TP n°1

- 1) Quelles sont les propriétés principales d'un système UNIX
- 2) Quel est votre répertoire de connexion (ou de travail)
- 3) Lister le contenu de votre répertoire
- 4) A l'aide du manuel (« man ») et en observant le résultat des lignes de commande suivantes expliquer ce que font les commandes suivantes :
  - logname,
  - id,
  - tty,
  - ls,
  - who,
  - pwd,
  - cd,
  - cd..,
  - ls -la,
  - cat ,
  - date,
  - wc,
  - cal,
  - which,
  - touch,
  - grep

TP n°2 :

« L'arborescence du système de fichiers d'UNIX »

Exercice n°1 :

Construisez la sous arborescence suivante sous « rep » :

- Dans « rep » vous créez le répertoire « rep1 » et les fichiers « .c », « a » et « b ».
- Dans « rep1 » vous créez le répertoire « rep2 » et les fichiers « a1 », « b1 » et « .c1 ».
- Dans « rep2 » vous créez le répertoire « rep3 » et les fichiers « a2 », « b2 » et « .c2 ».
- Enfin dans « rep3 » vous créez le fichier « a3 ».

Donner les résultats de : « \$ ls ; ls . ; ls .. ; ls -a ; ls -a . ; ls -a .. »

Exercice n°2 :

- Placez-vous dans le répertoire « rep » et créer le fichier de nom « .fich1 ». Remonter d'un niveau dans l'arborescence (« \$cd .. ») puis effectuer les deux commandes « rmdir » et « ls » qui produiront les résultats suivants :

```
$ rmdir rep
```

```
rmdir : rep not empty
```

```
$ ls -l rep
```

```
total 0
```

Exercice n°3 :

Pour définir le chemin d'accès à un fichier et par conséquent son nom, nous disposons de deux techniques complémentaires qui sont le « nommage absolu » et le « nommage relatif ».

Expliquer ces deux notions que vous illustrerez en utilisant la commande « cd » en donnant deux chemins différents qui modifient votre répertoire courant pour devenir celui qui correspond au répertoire d'accueil d'un autre étudiant.

Exercice n°4 :

Créez un répertoire "temp1" sous le répertoire de travail.

- Créez un fichier "test1" sous le répertoire "temp1".
- Créez un nouveau répertoire "temp2".
- Copiez le fichier "test1" vers le répertoire "temp2".
- Supprimez le répertoire "temp1".
- A l'aide d'une seule directive, dupliquez "temp2" sur un nouveau répertoire "trav", tout en se débarrassant de "temp2".
- Renommez le fichier test1 en lui donnant "virus" comme nouveau nom.
- Trouvez votre login, votre nom du groupe, votre uid et votre gid. Pour cela consultez les fichiers "/etc/group" et "/etc/passwd"

TP n°3 :

« La politique d'accès aux fichiers d'UNIX »

Exercice n°1 :

`-rw-rw-r-- 1 c1 cours884 Jul 7 Fich1`

Quelles commandes permettent à l'utilisateur « c1 » de modifier les droits d'accès sur le fichier « Fich1 » pour avoir :

`-rwxr-xr-x 1 c1 cours884 Jul 7 Fich1`

Donnez une solution numérique et une solution avec les actions (cf. le cours).

Exercice n°2 :

Créez un répertoire WORK1 sous le répertoire WORK.

Analysez ses droits d'accès.

Quelles sont les permissions que possède votre voisin?

Exercice n°3 :

Modifiez les permissions d'accès sur WORK1 pour ne permettre aux trois catégories réunies (user, group et others) que la possibilité de lecture (utilisez la notation absolue « le signe = »). Prouvez vos résultats.

Exercice n°4:

Utilisez la notation symbolique pour restaurer tous les droits d'accès (r,w,x) pour le propriétaire, tout en les enlevant au groupe et aux autres.

Exercice n°5 :

Créez un nouveau fichier appelé « droits ». Placez les valeurs de ses droits sur les valeurs suivantes.

Pour ce faire, utilisez aussi bien la forme symbolique que l'octale.

`rw-rw-r--`

`rw--w----`

`rwxr-x--x`

`r-x-w---x`

Exercice n°6 :

Créez un nouveau répertoire dans votre répertoire courant et appelez-le « testcat ».

Créez dans ce répertoire un fichier « testdat ».

Modifiez les droits d'accès sur le répertoire selon la liste ci-dessous. Essayez, pour chaque combinaison des droits d'accès, de passer dans le répertoire, de lancer la commande ls dans « testcat » et de modifier le fichier « testdat » avec l'éditeur vi. Notez à chaque fois les actions possibles. la liste des autorisations d'accès est la suivante:

`rwX`

`rw-`

`r-X`

`--X`

`-wX`

Exercice n°7 :

Faites cadeau d'un fichier vide à votre voisin.

Essayez en suite de supprimer ce fichier. Que remarquez-vous?

Pouvez-vous reprendre d'une manière ou d'une autre, la propriété de ce fichier?

Demandez à votre voisin de manipuler les droits d'accès de ce fichier. est-ce qu'il réussit à effectuer cette opération?

Demandez lui de vous rendre votre fichier.

Exercice n°8 :

Créez une copie de WORK/bidonne, nommée bidon1 sur le répertoire WORK1 par la méthode des liens. Vérifiez que les deux fichiers bidone et bidon1 se partagent la même I-NODE.

Exercice n°9 :

Dupliquez bidon1 pour obtenir bidon2 sur le même répertoire WORK1, à l'aide de la commande **cp**. Est ce que les deux fichiers se partagent les mêmes I-NODE.

Exercice n°10 :

Analysez la manipulation suivante:

```
$chmod -w bidon2
```

Comparez les droits d'accès des fichiers bidon1 et bidon2.

Que remarquez-vous? Pourquoi ?

Exercice n°11:

A l'aide d'une simple manipulation (en utilisant la redirection >), copier la liste des utilisateur actuels sur le fichier bidonne.

Exercice n°12 :

Affichez sur votre écran les fichiers bidonne, bidon1 et bidon2.

Commentez.

Exercice n°13:

Quelle est la différence entre les deux appels suivants:

```
$cp fichier1 fichier2
```

```
et $cat <fichier1 > fichier2.
```

Dans les deux cas, le fichier fichier2 n'existe pas encore au moment de lancement de la commande.

TP N°4

« L'éditeur de texte vi »

Présentation de « vi »

L'éditeur de texte « vi » ne fait pas partie des éditeurs les plus évolués et les plus conviviaux. Par contre c'est un standard sur tous les systèmes UNIX, c'est-à-dire que quelque soit la version et le fournisseur du système que vous utilisez, « vi » est implanté. Il est très puissant pour travailler sur des fichiers systèmes (application de filtres ...) et ne nécessite que très peu de ressources. En outre les directives de « vi » sont souvent utilisables dans des commandes UNIX (less, more, sed ...).

Commandes générales pour démarrer, quitter et annuler.

Lancement de l'éditeur (le nom est optionnel) **vi** < nom_fichier >

Sauvegarde (le nom est optionnel) **:w** < nom_fichier >

Sortie

sans modification et sans sauvegarde **:q**

après modification et sans sauvegarde **:q!**

avec sauvegarde **ZZ** ou **:x** ou **:wq**

Divers

annuler la dernière commande **u**

répéter la dernière commande **.**

exécuter n (par ex. 10) fois une commande (par ex. dd) **10dd**

réafficher l'écran **CTRL L**

appliquer un filtre (par ex. trier par ordre alphabétique les lignes de 5 à 10 dans le texte) **: 5 , 10 ! sort**

Directives

Une directive c'est un ou plusieurs caractères. Une fin de directive c'est soit une fin implicite, soit la touche **ESC**.

1. Directives avec fin implicite

\$ positionnement en fin de ligne

yy copie la ligne courante dans un tampon

nyy copie n lignes dans un tampon

p copie le tampon après la ligne courante

x suppression du caractère courant

dw suppression du mot courant

dd suppression de la ligne courante

/chaîne recherche de l'occurrence d'une chaîne

2. Directives avec sortie par **ESC**

i insertion avant le curseur

a ajout après le curseur

Mode commande

On accède au mode commande (dernière ligne de l'écran) en tapant le caractère « : ». Les commandes autorisées sont :

toutes les commandes de sauvegarde et de sortie vues précédemment

n1 : aller à la ligne n1

\$: aller à la dernière ligne

d : détruire la ligne courante

n1,n2d : supprimer les lignes n1 à n2

m n : déplacer la ligne courante en ligne n

n1,n2 m n3 : déplacer les lignes de n1 à n2 en ligne n3

t n : dupliquer la ligne courante en ligne n

n1,n2 t n3 : dupliquer les lignes de n1 à n2 en ligne n3

s/chaine1/chaine2/ : substitution dans la ligne courante de la première occurrence de chaine1 par chaine2

s/chaine1/chaine2/g : idem que précédemment mais sur toutes les occurrences de chaine1

n1,n2s/chaine1/chaine2/g : idem pour les lignes de n1 à n2 ; ex. :
1,\$s/avoir/être/g

r : lire un fichier et l'insérer à la ligne courante

!commande : appeler une commande UNIX

Exercices

Placez vous dans une fenêtre sous le shell. Lancer l'éditeur « vi ». Après y avoir inséré le texte suivant, suivez les directives de modification du texte données ci-dessous :

Cette étude a pour but de faciliter l'accès à des serveurs applicatifs distants depuis le monde

Internet W3 en utilisant les techniques des systèmes répartis à objets.

Cet outil sera étendu afin d'améliorer la qualité des interfaces HTML générées, l'architecture

d'accès aux seveurs applicatifs (utilisation de CORBA) ainsi que les performances. L'utilité

de l'outil sera démontré par la mise en place d'un accès Internet à la simulation de quelques

serveurs : serveur de banque à domicile (ex : vidéoposte), serveur de commande (ex : LA

REDOUTE), serveur de réservation (ex : réunion téléphonique). Il s'agit de développer un

outil permettant aux fournisseurs d'application de disposer rapidement d'une interface utilisateur en HTML permettant l'accès à leurs applications à partir d'un serveur W3. L'outil générera, à partir d'une description formelle de l'interface programmatique de l'applications (IDL CORBA), les formulaires HTML constituant l'interface utilisateur et le code permettant l'accès du serveur W3 au serveur applicatif lorsque l'utilisateur final remplit ces formulaires.

Déplacer le deuxième paragraphe pour le mettre en fin de texte.

Commande : _____

Sauvegarder le contenu dans un fichier de nom « TDediteur » sans quitter « vi ».

Commande : _____

Insérer le texte suivant entre l'avant dernier et le dernier paragraphe : « On réutilisera une première version d'outil fournie par ANSA ».

Commande : _____

Utiliser la commande de substitution pour remplacer partout dans le texte « W3 » par « World Wide Web ».

Commande : _____

Ajouter en fin de fichier le contenu du fichier /etc/passwd.

Commande : _____

Utiliser la commande de substitution pour remplacer partout dans le texte « /bin/bash » par « /bin/csh ».

Commande : _____

Compter en utilisant les commandes de recherche de chaîne le nombre d'occurrence du mot « de ».

Commande : _____

Appeler la commande « man vi » sans quitter l'éditeur.

Commande : _____

Quitter l'éditeur en sauvegardant le contenu.

Commande : _____

Vous voulez utiliser les directives de déplacement (par exemple « \$ ») mais le caractère \$ s'affiche dans votre texte. Pourquoi et que faire ?

Vous venez de détruire une ligne par erreur, que faire ?

Commande :

TP N°5

Exercice 1

Contrairement aux processus lancés simultanément qui s'exécutent sans relation entre eux, des processus concurrents sont synchronisés entre eux par la production d'information de l'un, et la consommation d'information de l'autre. Il est nécessaire pour cela, que le processus producteur soit apte à produire des caractères sur la sortie standard, et que le processus consommateur soit apte à lire des caractères sur l'entrée standard.

C'est ce que réalise la commande « pipe » dont la syntaxe est :

commande1 | commande2

Exemple : **\$ ps -alx | grep bash**

« ps » produit une liste de processus ; le pipe sert de raccordement de la sortie standard du processus « ps » sur l'entrée standard du processus « grep », qui va lui-même n'afficher que les lignes contenant la chaîne « bash ».

1. Exécuter la ligne de commande de cet exemple.
2. Créer un fichier texte de nom « text1 » dont le contenu est :

1 : la commande *pipe* sert à la communication entre processus

3 : la commande *tee* est utile pour capturer les informations qui circulent dans un pipe

2 : la commande *tee* recopie son entrée standard sur sa sortie standard et sur un fichier

4 : la commande *tee* peut être utilisée pour sauvegarder dans un fichier les traces des informations qui circulent sur sa sortie standard.

1 : qu'est-ce qu'un *pipe* et que fait la commande *tee* ?

- i. en utilisant la commande « | » écrire une commande qui compte le nombre de ligne contenant le mot « pipe » dans le fichier « text1 ».
- ii. Regarder dans le manuel la commande « tee ». En utilisant cette commande modifier le résultat obtenu en i) afin de récupérer dans un fichier « text2 » les lignes contenant le mot « pipe ».
- iii. Ecrire une ligne de commande qui permettent de créer un fichier « text3 » qui contiendra les lignes du fichier « text1 » contenant le mot « pipe ». Ces lignes devront être triées sur le premier champ de chaque ligne. Enfin la commande affiche le nombre de ces lignes.

Exercice 2

- a. Combien y a-t'il de processus actifs sur le système ?
- b. Combien d'utilisateurs sont connectés sur le système ? En utilisant la commande « sort », afficher la liste de ces utilisateurs connectés, triée :
  - i. par ordre alphabétique.
  - ii. selon l'heure de connexion.
- a. Combien le répertoire « /etc » a t'il de fichiers répertoire ? Utiliser les commandes « ls », « grep » et « wc ».
- b. Afficher la liste des fichiers du répertoire courant, triée par ordre de taille des fichiers.

Exercice 3

1. Utiliser la commande « cat » et l'opérateur « > » pour créer les fichiers « fich1 » et « fich2 ». Toujours en utilisant la commande « cat » mais cette fois en regardant le manuel créer le fichier « fich3 » constitué de la concaténation des fichiers « fich1 » et « fich2 ».
2. Lancer la commande « cat fich1 fich-inexistant » avec le fichier « fich-inexistant » inexistant et le fichier « fich1 » existant.
3. Nous pouvons rediriger la sortie standard en utilisant l'opérateur « > » ; par exemple « cat fich1 fich-inexistant > trace » ou
« cat fich1 fich-inexistant 1>trace ».

Lancez les deux commandes ; que constatez-vous ?

4. Lancez la commande du a) en redirigeant la sortie d'erreur dans le fichier « err ». Comment peut-on rediriger la sortie standard sur la sortie d'erreur ?

TP N°6 :

Programmation Structurée Shell

EX 1:

Ecrire un programme shell qui affiche le nombre de ses paramètres, la liste de ses paramètres ainsi que son PID.

EX 2:

Ecrire un programme shell « listefich » qui accepte comme paramètre un nom. Ce programme doit vous renseigner sur le type du fichier dont le nom est spécifié, ses autorisation d'accès pour l'exécuteur et si se fichier est vide ou pas?

Ex 3:

Créer un lien entre deux fichiers et testez-le.
Que remarquez-vous ? pourquoi ?

EX 4:

Ecrire les procédures de test nécessaires qui traduisent les phrases suivantes:

a- « Vrai si fichier1 ou fichier2 accessibles en lecture et fichier3 accessible en exécution. »

b- « Vrai si fichier1 et fichier2 sont accessibles en lecture ou ecriture et fichier3 est un répertoire. »

c- « Vrai si fichier1 est accessible en exécution, fichier2 est accessible en lecture et fichier3 est accessible écriture. »

d- « Vrai si fichier est accessible en lecture et exécution, mais non accessible en écriture.»

e- « Vrai si fichier1 est plus récente que fichier2, mais moins récente que fichier3.»

(N'utilisez pas ni "ot", ni "nt" non plus.)

f- «Vrai si deux nombres sont égaux .»

g- « Vrai si fichier existe .»

h- « Vrai si fichier1, fichier2 et fichier3 son liés .» (N'utilisez pas "ne".)

Ex 5:

Ecrire la négation de :

```
$test \(! $a=$c -o ! $b=$c)
```

EX 6:

Ecrire un programme pour tester si le fichier en donnée est un fichier ordinaire répertoire ou spécial.

EX 8 :

Ecrire une commande lsdire qui affiche la liste des sous répertoire d'un répertoire donné (par défaut le répertoire courant).

EX 9:

Ecrire une nouvelle version de lsdire qui peut admettre plusieurs arguments. On affichela liste des sous répertoires des répertoire donnés comme arguments.

Ex 10:

Créer un programme shell `dos_sh`, pour mettre en place, pour l'utilisateur une interface similaire à celle de MSDOS.

Nous souhaitons en particulier que le prompt affiche le chemin actif et que les commandes puissent être exploitées.

COMMANDE	SIGNIFICATION
help	Affiche le texte d'aide
del fichier	Supprimer le ou les fichiers avec confirmation
edit fichier	Editer un fichier
dir	Affichage du répertoire actif (en tenir compte des options /p et /w)
copy fichier1 fichier2	Copier un fichier
type fichier	Afficher le contenu d'un fichier
ren fichier	Renommer un fichier

TP N°7 :

Programmation Structurée Shell

EX 1 :

Ecrire un programme qui calcule la moyenne, le produit et la somme d'une série de valeurs finissant par zéro.

EX 2 :

Créer la commande **crefich** obéissant à la syntaxe suivante:

\$crefich nom quantité

Son rôle est de créer un ensemble de fichiers appelés nom1, nom2, ... , nomN. La création de chaque fichier doit être validée en interactif par l'utilisateur.

EX 3 :

Ecrivez un script permettant de positionner toutes les autorisations d'accès de tous les fichiers normaux du répertoire actif sur la valeur 640. Si un paramètre est indiqué, l'opération ne portera que sur le fichier correspondant. En absence de paramètre, tous les fichiers normaux du catalogue actif doivent être traités.

EX 4 :

Créez un script appelé « **supprime** ». Vous lui confierez des noms de fichiers et pour chaque fichier, il s'agira de vérifier que ce sont des fichiers normaux. Tous les autres types ne sont pas acceptés. Pour les fichiers permis, la question devra être posée s'il faut les supprimer ou non. Une réponse **oui** entraînera la suppression de ce fichier.

EX 5 :

Etendez le script précédent « **supprime** » pour pouvoir également supprimer des répertoires. Si un nom de répertoire est contenu dans les paramètres, il faut demander si ce répertoire est à supprimer avec tous les fichiers et sous répertoires qu'il contient.

EX 6 :

Ecrire un programme shell qui affiche la N^{ième} ligne d'un fichier donné. Ce programme doit admettre comme premier paramètre le nom du fichier et comme deuxième paramètre le numéro de la ligne à afficher.

EX 7 :

Ecrire une autre version de **rm** qui déplace les fichiers vers un répertoire temporaire Au lieu de les détruire immédiatement et une autre commande qui vide périodiquement ce répertoire.

EX 8 :

Ecrire une commande **lstree** qui affiche la liste des répertoires du sous arbre courant.

EX 9 :

Ecrire une commande **lsdir** qui peut admettre plusieurs arguments et qui affiche la liste des sous répertoires et des fichiers des répertoires donnés comme argument.

TP N°8 :

Les Filtres

Avant de commencer ce TP, créez un nouveau répertoire "tp6" et créez dans ce répertoire les fichiers suivants:

un, deux, trois, quatre, sept, Dix, dix, Treize, Quatorze et Trente-sept.

EX1 :

Transférez la sortie de la commande ls à la commande grep et recherchez les lignes ayant les caractéristiques suivantes:

- le texte "ua" est placé à un endroit quelconque de la ligne,
- le texte "ze" est en fin de la ligne,
- le texte "Dix" ou "dix" est seul dans la ligne,
- le texte "sept" est dans la ligne, en minuscule ou en majuscule.

Affichez à chaque fois les numéros de lignes.

EX2 :

Redirigez la sortie de la commande ls vers un fichier D.

Recherchez en même temps le mot "treize" et le mot "Quatorze" dans le fichier D. Il y a deux solutions, les quelles?

EX3 :

Copiez le fichier "/etc/passwd" dans votre répertoire courant. Dans ce fichier, cherchez la ligne commençant par votre nom d'utilisateur.

EX4 :

Dans le fichier de mots de passe, cherchez les lignes commençant par un nom devant le double point. Ces noms doivent être d'une longueur de 4 ou 5 caractères.

EX5 :

Utilisez grep pour construire un annuaire automatique qui donne pour chaque nom donné le numéro de téléphone et l'adresse.

EX6 :

Créez un fichier sortdat qui contient plusieurs lignes (au moins 5 lignes) de la forme :
nom prénom code postale ville numéro de téléphone

- triez ce fichier d'après les prénoms,
- effectuez le tri de sortdat sur les codes postaux, (pensez au tri numérique)
- effectuez un tri sur les villes en ordre décroissant,
- effectuez un tri sur les deuxièmes et troisièmes lettres du prénom en ordre décroissant.
- effectuez un tri d'après le deuxième nombre du numéro de téléphone. En cas de nombres égaux, vous effectuerez un tri secondaire sur le premier nombre.

EX7 :

En une seule commande, triez en arrière plan le contenu du répertoire de connexion sur le champ taille du fichier tout en sauvegardant la liste en entrée sur un fichier sauve1.

EX8 :

Créez une commande qui permet de chercher le nombre de lignes contenant un mot donné. Le mot cherché et le nom de fichier sont passés en paramètres.

EX9 :

Utilisez `grep` et `sort` pour afficher la liste des fichiers ordinaires du répertoire de connexion trié sur les numéros des I-NODES.

TP N°9 : Les Filtres find et awk

EX1 :

Recherchez, a partir du répertoire de connexion, tous les fichiers dont les noms correspondent au critères suivants:

- longueur de trois caractères,
- commençant par une majuscule,
- terminant par ".c" ou ".f",
- contenant un chiffre.

EX2 :

Recherchez, dans toute l'arborescence, les fichiers qui vous appartiennent.

EX3 :

Recherchez, à partir de votre répertoire de connexion, tous les fichiers normaux auxquels vous n'avez pas accédé depuis deux semaines.

EX4 :

Ecrire un script qui recherche tous les fichiers vides du répertoire courant et qui offre le choix de suppression de chacun des fichiers trouvés.

EX5 :

Introduire dans un fichier **ABC** la ligne suivante:

```
awk '{print}' $1
```

Quel est l'effet de l'appel de script **ABC** avec le nom du fichier fich1 comme paramètre?

A quelle commande peut-on comparer ce programme awk?

EX6 :

Quelle sont les lignes retournées par ce programme?

```
awk 'NR%2==1 {print}' $1
```

EX7 :

Ecrire un programme **awk** permettant d'afficher la taille totale des fichiers dans le répertoire courant.

Le résultat doit être de la forme :

```
Fich1 taille1
```

```
Fich2 taille2
```

```
..
```

```
..
```

```
..
```

```
Fichn taillen
```

```
Total : taille totale
```

EX8 :

Ecrire un programme **awk** qui cherche le fichier ayant la taille la plus grande dans l'arborescence du répertoire de connexion.

EX9 :

Afficher toutes les lignes du fichier de mot de passe contenant un numéro d'utilisateur pair

(UID).

EX10 :

Ecrivez un programme **awk** permettant de retourner à l'écran un fichier, mais en sens inverse.

EX11 :

Ecrire un programme **awk** permettant de déterminer le nombre de mots, le nombre de lignes et le nombre de caractères composant un fichier donné.

EX12 :

Ecrire un programme **awk** qui remplace les tabulations par des espaces.

EX13 :

Ecrire un programme **awk** qui remplace un mot par un autre dans un ou plusieurs fichiers.