les lignes du fichier <code>exemple.html</code> qui contiennent une balise simple ? En cas de difficulté pour établir le motif de <code>egrep</code> , allez-y pas à pas. Par exemple, commencez par identifier les lignes contenant un symbole <code><</code> , etc.	
---	--

L'utilisation de majuscules ou de minuscules est indifférent à la syntaxe des balises.

Comment savoir si votre fichier contient des balise dont le nom est entièrement en majuscule ?	
--	--

Le html est un langage parenthésé, c-a-d qu'en général les balises fonctionnent par paire, comme les parenthèses. La balise ouvrante est de forme `<nomBalise>` est la balise fermante est de forme `</nomBalise>`

Combien de balises simples ouvrantes contient le fichier <code>exemple.html</code> ? Attention : il vous faudra ici utiliser deux options de la commande <code>egrep</code> : celle qui compte les lignes et celle qui rend la recherche insensible à la casse ²	
---	--

Même question pour les balises fermantes	
--	--

Tiens, il y a beaucoup plus de balises ouvrantes que de fermantes. Pour voir à quoi tient ce phénomène étrange, nous allons nous concentrer sur les balises en majuscules (nettement moins fréquente dans le fichier analysé).

Demandez l'affichage des balises ouvrantes simples dont le nom est en majuscule. Demandez ensuite à afficher les balises fermantes dont le nom est en majuscule. Quelle balise fermante n'a pas de balise ouvrante simple correspondante ?	
--	--

Pour explorer ce mystère, affichez toutes les lignes contenant le mot <code>TABLE</code> (sans s'occuper des symboles <code><</code> , ou bien <code>></code>).	
--	--

En voyant à l'écran le résultat de la commande précédente vous comprenez que certaines balises ouvrantes peuvent voir préciser un certain nombre d'*attributs* sous la forme de couples `variable=valeur`, donnant des indications sur le formatage de la zone concernée dans le document html.

Affichez toutes les lignes du fichier contenant une balise ouvrante <i>non-simple</i> , c-a-d contenant des attributs.	
--	--

Enfin, parlons des commentaires. Comme dans la plupart des langages de programmation, il est possible d'indiquer des zones de commentaires dans un fichier html. Tout commentaire doit ici être précédé de `<!--` et terminé par `-->`

Affichez tous les commentaires tenant en une seule ligne. Attention, le caractère "!" est un métacaractère, n'oubliez donc pas de le déspecialiser quand vous l'indiquez dans le motif (c-a-d faites-le précéder d'un <i>backslash</i>).	
---	--

Nous voulons maintenant enlever les lignes inutiles de ce fichier `exemple.html`

Quelle commande permet de créer une version <code>exempleShort.html</code> du fichier où toutes les lignes contenant un commentaire complet ont été supprimée? Indice : demandez les lignes qui <i>ne contiennent pas</i> le motif que vous cherchez (option <code>-v</code>) et redirigez le résultat dans un fichier.	
--	--

4 Scripts (20 mn) @

4.1 Principe

Créez un répertoire `bin` juste sous la racine de votre arborescence personnelle.

Dans ce répertoire, éditez un fichier `sauve.sh` qui contient une seule ligne (suivi d'un retour à la ligne) : sur cette ligne indiquez la commande permettant de créer une archive compressée du répertoire `Unix`. Faites que l'archive soit créée à la racine de votre arborescence.

Bien-sûr vous aurez tout intérêt à procéder par copier-coller d'une commande testée au préalable sur le Terminal de commandes.

Donnez ensuite les droits d'exécution à ce fichier et demandez son exécution.

4.2 Initialisation de variable depuis une commande

Une fois que cet embryon de script fonctionne, nous allons le rendre plus intéressant en ajoutant dans le nom de l'archive la date à laquelle elle a été créée.

Expérimentez dans le terminal la commande suivante : `date "+%d-%m-%y"` L'idée est de se servir du résultat de cette commande comme d'une partie du nom de l'archive. Donc il faut modifier le script pour qu'une partie du nom de l'archive contienne une partie variable. Vous l'avez compris, nous allons passer par une variable intermédiaire.

Pour cela, nous allons définir une variable `D` à laquelle nous allons donner la valeur résultant de l'exécution de la commande de date vue ci-dessus. Pour cela il faut utiliser la syntaxe suivante :

```
nomVar='cmde et ses arguments'
```

où les quotes entourant la commande sont des *backquotes* (c-a-d s'obtiennent par une autre touche que l'apostrophe habituelle). Faites des essais dans le terminal jusqu'à avoir une syntaxe qui permet de positionner correctement une telle variable.

Ensuite, ajoutez une première ligne au script qui définit cette variable `D` et introduisez cette variable dans le nom de l'archive créée à la deuxième ligne du script. Par exemple, l'archive pourra avoir le nom `unix- D .tar.gz`

Après avoir sauvegardé le script, demandez son exécution et vérifiez si une archive au bon nom a bien été créée dans votre répertoire d'accueil.

4.3 Gestion d'un paramètre

Tout au long de l'année, vous n'allez pas vouloir archiver toujours le même répertoire. Ceci veut dire que pour que le script `sauve.sh` devienne intéressant, il faut qu'on puisse lui indiquer en paramètre le nom du répertoire que l'on souhaite archiver.

Pour cela, modifiez le contenu de la commande d'archivage dans le script pour qu'elle intègre un paramètre qui sera passé au script : ce paramètre est accessible sous le nom `$1` (un deuxième paramètre serait `$2` et ainsi de suite).

Faites un essai et vérifiez le contenu de l'archive créée (option `t` de la commande `tar` ; bien-sûr n'oubliez pas cette fois en demandant l'exécution du script de lui donner un argument séparé de son nom par un espace, comme il se doit).