EISTI	INGENIEURS 1ERE ANNE[illegible] EXAMEN SE 17-01-201[illegible]
Rédigé par : équipe pédagogique du cours de Syst. D'expl.	Script shell et fonctions d'un OS
A l'intention de : Etudiants d'ING1	Tout document manuscrit est autorisé

Exercice 1 : Script shell -

Question 1 : Écrire un script shell qui prend en paramètre un fichier formaté comme indiqué ci-dessous et affiche tous les livres d'un auteur passé en paramètre.

Question 2 : Prendre en compte l'option -h

Question 3 : Tester tous les cas possibles

Exemple de fichier (bibliotheque):

Leon Shklar & Rich Rosen @ Web Application Architecture
Bernard Desgraupes @ LaTeX : apprentissage, guide et référence
Vipul Kashyap & Leon Shklar @ Real World Semantic Web Applications
Ernest E. Rothman & Brian Jepson & Rich Rosen @ Mac OS X for Unix Geeks (Leopard)
Bernard Desgraupes @ Passeport Pour Unicode

Exemple d'appel (en gras l'appel, en italique la réponse du script) :

./script bibliotheque Desgraupes

Desgraupes
Passeport Pour Unicode
LaTeX : apprentissage, guide et référence

./script bibliotheque Shklar

Shklar
Web Application Architecture
Real World Semantic Web Applications

Exercice 2: script shell

Écrire un script shell qui affiche un paramètre sur 3 de la lignes de commande.

Exercice 3 : système de fichiers

L'indexation de bloc mémoire dans un système de fichier EXT2 est définie de cette manière :*

- Les 10 premiers champs pointent chacun sur 1 bloc de données ;
- Le champ 11 (simple indirection) pointe vers 256^1 blocs de données ;
- Le champ 12 (double indirection) pointe vers 256^2 blocs de données ;
- Le champ 13 (triple indirection) pointe vers 256^3 blocs de données.

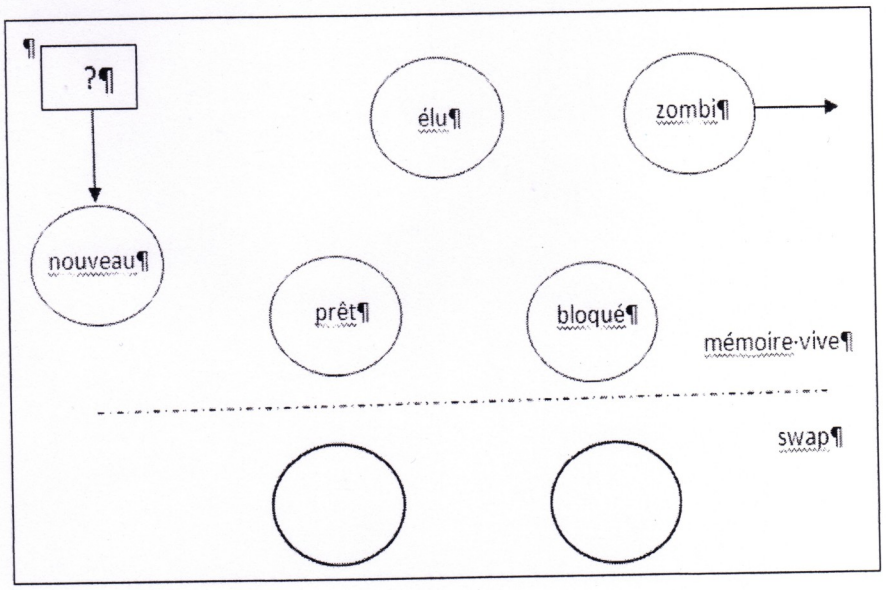
1. Comment peut-on calculer la taille maximale d'un fichier dans ce système de fichier EXT2 ?
2. Pour une taille d'un bloc de 1kB, quelle est cette taille maximale

EISTI – Ecole Internationale des Sciences du Traitement de l'Information	INGENIEURS 1ERE ANNEE EXAMEN SE 17-01-2012
Rédigé par : équipe pédagogique du cours de Syst. D'expl.	Script shell et fonctions d'un OS
A l'intention de : Etudiants d'ING1	Tout document manuscrit est autorisé

Exercice 4 : processus

Soit le schéma ci-dessous, les cercles représentent les différents états d'un processus

1. Donner le nom des 2 états manquants
2. En une phrase expliquer pour chacun de ces états comment et pourquoi un processus s'y trouve
3. Comment sont reliés ces états entre eux (sens des flèches), qualifier chaque relation (nom)
4. Remplacer le point d'interrogation par le nom de la fonction C sous linux



Exercice 5 : gestion de la mémoire

1. Définir ce que c'est un défaut de page
2. Où stocker les pages délogées de la RAM ?
3. Expliquer ce que c'est la partition de swap et le fichier de swap et donner les avantages et les inconvénients de chacun

Processus.

Création :

Création : appel système int fork ();
Crée tous les processus
Notion de père / fils
Notion de PID (unique)
PID 0 : crée au démarrage avant init

Contexte d'un processus :

Environnement utilisateur : zone programme (code à exécuter) : partageable entre plusieurs processus
- zone données (variables) : possibilités d'utilisation du segment de mémoire partagé
- zone pile utilisateur (pour les variables locales et les appels de fonctions) : modifiable dynamiquement

Environnement matériel :
- Compteur ordinal
- pointeur de pile
- registre de travail
- registre de données
- registre pour la mémoire virtuelle

Environnement système
- Table des processus
- Zone u : donné privé au processus uniquement manipulable par le noyau
- Table des régions (permet l'utilisation de mémoire partagée)

Etat d'un processus sous Unix
- Naissance d'un processus : mi prêt, mi endormi ; état initial de tous processus
- Prêt mais le swapper doit le transférer en mémoire centrale pour le rendre éligible
- Exécution en mode utilisateur
- Exécution en mode noyau
- Eligible (prêt à exécuter)
- Endormi en mémoire centrale
- Endormi en zone de swap (sur disque par exemple)
- Zombie : réalisation d'un exit ; uniquement dans la table des processus où il est conservé le temps pour que son processus père de récupérer le code de retour et d'autres informations de gestion (coût de l'exécution sous forme de temps, et d'utilisation de ressources)

états possibles

Ordonnanceur

Processus init

lancement du noyau : démarrage du processus init
Père de tous les processus : PID 1
Lecture du fichier /etc/inittab
Informations de la forme :
id : runlevel : action : processus
id : identifiant
runlevel : niveau de démarrage
action : manière de lancer le processus
processus : ce qui est lancé

Manière de lancer le processus :

respawn : relance le processus une fois celui-ci terminé
wait : attend la fin du processus avant de continuer
boot : doit être lancé au démarrage uniquement
ctrlaltdel : processus lancé quand séquence tapé

Ordonnanceur de bas niveau (CPU scheduler) :
utilisation de l'un des algorithme précédent
aux processus résidant en mémoire centrale
Ordonnanceur de haut niveau (medium term scheduler)
- retire de la mémoire les processus qui y sont restés assez longtemps
- transfère en mémoire des processus résidant sur disque

BIOS : Contenu dans le ROM
lit le premier secteur de la partition activée
Disque MBR
Unix : charge un loader
LILO : LInux LOader
GRUB : GRand Unified Bootloader
Doivent charger le système

POSIX : Portable Operating System Interface X

Normes sur de nombreux choses (interface logicielle, utilisateur : ligne de commande standard et l'interface script est le Korn shell. Fonctions qui devaient exister ... pour avoir la norme.

Dans les systèmes d'exploitation, l'ordonnanceur désigne le composant du système d'exploitation qui choisit les processus qui vont être exécutés par les processeurs d'un ordinateur
Permet à tous les processus de s'exécuter et d'utiliser le processeur de manière optimale du point de vue de l'utilisateur.

Algorithme : PCTE (plus court temps d'exécution) :
inverse du temps d'exécution
- plusieurs travaux d'égale importance se trouve dans une file
- élection du plus court
- temps d'attente moyen minimal
Si toutes les tâches sont présentes dans la file d'attente au moment où débute l'assignation

Shell

Definition d'une variable :

```
maVariable="une variable"
maVariable2=5%
echo $maVariable affiche la valeur de maVariable
```

Substitution dans les chaînes de caractères :

```
>$ maVariable="variable"
>$ echo "Voici une $maVariable" affiche Voici une variable
```

Pour non substitution remplacer " " par ' '

Stockage du résultat d'une commande dans une variable

```
maVariable='ls -l | grep -e /^d/'
```

Trois types de chaînes

- chaîne littérale : '
- chaîne substituée : "
- chaîne exécuté : `

Troncatures :

basename : enlève le chemin ou l'extension d'un fichier
dirname : garde le chemin
ex: basename /etc/apache2/http.conf
http.conf
dirname /etc/apache2/http.conf
/etc/apache2/

Troncatures avancées :

${var#mod} : suppression de la plus courte sous chaîne à gauche
${var##mod} : suppression de la plus longue sous chaîne à gauche
${var%mod} : suppression de la plus courte sous chaîne à droite
${var%%mod} : suppression de la plus longue sous chaîne à droite
${var:ind} : " " les caractères de 0 à ind d'une chaîne
${var:ind:nb} : extrait nb caractères à partir de ind (mettre ind pour partir de la fin)
${#var} : donne la longueur de la chaîne
${var/mod/} : supprime la première occurence de mod dans var
${var//mod/} : supprime toutes les occurences de mod dans var
${var/mod1/mod2} : remplace la première occurence de mod1 par mod2
${var//mod1/mod2} : remplace toutes les occurences de mod1 par mod2

Notation

- Négation : ! expr
- Conjonction : expr1 -a expr2
- Disjonction : expr1 -o expr2

Test sur les fichiers

- -e fic : existence du fichier
- -s fic : existence d'un fichier non vide
- -f fic : existence du fichier ordinaire
- -d fic : existence du répertoire
- -r fic : existence du droit de lecture
- -w fic : existence du droit d'écriture
- -x fic : existence du droit d'exécution

Test sur les nombres :

- n1 -eq n2 : =
- n1 -ne n2 : ≠
- n1 -lt n2 : <
- n1 -le n2 : ≤
- n1 -ge n2 : ≥
- n1 -gt n2 : >

Alternatives multiples

```
if [ cond ]   (espace avec un espace de chaque côté de cond)
then
...
elif [ cond ]
then
...
else
...
fi
```

Aiguillages multiples

```
case var in
cas1) ... ;;
cas2) ... ;;
*) ... ;;
esac
```

(on peut mettre des ou "Chien" | "Chat" | "Souris")

Boucle

```
for var in liste_valeur
do
...
done
```

```
for variable in 'valeur1' 'valeur2' 'valeur3'
do
done
```

(On peut utiliser une variable à la place de valeur)

Test en tête

```
while [ cond ]
do
...
done
```

marche avec des ou

. ^ début de ligne

. . un caractère quelconque

$ fin de ligne

. x* zéro ou plus d'occurence du caractère x

x+ une ou plus occurences du caractère x

x? une occurence unique du caractère x

[...] plage de caractère permis

[^...] plage de caractère interdit

\{n\} par défaut le nombre de répétitions n du caractère placé devant

Ex: [a-z]* recherche d'occurence des lettres permises

^[0-5]\{4\}$ a pour signification, du début à la fin du fichier $, recherche les nombre [0-5] de 4 chiffres \{4\}

grep: -v affiche les lignes ne contenant pas la chaîne

-c compte le nombre de ligne contenant la chaîne

-n chaque ligne contenant la chaîne est numérotée.

-x ligne correspondant exactement à la chaine.

-l affiche le nom des fichiers qui contiennent la chaine.

find <répertoire de recherche> <critère de recherche> -print

- nom recherche sur le nom du fichier.
- type recherche sur le type (d=rep, c=cara, f=fichier)

head -n 9 fichier.txt | tail -n 5 prends les 5 dernières des 9 premières

Script basename

```
#!/bin/bash
STRING="$1"
STRING2="$2"
STRING3="${STRING##*/}"
if [ -z $STRING2 ]
then
echo "${STRING##*/}"
else
echo "${STRING3%.*}"
fi
```

Script qui affiche un paramètre sur 2

```
#!/bin/bash
let "a=$#"
let "b=$#%2"
let "c=0"
let "d=$a/2+1"
for i in `seq 1 $d`;
do
echo "$1"
shift
shift
done
```

Script qui applique une commande sur une liste de fichier et donne l'aide avec -h

```
#!/bin/bash
Commande="$1"
case $2 in
-h)
let "a=$#-1"
echo "-------"
shift
for fichier in `seq 2 $a`
do
$Commande $2
shift
done;;
*)
for fichier in `seq 2 $#`
do
$Commande $2
shift
done;;
esac
```

Pour passer de décimal à binaire :

77 | 2
1 | 38 | 2
0 | 19 | 2
1 | 9 | 2
1 | 4 | 2
0 | 2 | 2
0 | 1

Ce sont les restes à chaque fois

77 = 1001101

1 2 3 4 5 6 7

3 2 3 4

3 4

4.

Commandes Unix

cd : permet de changer de repertoire courant
r: lecture du fichier / Consultation de la liste des fichiers
w: modification du fichier / Ajout / Suppression de fichiers
x: execution du fichier / Positionnement dans le repertoire

ls : permet de lister des fichiers
-l : permet de visualiser les droits des fichiers
-i : permet de visualiser les i-nodes

shutdown arrête l'ordinateur
shutdown -r le reboot

ps -l-F : Localisation du processus
0 Hors de la mémoire centrale
1 En mémoire
2 Processus système
4 Verouillé (en attente d'E/S)
8 Vidage

ps -l-S : Etat du processus

#!/bin/bash pour indiquer le nom du shell utilisé

C'est un commentaire (pour afficher un commentaire)

echo permet d'afficher une variable
echo $maVariable # affiche ce que contient maVariable
echo 'Bonjour' # affiche Bonjour
Pour mélanger texte et variable, il faut utiliser echo " "

\' pour écrire une apostrophe \n pour retour à la ligne et utiliser echo -e au lieu de echo

read nomvariable demande à l'utilisateur d'entrer un parametre

ex :
```
1 #!/bin/bash
2
3 read nom prenom
4 echo "Bonjour $nom $prenom !"
```

read -p 'Salut ça va' nom-Variable permet de dire quelquechose avant la saisie.
read -p " -n 5 nom-variable limite à 5 le nombre de caractère qui peut être tapé
read -p " -t 5 nom-Variable limite à 5 secondes le temps de saisie
read -p " -s nom-Variable pour ne pas que l'on voit ce que l'on écrit

let "a=5" affecte à a la valeur 5
let "b=3"
let "c=a+b" let permet de manipuler les nombre contrairement à " "
espace

On peut utiliser : + - * / ** puissance % modulo

Pour lire un script il faut se donner les droits, pour cela :
$ chmod +x essai.sh

grep donne les lignes
grep -n donne les lignes et numero
grep -l donne fichiers

Variable disparait à la fin du script alors qu'une variable d'environnement peut être utilisée dans plusieurs programme.

env affiche les variables d'environnement

On peut utiliser ces variables avec echo comme une variable classique

---

Pour utiliser des paramètres : ./variables.sh param1 param2 ... paramn

Pour les réutiliser dans le script : $# contient le nombre de paramètres
$0 contient le nom du script exécuté (ici "./variables.sh"
$1 contient le 1er paramètre
$n contient le même paramètre (9 au max)

shift décale les paramètre : après shift, param 1 devient ce qui était param 2, param 2 param 3 etc

---

tableau=('valeur0' 'valeur1' 'valeur2') crée un tableau de 3 case (commence case 0)

tableau[2]='valeur2' pour changer une valeur ou créer le tableau à la main

${tableau[2]} pour afficher la valeur Un tableau peut avoir autant de cases que l'on veut et on peut en sauter

echo

echo ${tableau[*]} affiche tout le tableau d'un coup

Différents tests possibles sur chaînes

$chaine1 = $chaine
$chaine1 != $chaine2 Teste si les 2 chaînes sont différentes
-z $chaine Teste si la chaine est vide
-n Teste si la chaîne est non vide

Test sur nombre

$num1 -eq $num2 test si les nombres sont égaux
$num1 -ne $num2 test si les nombres sont différents
$num1 -lt $num2 test si num1 est inférieur (<) à num2
$num1 -le $num2 test si num1 est inférieur ou égal (<=) à num2
$num1 -gt $num2 test si num1 est sup (>) à num2
$num1 -ge $num2 test si num1 est sup ou égal (>=) à num2

Test sur fichier

-e $nomfichier Test si le fichier existe
-d $nomfichier Test si le fichier est un répertoire (Sous Linux fichier et répertoires sont vus
-f $nomfichier " fichier comme fichiers)
-L $nomfichier " lien symbolique
-r " lisible (r)
-w " modifiable (w)
-x " exécutable (x)
$fichier1 -nt $fichier2 Test si le fichier 1 est plus récent que le 2
$fichier1 -ot $fichier2 Test si le fichier 2 est plus vieux que le 2

[ ] && [ ] pour un et sur conditions
[ ] || [ ] pour un ou sur conditions

---

if [! -e fichier] pour inverser un test

---

for classique
for i in `seq 1 10`; et `seq 1 2 10`; fait un pas de 2.
do echo $i
done.

for fichier in 'ls'

cat head tail more
grep
cut
sort trucs des mots

Process Block Control (PCB)

. Etat courant du processus
. Change à chaque instant
. Contient : Copie du PSW (dernière interruption)
Etat du processus
Informations sur les ressources utilisées
Informations nécessaires pour la reprise des processus

Stocké en mémoire pour cause d'utilisation intensives

Interruptions

Signal généré par le matériel => Commutation de Contexte
Conséquence d'un événement interne, ou extérieur
Modification d'un indicateur permettant au SE de traiter le signal

Ressources : Tout ce qui est nécessaire à l'avancement d'un processus

. Processeur
. Mémoire
. Périphérique
. Bus
. Fichier
. ...

. Défaut de ressources => Mise en attente du processus (en général)

Cycle de ressources :

. Demande du processus au SE
. Demandes implicites (processeur)
. SE alloue une ressource à un processus
. Application reprise => Possibilité d'utiliser cette ressources jusqu'à libération par le processus ou SE la reprenne : préemption

Système de fichiers

. SE : masque les spécificités des disques (et E/S)
. modèle de programmation "unique", indépendant
. Appels systèmes : création, suppression, lecture écriture, ouverture, fermeture.
. Regroupé en arborescence
. Assure la protection des fichiers

Catégorie de SE

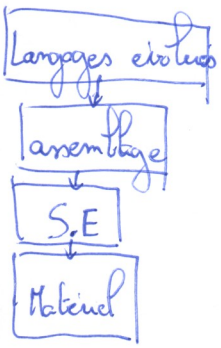
Systèmes monolithiques
Systèmes en couche
Micro-noyau
Systèmes mixtes (utilisation des deux premières méthode : Linux, Windows NT)

Eléments de base d'un SE

Système d'exploitation : programmes système qui permettent le fonctionnement de l'ordinateur

Deux grands types de SE :
- systèmes constructeurs (spécifiés pour un type de machine)
- systèmes ouverts (indépendants de la plate-forme matérielle)

Objectifs d'un SE :
- Présentation : abstraction plus simple et plus agréable que le matériel
  machine virtuelle
- Gestion : ordonne et contrôle l'allocation des processeurs
  mémoires, périphériques, des réseaux
  protège les utilisateurs dans le cas d'usage partagé



Fonctionnalités d'un SE :

Gestion de la mémoire
Exécution des E/S
Multiprogrammation, temps partagé, parallélisme : interruption, ordonnancement, répartition en mémoire, partage des données.
Lancement des outils du système (compilateurs, ...) et des outils pour l'administrateur du système
Lancement des travaux
Protection, sécurité : facturation des services
Réseaux.

Processus : Programme qui s'exécute
Données
Pile
Compteur ordinal
Pointeur de pile d'appel de fonction
Etat des registres

Process Status Word (PSW)
- Emplacement logique
- Conservé par le système
- Contient : Valeur du compteur ordinal
  Informations sur les interruptions (masquées ou non)
  Privilège du processeur (mode maître ou esclave)
  ... (format spécifique à un processeur)

Appels systèmes
Actions possibles :
fork : création d'un processus (fils) par un processus actif
kill : Destruction d'un processus
sleep, wait : Mise en attente, réveil d'un processus
scheduling : suspension et reprise d'un processus
malloc, free : demande de mémoire supplémentaire ou restitution de mémoire utilisée
wait : attendre la fin d'un processus fils
IPC, semaphore : Echanges de messages avec d'autres processus
Signaux : spécifications d'actions à entreprendre en fonction d'événements extérieurs asynchrones
nice : modifier la priorité d'un processus