RESERVE A L'ANONYMAT	**UNIVERSITÉ BORDEAUX 1** **Concours externe de technicien recherche et formation** **Technicien d'exploitation et de maintenance** **BAP E – SESSION 2012 – 11 juin 2012**
NE RIEN	Nom : ..... Nom marital (pour les femmes mariées) : ..... Prénom : ..... Date de naissance :
INSCRIRE DANS CES CASES	**UNIVERSITÉ BORDEAUX 1** Concours externe de technicien Recherche et Formation Technicien d'exploitation et de maintenance - BAP E – session 2012 **EPREUVE PROFESSIONNELLE D'ADMISSION** **Durée : 1 heure – Coefficient 4**

Les copies sont anonymes, elles ne doivent comporter aucun signe distinctif susceptible de permettre l'identification des candidats, en dehors du cadre prévu à cet effet. Tout signe distinctif entraîne l'annulation de la copie.

Le sujet que vous devez traiter comporte 14 pages numérotées de 1 à 14. Assurez-vous que cet exemplaire est complet. S'il est incomplet, demandez un autre exemplaire au surveillant.

Barème indicatif :

- Question 1 : 2 points
- Question 2 : 7 points
- Question 3 : 2 points
- Question 4 : 3 points
- Question 5 : 2 points
- Question 6 : 4 points

AUCUN DOCUMENT NI MATERIEL N'EST AUTORISE

1. Question 1

Dans le cadre d'une installation d'un système d'exploitation à distance, votre supérieur vous demande de démarrer un ordinateur de type PC sur le réseau filaire en PXE. Comment réalisez-vous cette opération ?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Question 2

Expliquez de façon détaillée les **étapes** de résolution d'un problème matériel, depuis la demande de l'utilisateur jusqu'au dépannage complet. La prise en charge de l'incident est réalisée par un technicien aux niveaux 0 et 1 en utilisant un logiciel de gestion d'incidents.

Une réponse plus méthodologique que technique est attendue.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

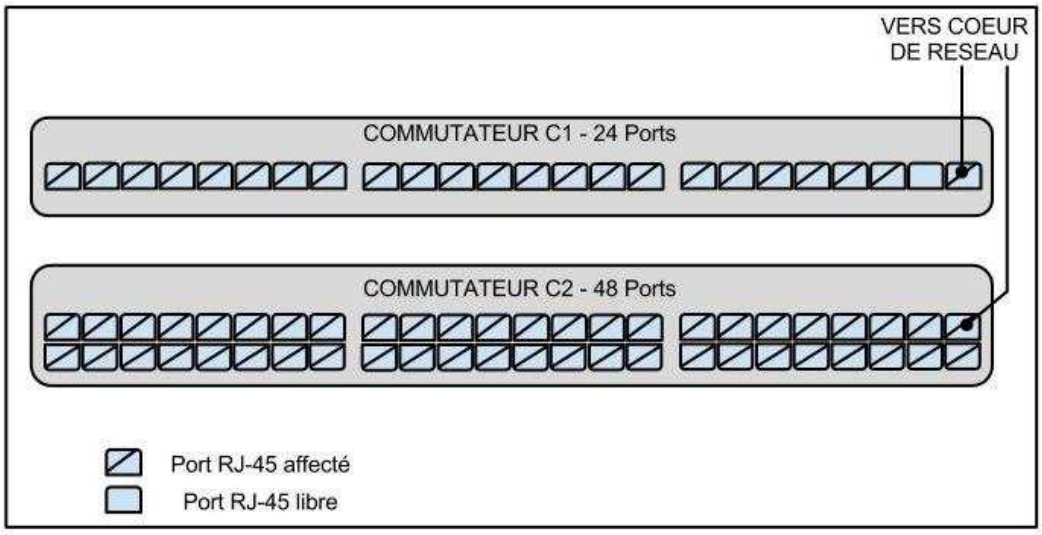
.....

.....

3. Question 3

La distribution réseau d'une grande salle de libre-service étudiant est assurée conformément au schéma ci-dessous.

Le commutateur C2 tombe en panne. Afin de rétablir le service au plus vite en attendant de le remplacer, vous utilisez les équipements disponibles en stock, à savoir des commutateurs 12 ports non stackables.



1. Combien vous en faut-il ? Justifier la réponse

.....

.....

.....

.....

.....

.....

.....

.....

2. Certains de ces commutateurs 12 ports sont POE. Pouvez-vous quand même les utiliser ?

.....

.....

.....

.....

.....

.....

3. Sur le commutateur C1, les 6 ports précédant le seul port libre sont utilisés pour desservir un service administratif (situé sur un autre réseau virtuel, bien sûr). Pouvez-vous malgré tout utiliser ce port libre pour les étudiants ?

.....

.....

.....

.....

.....

.....

.....

.....

4. Question 4

Pour les besoins d'un TP de mesures, un enseignant vous demande d'installer un logiciel qui contrôle une sonde lumineuse.

Lors de l'installation sur les postes pédagogiques (équipés du système d'exploitation Microsoft Windows XP SP3), en mode administrateur, vous ne rencontrez aucun problème ; mais ne connaissant pas le logiciel, vous vous contentez de vérifier qu'il se lance correctement.

Au cours d'un test de fonctionnement, l'enseignant (avec son compte dépourvu de privilèges) vous signale un problème : la sonde est bien détectée par le logiciel mais aucun résultat n'est acquis. Dans la documentation, il est précisé que le logiciel utilise un fichier de résultats temporaires pour l'acquisition des données. Vous décidez d'analyser le fonctionnement de l'application (LightSensor.exe) à l'aide du logiciel « Process Monitor » dont vous trouverez une capture d'écran en annexe 1.

1. A quoi sert le logiciel « Process Monitor » ?

.....

.....

.....

.....

.....

.....

2. Que vous indique-t-il sur le problème rencontré ?

.....

.....

.....

.....

.....

.....

3. Comment résoudre le problème ?

.....

.....

.....

.....

.....

.....

5. Question 5

Le script suivant (Visual Basic Script) est déployé par GPO sur les ordinateurs équipés du système Microsoft Windows d'une salle d'enseignement pour les besoins d'un TP en CAO (Conception Assistée par Ordinateur).

```
Option explicit

Dim source_dir, xmlfiles, target_dir, file, objFS, objFile

source_dir = "\\srv\scripts\"
xmlfiles = Array("CompositesGVSContourOnly.xml", "CompositesGVSExplodedOnly.xml",
"CompositesGVSFlattenOnly.xml")
target_dir = "C:\Program Files\CAO\resources\templates\"

set objFS = CreateObject("Scripting.FileSystemObject")

For Each file In xmlfiles
	if Not objFS.FileExists(target_dir & file) then
		if objFS.FileExists(source_dir & file) then
			Set objFile = objFS.GetFile(source_dir & file)
			objFile.Copy(target_dir)
		end if
	end if
Next
```

1. Que fait ce script ?

.....

.....

.....

.....

.....

.....

2. Quel est le rôle de la GPO ?

.....

.....

.....

.....

.....

.....

6. Question 6

La direction des études vous demande de mettre à jour une liste de diffusion étudiante basée sur le moteur de liste SYMPA. Pour se faire, elle vous communique un fichier Microsoft Excel contenant les colonnes suivantes :

- N° d'étudiant
- Nom
- Prénom
- Date de naissance
- Adresse mail

La liste de diffusion est : **etu-promo1**, elle ne contient aucune adresse email au moment de la demande.

Le moteur de listes est accessible à l'adresse mail suivante : sympa@listes.univ.fr

Le guide de l'utilisateur du moteur de listes SYMPA est disponible en annexe 2.

Question : A l'aide des commandes Unix **mawk** (voir page de manuel en annexe 3) et **mail**, expliquez comment vous répondez à la demande.

.....

.....

.....

.....

.....

.....

.....

.....

.....

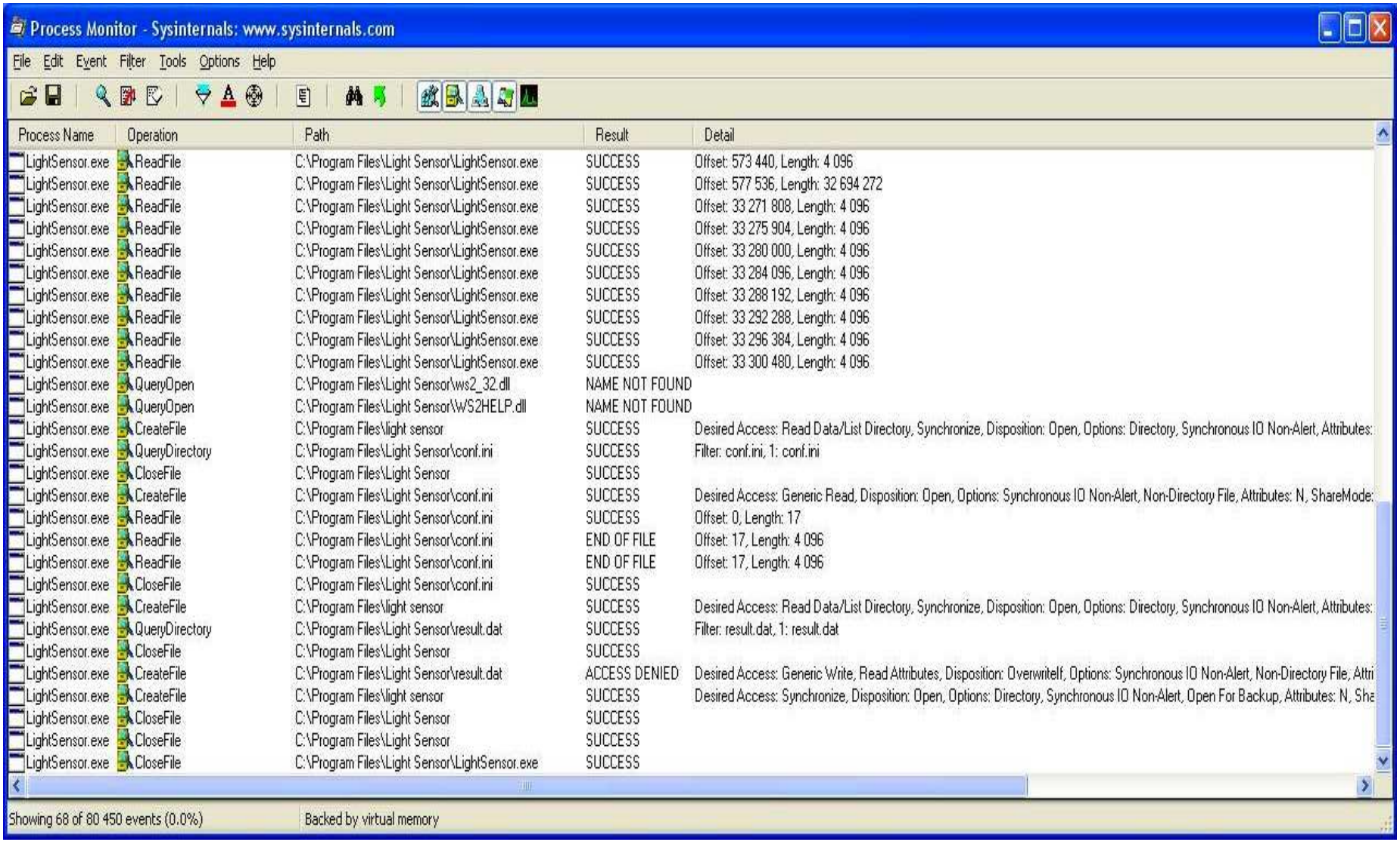
.....

.....

.....

.....

Annexe 1 : Process Monitor



Annexe 2 : SYMPA

SYMPA -- Système de Multi-Postage Automatique

Guide de l'utilisateur

SYMPA est un gestionnaire de listes électroniques. Il permet d'automatiser les fonctions de gestion des listes telles que les abonnements, la modération et la gestion des archives.

Toutes les commandes doivent être adressées à l'adresse électronique sympa@listes.univ.fr.

Il est possible de mettre plusieurs commandes dans chaque message : les commandes doivent apparaître dans le corps du message et chaque ligne ne doit contenir qu'une seule commande. Sympa ignore le corps du message si celui-ci n'est pas de type "Content-type: text/plain", mais même si vous êtes fanatique d'un agent de messagerie qui fabrique systématiquement des messages "multipart" ou "text/html", les commandes placées dans le sujet du messages sont reconnues.

Les commandes disponibles sont :

```
 HELp                          * Recevoir ce fichier d'aide
 LISts                         * Recevoir l'annuaire des listes gérées sur ce
                                 noeud
 REView <list>                 * Recevoir la liste des abonnés à <list>
 WHICH                         * Recevoir la liste des listes auxquelles
                                 on est abonné
 SUBscribe <list> Prénom Nom * S'abonner ou confirmer son abonnement à <list>
 SIGnoff <list|*> user@univ.fr    * Quitter <list>, ou toutes les listes
                                 (user@univ.fr est facultatif)

 SET <list|*> NOMAIL           * Suspendre la réception des messages de <list>
 SET <list|*> MAIL             * Recevoir les messages en mode normal
 SET <list|*> DIGEST           * Recevoir les messages en mode compilation
 SET <list|*> SUMMARY          * Recevoir la liste des messages uniquement
 SET <list|*> NOTICE           * Recevoir l'objet des message uniquement

 SET <list|*> CONCEAL          * Passage en liste rouge (adresse d'abonné cachée)
 SET <list|*> NOCONCEAL        * Adresse d'abonné visible via REView

 INFO <list>                   * Recevoir les informations sur <list>
 INDex <list>                  * Recevoir la liste des fichiers de l'archive
                                 de <list>
 GET <list> <fichier>          * Recevoir <fichier> de l'archive de <list>
 LAST <list>                   * Recevoir le dernier message de <list>
 INVITE <list> <e-mail>        * Inviter <e-mail> à s'abonner à <list>
 CONFIRM <clef>                * Confirmer l'envoi d'un message
                                 (selon la configuration de la liste)
 QUIT                          * Indiquer la fin des commandes
                                 (pour ignorer une signature)
```

Commandes réservées aux propriétaires de listes :

```
 ADD <list> user@host Prenom Nom * Ajouter un utilisateur à <list>
 DEL <list> user@host            * Supprimer un utilisateur de <list>
 STATS <list>                    * Consulter les statistiques de <list>
 EXPire <list> <ancien> <delai>  * Déclancher un processus d'expiration pour
                                   les abonnés à <list> n'ayant pas confirmé
                                   leur abonnement depuis <ancien> jours.
                                   Les abonnés ont <delai> jours pour
```

```
                                  confirmer
 EXPireINDex <list>             * Connaître l'état du processus d'expiration
                                  en cours pour la liste <list>
 EXPireDEL <list>               * Désactiver le processus d'expiration de
                                  <list>

 REMind <list>                  * Envoyer à chaque abonné un message
                                  personnalisé lui rappelant l'adresse
                                  avec laquelle il est abonné
```

Commandes réservées aux modérateurs de listes :

```
 DISTribute <list> <clef>       * Modération : valider un message
 REJect <list> <clef>           * Modération : invalider un message
 MODINDEX <list>                * Modération : consulter la liste des messages
                                                à modérer
```

Powered by Sympa 4.1.2 : http://listes.cru.fr/sympa/

Annexe 3 : mawk (extraits de la page de manuel)

```
MAWK(1)
USER COMMANDS
MAWK(1)

NAME
       mawk - pattern scanning and text processing language

SYNOPSIS
       mawk [-W option] [-F value] [-v var=value] [--] 'program text' [file ...]
       mawk [-W option] [-F value] [-v var=value] [-f program-file] [--] [file
...]

DESCRIPTION
       mawk is an interpreter for the AWK Programming Language.  The AWK
language is useful for manipulation of data files, text retrieval and
processing, and for prototyping and experimenting with algorithms.  mawk is a
new awk meaning it implements the AWK lan
       guage as defined in Aho, Kernighan and Weinberger, The AWK Programming
Language, Addison-Wesley Publishing, 1988.  (Hereafter referred to as the AWK
book.)  mawk conforms to the Posix 1003.2 (draft 11.3) definition of the AWK
language which contains a  few
       features not described in the AWK book,  and mawk provides a small number
of extensions.

       An  AWK  program  is a sequence of pattern {action} pairs and function
definitions.  Short programs are entered on the command line usually enclosed in
' ' to avoid shell interpretation.  Longer programs can be read in from a file
with the -f option.  Data
       input is read from the list of files on the command line or from standard
input when the list is empty.  The input is broken into records as determined by
the record separator variable, RS.  Initially, RS = "\n" and records are
synonymous with lines.  Each
       record is compared against each pattern and if it matches, the program
text for {action} is executed.

OPTIONS
       -F value       sets the field separator, FS, to value.

       -f file        Program text is read from file instead of from the command
line.  Multiple -f options are allowed.

       -v var=value   assigns value to program variable var.

       --             indicates the unambiguous end of options.

       The above options will be available with any Posix compatible
implementation of AWK, and implementation specific options are prefaced with -W.
mawk provides six:

       -W version     mawk writes its version and copyright to stdout and
compiled limits to stderr and exits 0.

       -W dump        writes an assembler like listing of the internal
representation of the program to stdout and exits 0 (on successful compilation).

       -W interactive sets unbuffered writes to stdout and line buffered reads
from stdin.  Records from stdin are lines regardless of the value of RS.

       -W exec file   Program text is read from file and this is the last
option. Useful on systems that support the #!  "magic number" convention for
executable scripts.
```

-W sprintf=num adjusts the size of mawk's internal sprintf buffer to num bytes. More than rare use of this option indicates mawk should be recompiled.

-W posix_space forces mawk not to consider '\n' to be space.

The short forms -W[vdiesp] are recognized and on some systems -We is mandatory to avoid command line length limitations.

THE AWK LANGUAGE

1. Program structure

An AWK program is a sequence of pattern {action} pairs and user function definitions.

A pattern can be:

```
BEGIN
END
expression
expression , expression
```

One, but not both, of pattern {action} can be omitted. If {action} is omitted it is implicitly { print }. If pattern is omitted, then it is implicitly matched. BEGIN and END patterns require an action.

Statements are terminated by newlines, semi-colons or both. Groups of statements such as actions or loop bodies are blocked via { ... } as in C. The last statement in a block doesn't need a terminator. Blank lines have no meaning; an empty statement is
terminated with a semi-colon. Long statements can be continued with a backslash, \. A statement can be broken without a backslash after a comma, left brace, &&, ||, do, else, the right parenthesis of an if, while or for statement, and the right parenthe
sis of a function definition. A comment starts with # and extends to, but does not include the end of line.

The following statements control program flow inside blocks.

```
if ( expr ) statement

if ( expr ) statement else statement

while ( expr ) statement

do statement while ( expr )

for ( opt_expr ; opt_expr ; opt_expr ) statement

for ( var in array ) statement

continue

break
```

2. Records and fields

Records are read in one at a time, and stored in the field variable $0. The record is split into fields which are stored in $1, $2, ..., $NF. The built-in variable NF is set to the number of fields, and NR and FNR are incremented by 1. Fields above $NF
are set to "".

Assignment to $0 causes the fields and NF to be recomputed. Assignment to NF or to a field causes $0 to be reconstructed by concatenating the $i's separated by OFS. Assignment to a field with index greater than NF, increases NF and causes $0 to be recon

structed.

Data input stored in fields is string, unless the entire field has numeric form and then the type is number and string. For example,

```
echo 24 24E |
mawk '{ print($1>100, $1>"100", $2>100, $2>"100") }'
0 1 1 1
```

$0 and $2 are string and $1 is number and string. The first comparison is numeric, the second is string, the third is string (100 is converted to "100"), and the last is string.

3. Input and output

There are two output statements, print and printf.

print writes $0 ORS to standard output.

print expr1, expr2, ..., exprn
writes expr1 OFS expr2 OFS ... exprn ORS to standard output. Numeric expressions are converted to string with OFMT.

printf format, expr-list
duplicates the printf C library function writing to standard output. The complete ANSI C format specifications are recognized with conversions %c, %d, %e, %E, %f, %g, %G, %i, %o, %s, %u, %x, %X and %%, and conversion qualifiers h and l.

The argument list to print or printf can optionally be enclosed in parentheses. Print formats numbers using OFMT or "%d" for exact integers. "%c" with a numeric argument prints the corresponding 8 bit character, with a string argument it prints the first character of the string. The output of print and printf can be redirected to a file or command by appending > file, >> file or | command to the end of the print statement. Redirection opens file or command only once, subsequent redirections append to the already open stream. By convention, mawk associates the filename "/dev/stderr" with stderr which allows print and printf to be redirected to stderr. mawk also associates "-" and "/dev/stdout" with stdin and stdout which allows these streams to be passed to functions.

AUTHOR

Mike Brennan (brennan@whidbey.com).