ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE | **ANNEE UNIVERSITAIRE 2020/2021**

EPREUVE DE	: S.Expl. & Mise en Oeuvre UNIX	DATE	:01 Avril 2021
CONTROLE	: Devoir Surveillé	CLASSE	: II1
DUREE	: 1 heure 30	NB. PAGES	: 4
DOCUMENTS	: non autorisés		
ENSEIGNANTS	: I.Amdouni, F.Ben Abdallah, C.Houaidia, H.Idoudi, M.Sellami		

NOM & PRENOM : …… CLASSE :……

Exercice 1 : QCM (0.75*8= 6 pts)

Cocher la ou les bonnes réponses

1. Les applications communiquent avec le noyau en utilisant ….. ?
   - ☐ Le shell
   - ☐ Le script shell
   - ☐ Un programme C
   - ☐ **Des appels système**
2. Lequel représente le répertoire personnel de l'utilisateur?
   - ☐ /
   - ☐ .
   - ☐ ..
   - ☐ **~**
3. Un système _____ permet à plusieurs utilisateurs _____.
   - ☐ à temps partagé : d'exécuter leurs programmes en même temps
   - ☐ à temps partagé : de partager la mémoire
   - ☐ **à temps partagé : d'avoir l'illusion de programmes s'exécutant en même temps**
   - ☐ de traitement par lots : de faire du temps partagé
   - ☐ aucune des combinaisons précédentes
4. Il y a deux liens physiques vers le fichier « file1 » qui sont « p1 » et « p2 » et un lien symbolique « s1 ». Que se passe-t-il si nous supprimons le fichier « file1 »?
   - ☐ **Nous pourrons toujours accéder au fichier avec p1 et p2 mais pas avec s1**
   - ☐ Nous ne pourrons pas accéder au fichier avec p1 et p2 mais avec s1
   - ☐ Nous pourrons accéder au fichier avec n'importe p1, p2 et s1
   - ☐ Nous ne pourrons pas accéder au fichier avec p1, p2 et s1
5. Les informations d'attribut d'un fichier, sous UNIX, sont stockées dans quelle structure sur le disque?
   - ☐ **Inode**
   - ☐ Blocs de données
   - ☐ Blocs de fichiers
   - ☐ Fichier de répertoire
6. Lequel des éléments suivants n'est pas un type de fichier valide sous Linux?
   - ☐ Liens symbolique
   - ☐ FIFO
   - ☐ **Inode**
   - ☐ Socket
7. Un _____ est _____.
   - ☐ processus : un programme
   - ☐ **processus : une abstraction d'un programme en exécution**
   - ☐ processus : un programme sur le disque
   - ☐ processus : une copie des valeurs des registres
   - ☐ aucune des combinaisons précédentes
8. Un utilisateur exécute la commande suivante : **$ chmod + x myfile.txt**, Lequel des énoncés suivants est vrai pour la sortie de cette commande?

- ☐ La commande entraîne l'ajout d'une permission d'exécution à l'utilisateur qui a exécuté cette commande
- ☐ La commande entraîne l'ajout d'une permission d'exécution pour le propriétaire du fichier
- ☐ La commande entraîne une erreur car le fichier n'est pas un fichier exécutable
- ☐ **La commande entraîne l'ajout d'une permission d'exécution pour tous (utilisateur, groupe et autres)**

Exercice 2 : Répondre aux questions suivantes : (0.5+1+1.5+1.5=4.5 pts)

1. Sous UNIX, quel répertoire standard renferme la majorité des fichiers de configuration des services système et réseau ?

/etc ……

2. Donner l'équivalent numérique (octal) de chaque permission :
   - r-------- : ……400……
   - rw-r--r-- : ……644……
   - r-x--x--- : ……510……
   - rwxrwxrwx : ……777……
3. Expliquer que font les commandes suivantes :
   - `ls | grep '^..$'`

   imprime les noms des fichiers constitués de deux lettres exactement
   - `ls | grep '.*m.*m.*'`

   imprime les noms de fichiers qui Contiennent au moins deux lettres «m»
   - `ls –l | grep '^….r.x'`

   ……
4. En utilisant grep, écrire une ligne de commande qui permet de chercher les lignes qui commencent et se terminent par le même mot. Sachant qu'un mot est constitué uniquement de lettres (minuscules ou majuscules) suivi d'un espace.

…… grep '^\([a-zA-Z;. ,]*\) .*\1$' f.txt ……

……

Exercice 3 (4 pts):

Écrire un script qui affiche le plus grand de tous ses arguments; en s'assurant que tous les arguments doivent être des nombres. Si un des arguments n'est pas numérique, il faut sortir du script en signalant l'erreur.

```
if [ $# -eq 0 ]                                   0.5pt  +   0.5pt (forme du script)
then
    echo "usage: $0 p1 p2 ..p5"
    exit -1
fi
max=0
for i in $*                          0.5pt
do
    echo "param:" $i
    expr $i + 1 2> /dev/null   0.75pt
    #echo "code:" $?
    if [ $? -eq 0 -o $? -eq 1 ]
    then
        echo "$i: Numerique"
        if [ $i -gt $max ]                0.75pt
        then
            max=$i
        fi
    else
        echo "$i: Non Numerique"   0.5pt
        exit -1
```

```
    fi
done
echo "Le maximum est :" $max   0.5pt
```

Exercice 4 (5.5 pts)

On se propose d'écrire un script shell (`Gannuaire`) pour gérer l'annuaire des agents d'une société. Cet annuaire comporte une série de lignes, chacune composée de 5 champs : nom, prénom, numéro de téléphone, bureau et service. Les champs sont séparés par le caractère deux-points comme le montre l'exemple ci-après :

```
Zahrouni:Fatma:4781:D311:perf
Belaid:Jamel:4762:B309:sys
```

Le script shell est appelé avec une seule option (–n, –a, –d , -l ou -m) pour réaliser chacune des fonctions ci-après :

1. rechercher un inscrit à partir de son nom ou d'une partie seulement de son nom et sans distinction des majuscules/minuscules (`#./Gannuaire –n fichier_annuaire`)
2. ajouter un nouvel inscrit (`#./Gannuaire –a fichier_annuaire`)
3. supprimer un inscrit (`#./Gannuaire –d fichier_annuaire`)
4. lister le personnel d'un service (`#./Gannuaire –l fichier_annuaire`).
5. afficher la liste des inscrits sous la forme Nom:Prénom (`#./Gannuaire –m fichier_annuaire`)

En outre, le programme doit respecter les consignes ci-après :

- tester la syntaxe d'appel : il faut 2 arguments : le nom du fichier annuaire et au maximum une option. Tester si le fichier annuaire existe et est accessible en lecture
- pour l'ajout d'un inscrit, demander interactivement à l'utilisateur de saisir successivement chacun des champs et demander la confirmation avant d'enregistrer la nouvelle ligne en fin de fichier et s'assurer que l'inscrit n'existe pas.
- pour la suppression/recherche d'un inscrit, demander à l'utilisateur de saisir le nom de l'agent à rechercher/supprimer (premier champ).
- pour lister le personnel d'un service, demander à l'utilisateur de saisir le nom du service (dernier champ).

```
ajout()
{
read -p"Nom : " nom
read -p"Prenom : " prenom
read -p"Numero de telephone : " tel
read -p"Bureau : " buro
read -p"Service : " serv

echo "$nom:$prenom:$tel:$buro:$serv"
echo "Confirmez-vous ? (o/n)"
read reponse
case "$reponse" in
o|oui)
    echo "$nom:$prenom:$tel:$buro:$serv"
    grep -i "$nom:$prenom:$tel:$buro:$serv" "$Annuaire" > t.tmp
    if test -s t.tmp
    then
            echo "Utilisateur existe deja"
    else
            echo "$nom:$prenom:$tel:$buro:$serv" >> $Annuaire
            echo Utilisateur $nom enregistré
    fi
    ;;
n|non)      echo "Information NON enregistrée"
    ;;
esac
```

```
}


if [ $# -lt 2 ]
then            echo $#
      echo "Syntaxe : $(basename $0) [-opt] Nom_Annuaire"
      exit
else
      eval Annuaire=\${$#}
      if [[ ! -e "$Annuaire" || ! -r "$Annuaire" ]]
      then
                  echo "\"$Annuaire\" devrait être le nom d'un fichier annuaire lisible"
                  exit
      fi
case $1 in
-n) if [ $# -ne 3 ]
      then      echo "Syntaxe : $(basename $0) -n nom Nom_Annuaire"
                  exit
      else
                  #grep -i "^[^:]*$2[^:]*:" "$Annuaire"
                  grep -i "$2" "$Annuaire"
      fi;;

-a) ajout ;;
-d) echo $#
      if [ $# -ne 3 ]
      then    echo "Syntaxe : $(basename $0) -d Nom Nom_Annuaire"
            exit
      else
            if grep "^$2" $Annuaire >/dev/null 2>&1
            then
                  grep -v "^$2" $Annuaire >/tmp/ann.$$
                  mv $Annuaire ${Annuaire}.old
                  mv /tmp/ann.$$ $Annuaire
                  echo $2 supprime
            else
                  echo "L'utilisateur $2 n'est pas inscrit"
            fi
      fi
      ;;
-l) if [ $# -ne 3 ]
      then    echo "Syntaxe : $(basename $0) -l Nom_service Nom_Annuaire"
                  exit
      else
                  #grep "^[^:]*:[^:]*:[^:]*:$2.*:" $Annuaire #| cut -d: -f1|sort
                  grep "$2" $Annuaire
      fi;;

-m) cut -d: -f1,2 $Annuaire ;;
*)    echo "Option ($1) inconnue";exit;;
esac
fi
```