Costa da Quinta, Joao Filipe

TP1 - Systémes informatiques

exo 2.1)

a)

<$ man man>

Cette commande explique à l'utilisateur comment il peut utiliser le manuel à son avantage, ce manuel est disponible à tout moment! Pour apprendre à le naviguer, on doit appuyer sur la touche h (help)

<$ man echo>

Echo correspond tout simplement à la commande utilisé pour afficher une ligne de texte, en lisant le manuel on apprend qu'on peut utiliser avc d'autres paramètres pour avoir diffèrents types d'affichage.

<$man head>

Permet d'afficher le début (les 10 premières lignes par défaut) d'un fichier en ou group de fichiers, si c'est plusieurs il affiche aussi les noms de chaque fichier pour pouvoir les distinguer, le manuel nous suggère aussi la commande <$ tail>, qui fait la même chose que head, mais plûtot que montrer le début du fichier, montre la fin.

b)

'bash how to merge two files'

En tapant ca sur un moteur de recherche on trouve assez facilement la commande : <$ cat [file1.txt] [file2.txt] >[./filedestination.txt]> qui copie [file1.txt] suivi de [file2.txt] dans [./filedestination.txt], si [./filedestination.txt] existe pas, il est crée, s'il existe déjà, il est formaté, et reprend les nouvelles informations. Il est aussi possible d'utiliser la commande <$ paste [file1.txt] [file2.txt] >[./filedestination.txt]> qui met le contenu de [file2.txt] à droite du contenu de [file1.txt] dans [./filedestination.txt].

exo 2.2)

Comme expliqué dans l'exo 2.1a) echo permet de afficher une ligne de texte, mais elle peut faire plus que tout simplement afficher un message. Par exemple, <$ echo "salut je suis" | wc > wordcount.txt> wc compte le nombre de lignes, de mots, et bytes dans "salut je suis" et enregistre ensuite le

résultat dans wordcount.txt. <$ echo> prend comme paramètres [short option] [string], si on utilise -e qui permet l'interpretation de backlash escapes, on pourra formater le texte autrement <$ echo -e "salut \nes"> imprime: salut(saut à la ligne)es.

exo 2.3)

Pour créer un simple fichier vide, on utilise la commande <$ mkdir [nom du fichier]> qui crée un fichier nomé [nom du fichier] dans le répertoire dans lequel on se trouve. pour le visualiser, on peut faire <$ ls && mkdir [nom du fichier] && ls> on aura d'abbord la liste du contenu dans le repertoire, ensuite on cree le ficher [nom du fichier] et finalement dans une nouvelle ligne le contenu dans le dossier suite à la création du fichier.

exo 2.4)

Je travaille sur un environement windows avec WSL, où la commande <$ nautilus> ne marchait pas "Unable to init server..", sur internet j'ai appris que c'est une commande "plûtot drole que utile", qui initialise une fenetre graphique du bureau à distance, en lanceant la machine virtuelle de l'unige j'ai remarqué que effectivement suite à <$ nautilus> une fenettre de bureau était affiché.

exo 2.5)

a)

<$ wc -w <foo.txt>

Il faut lire ceci de droite vers la gauche, on prend le fichier [foo.txt] (il doit exister dans le répertoire où on se trouve), et on utilise ce fichier comme input pour wc qui compte le nombre de lignes, mots et bytes, vu qu'on rajoute l'extention -w, on aura que le nombre de mots dans [foo.txt] comme output.

<$ cat foo.txt | wc -w>

J'ai déjà expliqué ce qu'elle fait <$ cat> dans exo2.1b), ici vu que on met que un fichier apres <$ cat> le output sera le fichier lui meme, mais plutot que l'enregistrer dans un autre fichier, l'usage de <|> rend ce output, le input de wc -w qui, comme avant, va print le nombre de mots dans le fichier [foo.txt]. Par contre, vu qu'on utlise <$ cat>, on aurait pu faire <$ cat foo.txt [fichier2.txt] | wc -w> qui va concatener les deux fichiers, et comtper le nombre de mots des 2 fichiers.

<$ wc -w foo.txt>

Ici on utilise la fonction wc avc le paramétre -w, qui va donc donner le nombre le nombre de mots qui se trouvent dans le fichier apres -w, ici [foo.txt].

Les trois commandes ont le même output, mais elles ne le font pas de la même facon.

b)

Comme expliqué dans exo2.1a) <$ head foo.txt> affiche les 10 (par défaut) premières lignes de [foo.txt] pour, alors que <$ tail foo.txt> affiche les 10 dernières lignes. En rajoutant le paramètre <-n 6> on aura que les 6 premières/dernières lignes respectivement.

c)

Si [foo.txt] existe, le output dans [out1.txt] est [foo.txt] ordonné alphabétiquement, et [out2.txt] est vide.

si [foo.txt] n'existe pas, le output dans [out1.txt] est vide, et [out2.txt] est le texte suivant "sort: cannot read: foo.txt: No such file or directory"

d)

<$ evince foo.pdf> ouvre le fichier en question en avant plan, c'est à dire qu'on a pas accés à au terminal (on peut pas rentrer des commandes).

<$ evince foo.pdf &> ouvre aussi le fichier mais en arrière plan, on peut donc rentrer des commandes dans le terminal sans problème.

exo 2.6)

Mon script est répartie entre 4 fonctions, tout d'abbord il faut créer le repoertoire de destination des images que s'il existe pas, <$ mkdir -p> fait exactement ca.

En suite on copie les images, et seulement les images, la boucle for nous assure qu'on regarde tous les dichiers dans le dossier de départ, et ensuite le if qui vérifie que chaque fichier est bien une image, grace à <$ file --mime-type -b fichier> si ce fichier en question est une image, on aura la réponse <image.*>, et si c'est bien une image, on la copie dans le dossier d'arrivée.

Il faut aussi renomer les images (nettoyer lerus noms), fait à l'aide de <$ mv oldname newname> qui en fait supprime le fichier oldname et le recrée avec le nom newname, grace à <$ tr -d "*"> on enleve au oldname le symbole entre les guillemets.

Pour terminer, on change le format de l'image et la résolution si on a un 3eme argument, un if vérifie précisément son existance, s'il exisque on suit la commande qui convert avc resize, s'il existe pas, on modifie juste l'extention.