Centre de Tele-enseignement Universitaire

UNIVERSITÉ DE FRANCHE-COMTÉ

Licence 3 – Année 2019-2020

Système et Réseau

Durée : 3 heures – **Session** 1 du 23 Janvier 2020

Ce sujet comporte : 8 page(s)

Matériel autorisé : Aucun document

Expliquez et détaillez les réponses aux questions de l'examen. Si cela est nécessaire, précisez les hypothèses que vous avez posées pour résoudre les exercices : il n'est pas possible de poser de question pendant l'épreuve et il peut parfois exister plusieurs interprétations différentes pour une question. Le barème est donné à titre indicatif.

Exercice 1 : Cours *(3 points)*

Question 1 : Comment sont déterminés les droits d'exécution d'un processus ? Justifier votre réponse à l'aide d'un exemple.

Question 2 : Qu'est-ce qu'une variable d'environnement, donner sa principale particularité. Citez deux variables d'environnement.

Question 3 : A quoi sert le fichier `/etc/passwd` ? Donner des exemples d'informations contenues dans ce fichier.

Question 4 : Comment trouver l'adresse de sous-réseau lorsqu'on ne connait que l'adresse IP d'une machine et le masque de sous-réseau ?

Exercice 2 : Ecriture de commandes *(3 points)*

Question 1 : Donner une commande qui ajoute le droit de lecture au groupe sur tous les fichiers ayant l'extension *tex* du répertoire *Texte* de l'utilisateur *Durand*.

Question 2 : Donner une commande qui affiche les processus s'exécutant sur la machine courante dont le nom contient la chaîne de caractères *bash*.

Question 3 : Donner une commande qui déplace tous les fichiers dont le nom commence par *VS* du répertoire courant dans le répertoire d'accueil de l'utilisateur *RENE*.

Question 4 : Donner une commande qui affiche les noms de login des utilisateurs connectés triés par ordre alphabétique.

Exercice 3 : Interprétation de script shell *(3 points)*

En utilisant le script bash ci-dessous :
— commentez chaque ligne ou groupe de lignes en vous aidant des numéros de lignes,
— proposez un nom explicite pour chacune des variables `D`, `S`, `F`, `Q` et `R`,
— résumez la fonction du script en quelques lignes.

```
1  #!/bin/bash
2
3  if [ $# -ne 1 ]; then
4     echo "Usage : $(basename $0) rep_questions"
5     exit 1
6  fi
7
8  if [ ! -d $1 ]; then
9     echo "Usage : $(basename $0) rep_questions"
10    exit 2
11 fi
12
13 D="/users/$(whoami)"
14 if [ ! -d $D ]; then
15     mkdir $D
16 fi
17
18 S=$(date +%Y-%m-%dT%H-%M)
19
20 I=0
21 for F in $1/*
22 do
23     I=$((I+1))
24     Q=$(tr '[A-Z]' '[a-z]' < $F)
25     R=""
26     while [ -z $R ]; do
27         echo "Question $I : $Q ? O/N"
28         read R
29         case $R in
30             "O"|"N")
31                 echo "$Q = $R" >> "$D/$S"
32                 ;;
33             *)
34                 echo "Non valide !"
35                 R=""
36                 ;;
37         esac
38     done
39 done
40 exit 0
```

Exercice 4 : Ecriture de Script shell *(6 points)*

Une entreprise reçoit une fois par mois les déclarations d'heures supplémentaires de ses succursales de Strasbourg et Bordeaux. Elle souhaite pouvoir traiter en une seule fois la totalité de ses déclarations à l'aide de son logiciel de paye *logicPaye*.

Chaque succursale envoie un fichier dont le nom est de la forme `PAYS.VILLE.seq` où seq est un numéro de séquence attribué par le logiciel de transfert. (Par exemple : FR.STRASBOURG.02158).

On suppose que l'entreprise ne reçoit que deux fichiers de déclarations d'heures supplémentaires par mois.

Question 1 : Script 1

Le but de cette question est d'écrire un script shell *traitementHeuresSup* appelé lors de la réception d'un fichier de déclaration d'heures supplémentaires. Lorsque les deux fichiers de déclarations de Strasbourg et de Bordeaux ont été reçus, le logiciel de paye *logicPaye* peut être exécuté.

Ecrire le script shell *traitementHeuresSup* qui effectue les opérations suivantes :

— prend en paramètre le nom complet du fichier (chemin absolu) reçu.
— affiche le nom de la ville de provenance du fichier reçu.
— si c'est le premier fichier reçu, stockage de son contenu dans le fichier `concatHeuresSup` du répertoire courant et suppression du fichier reçu.
— si un fichier a déjà été reçu, concaténation avec le fichier `concatHeuresSup` du répertoire courant et appel de *logicPaye* avec ce fichier. Si l'exécution de *logicPaye* se termine sans erreur, suppression du fichier reçu et du fichier *concatHeuresSup* sinon retour et affichage d'une erreur.

Question 2 : Script 2

Nous voulons ajouter les fonctionnalités suivantes au Script 1, **si** le fichier reçu provient de Strasbourg :

— enlever la première ligne du fichier reçu qui est une ligne d'entête
— ajouter un deux points `" :"` à la fin de chaque ligne du fichier

Ecrire les lignes de script réalisant ces fonctionnalités et indiquer où les ajouter dans le Script 1.

Question 3 : Script 3

Nous poursuivons l'ajout de fonctionnalités au Script 1. Supposons que chaque ligne de fichier est de la forme :

`nom : service : heures :` où *service* peut prendre les valeurs *S1*, *S2* ou *S3*.

Pour chaque ville (chaque fichier), nous voulons :

— effectuer la somme des heures supplémentaires par service
— afficher ces sommes accompagnées de la ville de provenance du fichier

Ecrire les lignes de script réalisant ces fonctionnalités et indiquer où les ajouter dans le Script 1.

Exercice 5 : Réseau *(5 points)*

Yoda confie à Luke le soin de réorganiser le réseau informatique des Jedi. L'Ordre Jedi comprend 3 rangs : Padawan, Chevalier et Maitre. Chaque Jedi possède un Databloc qui doit avoir une adresse unique sur le réseau.

Selon les analyses de Luke, l'Ordre compte 120 Padawans, 176 Chevaliers et 12 Maitres. En suivant les ordres de Yoda il doit proposer un découpage satisfaisant aux contraintes suivantes :

— chaque rang disposera de son sous-réseau avec son propre routeur,
— tous les rangs seront reliés par un sous-réseau d'inter-connexion unique,
— chaque sous-réseau devra être dimensionné au plus juste et le découpage sera fait par ordre de taille décroissante.

Question 1 : Combien de sous-réseaux seront nécessaires dans cette nouvelle organisation ? Justifiez.

Question 2 : En détaillant vos calculs, donnez la taille de chaque sous-réseau à réaliser.

Question 3 : L'administration du réseau galactique HOLONET propose à Luke de lui attribuer comme base de découpage l'une des 2 plages suivantes : `172.16.0.0` ou `210.168.40.0`.

1. Quelle est la classe de chaque adresse ? Justifiez.
2. Compte tenu des contraintes de l'énoncé, laquelle de ces plages Luke devrait-il retenir ? Argumentez votre réponse.

Question 4 : Yoda impose à Luke d'utiliser le réseau d'adresse `172.16.0.0` pour son découpage.

1. En utilisant la division euclidienne, convertissez l'adresse du réseau en binaire en détaillant vos calculs.
2. Effectuez le découpage en sous-réseaux. Pour chaque sous-réseau donnez, en justifiant vos calculs :
   — son adresse en notation CIDR,
   — l'adresse de diffusion,
   — la première et la dernière adresse **disponible** (et non pas utilisée).

Annexes SR

Options de la commande test :

option	paramètres	description
-e	fichier	vrai si fichier existe
-w	fichier	vrai si fichier existe et est autorisé en écriture
-r	fichier	vrai si fichier existe et est autorisé en lecture
-x	fichier	vrai si fichier existe et exécutable
-d	fichier	vrai si fichier existe et est un répertoire
-f	fichier	vrai si fichier existe et n'est pas un répertoire
-s	fichier	vrai si fichier existe et a une taille non nulle
-t	descripteur	vrai si descripteur est associé à un terminal

TABLE 1 – Test sur fichiers et les répertoires

option	paramètres	description
-z	chaine	vrai si chaine est vide
s1 = s2	s1, s2	vrai si s1 et s2 sont identiques
s1 != s2	s1, s2	vrai si s1 et s2 sont différentes

TABLE 2 – Test sur les chaînes de caractères

option	paramètres	description
n1 -eq n2	n1, n2	vrai n1 = n2
n1 -ne n2	n1, n2	vrai n1 != n2
n1 -gt n2	n1, n2	vrai n1 > n2
n1 -lt n2	n1, n2	vrai n1 < n2
n1 -ge n2	n1, n2	vrai n1 >= n2
n1 -le n2	n1, n2	vrai n1 <= n2

TABLE 3 – Test sur les nombres

Exemple de contenu d'un fichier `/etc/passwd`

```
alain::110:500:Alain Terrieur:/home/alain:/bin/bash
paul::120:500:Paul Hissont:/home/paul:/bin/zsh
```

Exemple de contenu d'un fichier `/etc/group`

```
etu::500:alain,paul
pro::510:alain,paul
exp::520:alain
```

Exemple de résultat de la commande `who`

```
user    :0       Dec 5  18:41
alain   pst/2    Nov 10 20:52
```

Commandes	**Description**	**Exemples**
at	exécution d'une commande à l'heure donnée	at 1023
cal	donne le calendrier	cal 2019
cat	affiche le contenu d'un fichier -n : précède chaque ligne par son numéro	cat fich cat -n fich
cd	changement de répertoire	cd rep
chmod	change les droits d'accès d'un fichier -R : changement récursifs aux sous-rép.	chmod u+x fich chmod -R a+x rep
comm	affiche les lignes communes à deux fichiers	comm fich1 fich2
cmp	compare deux fichiers et retourne le code 0 s'ils sont identiques	cmp fich1 fich2
cp	recopie un fichier -r : recopie de répertoire -i : demande avant d'effectuer une recopie	cp path1 path2 cp -r rep1 rep2 cp -i fich1 fich2
cut	affiche une partie de chaque ligne d'un fichier -c : par numéros de colonnes (1 à 3 et 8) -f : par champs séparés par des délimiteurs -d*c* : où *c* donne le caractère délimiteur	 cut -c1-3,8 fich cut -f1,2 fich cut -f1 -d. fich
date	affiche la date courante (format long) format année-mois-jour	date date +%y-%m-%d
df	donne l'espace libre dans les systèmes de fichiers	df
du	donne le nombre de blocs disques utilisés par un fichier ou un répertoire	 du rep
diff	donne les différences entre deux fichiers -i : majuscules = minuscules	diff fich1 fich2
echo	affiche une chaîne -n : affiche la chaîne sans nouvelle ligne	echo blabla echo -n blabla
find	recherche d'un fichier dans l'arborescence à partir du chemin *path* *-print* : affiche le résultat *-user* : nom du propriétaire *-group* : nom du groupe *-size* : taille en blocs *-exec* : commande à exécuter pour chaque fichier trouvé *-atime n* : nb jours depuis dernière consultation *-ctime n* : nb jours depuis création *-mtime n* : nb jours depuis dernière modification *Ex :* Suppression des fichiers de nom commençant par test ou de nom essai, créés aujourd'hui `find . \(-name "test*" -o -name essai\) -ctime 0 -exec rm {}\;`	find path -name fich1 find path -mtime 7
grep	recherche une chaîne dans un fichier -i : ignore les différences entre maj. et min. -n : numérote les lignes -v : sélectionne les lignes ne contenant pas chaîne -w : sélectionne les lignes où chaîne est un mot	grep chaine fich grep -i chaine fich
gzip gunzip	compresse un fichier décompresse un fichier compressé par gzip	gzip fich1 gunzip fich1.gz
head	affiche les premières lignes d'un fichier	head -n5 fich
hostname	donne le nom de la machine courante	
kill	tue un processus	kill pid
ls	liste le contenu du répertoire -a : affiche aussi les fichiers commençants par . -R : affiche aussi le contenu des sous-répertoires -i : donne le numéro d'inode	ls rep ls -a ls -R ls -i fich
mkdir	création d'un sous-répertoire	mkdir rep
more	affiche un texte page par page	more fich
mv	renomme un fichier ou répertoire	mv fich1 fich2

Commandes	**Description**	**Exemples**
nl	numérote les lignes d'un fichier	nl fich
nice	modifie la priorité d'un processus	nice +1 prgm
passwd	permet de changer le mot de passe (yppasswd)	passwd user
ps	liste les processus -a ou -e : donne la liste de tous les processus -u : donne le nom des propriétaires -o col1,col2,... : affiche uniquement les colonnes spécifiées	
pwd	affiche le nom du répertoire courant	
rm	efface un fichier -r : efface un répertoire (ou rmdir si répertoire vide) -f : force en cas de droits d'accès limités -i : demande confirmation	rm fich rm -r rep rm -rf rep rm -ri rep
seq	affiche la séquence de nombres demandée, séparés par un espace	seq 4 10
set	sans option ni argument : liste les variables du shell permet de positionner les paramètres positionnels Ex : *$1* prend la valeur a, *$2* la valeur b et *$3* la valeur c	 set a b c
sleep	attente d'un délais	sleep 10
sort	tri d'un fichier, basé sur les codes ASCII -r : tri inversé -d : tri uniquement sur les caractères, chiffres et espaces -i : ne trie que les codes ASCII entre 32 et 126 -f : sans différence entre min. et maj.	sort fich
tail	donne les dernières lignes d'un fichier	tail -n 10 fich
tar	permet de combiner plusieurs fichiers en une archive, sans compression -c crée une archive -A concatène à une archive existante	tar -c -f archive.tar fich1 fich2
tee	écrit sur un fichier et sur l'écran dans une suite de pipe	cat fich1 |tee fich2
time	donne les temps d'exécution d'une commande	time du -s
tr	transformation sur l'entrée standard -d : efface les caractères donnés	tr ch1 ch2 tr -d xyw
tty	donne l'identificateur du terminal	tty
wc	donne le nombre d'octets, mots et lignes d'un fichier -l : uniquement le nombre de lignes -w : uniquement le nombre de mots -c : uniquement le nombre d'octets	wc fich wc -lc fich
who	donne les utilisateurs connectés login terminal date de début de connexion	
whoami	donne le nom de l'utilisateur courant	
;	séparateur de commandes	date ; who
(commande)	séquence de commandes qui n'affectent pas le niveau courant	(cd /etc ; ls)
<	redirection de l'entrée depuis un fichier	read a < fich
>	redirection de la sortie sur un fichier	date > sortie
>>	ajoute la sortie à la fin du fichier	
&	exécute une commande en arrière plan	

Synthèse des différences d'interprétation des caractères spéciaux :

	Génération de noms de fichiers	**Expressions régulières**
?	un caractère quelconque, sauf <new line>	remplace 0 ou 1 fois le caractère qui précède
.	le caractère point	un caractère quelconque sauf <new line>
*	0 ou un nombre quelconque de caractères	remplace 0 fois ou n fois le caractère qui précède
`[a-i]`	un caractère entre a et i	un caractère entre a et i
`[!a-i]`	un caractère qui n'est pas entre a et i	un caractère entre a et i, ou `!`
`[^a-i]`	un caractère qui n'est pas entre a et i	un caractère qui n'est pas entre a et i
\	banalise le caractère qui suit	banalise le caractère qui suit
`^`	le caractère `^`	ce qui suit est en début de ligne
`$`	référence une variable	ce qui précède est en fin de ligne