SAE 1.03 : Installation d’un poste pour le développement

Compétence évaluée : Administrer des systèmes communicants complexes - Niveau 1

Apprentissages critiques :

- Identifier les différents composants (matériels et logiciels) d’un système numérique
- Utiliser les fonctionnalités de base d’un système multitâches / multi-utilisateurs
- Installer et configurer un système d’exploitation et des outils de développement

Manuel d'installation

Méthode étape par étape :

1. Installer VirtualBox 7.0 selon votre système d'exploitation
2. Lancer VirtualBox
3. Installer Debian 12 selon les propriétés de votre ordinateur
4. Retourner sur VirtualBox et appuyer sur le bouton Nouvelle
    a. Donner un nom à votre Machine virtuelle (VM)
    b. S'assurer que le type et la version correspondent bien aux logiciels téléchargés
    c. Implémenter ¼ de la RAM de votre pc dans la machine et pas plus car sinon votre machine va trop utiliser les composants de votre pc
    d. Indiquez le disque dur que vous voulez utiliser
    e. Ajouter au minimum 16 Go au disque dur virtuel
5. Ensuite cliquer sur configuration
    a. Système -> processeur -> Nombre de processeur = Un cran dans la zone verte juste avant la jonction entre zone verte et rouge
6. Démarrer la VM
7. Sélection du disque
    a. Cliquez sur le dossier juste à côté du menu déroulant
    b. Cliquez sur le bouton ajouter
    c. Sélectionner le fichier debian en .iso et confirmer
8. La VM démarre et vous devez ensuite sélectionner avec "Entrée" la ligne "Graphical Install"
    a. Sélectionner la langue que vous-voulez
    b. Sélectionner la zone géographique
    c. Sélectionner la configuration clavier (France = AZERTY)
9. Configurer le réseau
    a. Indiquer le nom de votre machine virtuelle
    b. Si nécessaire, indiquer le nom de domaine (inutile si pour votre machine)
10. Créer le mot de passe du superutilisateur 'root' (ATTENTION À BIEN LIRE LES CONSIGNES ÉCRITES SUR L'ÉCRAN)
11. Créer ensuite un compte utilisateur autre que le compte 'root'
    a. Entrer son nom d'utilisateur complet
    b. Entrer son identifiant (cela peut-être le nom complet)
    c. Créer son mot de passe
12. Partitionner les disques
    a. Sélectionner la configuration Manuel
    b. Sélectionner la ligne SCSI3
    c. Répondez ensuite oui à la création d'une nouvelle table

d. Sélectionner ensuite la table que vous avez créer avec le nombre d'espace qui lui est alloué
    i. Cliquez sur la ligne "Créer une nouvelle partition"
        1. Allouer lui l'espace disponible moins 2GB
        2. Sélectionner le bouton primaire
        3. Sélectionner le bouton début
        4. Vérifiez que la partition est en "fichier journalisé ext 4" et que le point de montage est / (la racine)
        5. Fin du paramétrage de la nouvelle partition
    ii. Reprendre la partie de la partition libre (normalement les 2GB restant)
        1. Créer la partition
        2. Lui allouer les 2 GB restant
        3. Sélectionner la ligne Logique
        4. Sélectionner ensuite la ligne "Utiliser comme" et changer le format pour qu'il soit en "espace d'échange ("swap")"
        5. Fin du paramétrage de la nouvelle partition
    iii. Terminer le partitionnement
    iv. Appliquer le changements sur les disques
e. Configurer l'outil de gestion des paquets
    i. Refuser l'analyse d'un CD ou DVD
    ii. Sélectionner le pays le plus proche géographiquement pour l'attribution d'un miroir
    iii. Sélectionner un serveur debian en fonction de vos préférences
    iv. Ajouter un proxy (facultatif)
f. Répondre à la question d'envoi de données anonymement (Pas d'impact sur la VM)
g. Sélectionner l'environnement de bureau en fonction des disponibilités de votre machine et les projets que vous voulez faire sur la VM (ici Xfce)
h. Accepter l'installation du système GRUB
i. Sélectionner le disque d'installation pour GRUB
j. Confirmer l'installation
k. (Lancement de la VM)
l. Sélectionner Debian sur la machine
m. Se connecter avec le compte utilisateur
n. Lancez le terminal
    i. Taper la commande 'su root'
    ii. Entrer votre mot de passe
    iii. Puis taper la commande 'gpasswd -a <nom d'utilisateur> sudo'
o. Déconnectez-vous

p. Reconnectez-vous avec le compte utilisateur
q. Ouvrez un terminal et entrez la commande ‘sudo apt install make gcc dkms linux-source linux-headers-$(uname -r)
    i. Entrez votre mot de passe
    ii. Confirmer avec enter ou en tapant “o”
r. Quitter le terminal et ouvrir dans ‘Périphérique’ - > ‘Insérer l’image CD des Additions invité’
s. Ouvrir un terminal dans le dossier ouvert
t. Entrez la commande ‘sudo sh VBoxLinuxAdditions.run’
    i. Entrer votre mot de passe
u. Fermer tous les onglets
v. Éteindre la VM et la relancer
w. Aller dans ‘Périphérique’ -> ‘Presse-papier’ et cliquez sur Bidirectionnel

Nous avons fait le choix de mettre de 8 Go de RAM car nous pensions qu’avec moins la VM aller ralentir et avec plus le PC n’allait pas supporter et lui se mettre à ralentir. Concernant les 4 processeurs alloués à la machine, le PC en comportant 16, nous pensons qu'allouer ¼ des processeurs à la machine lui permettrait de n’avoir aucun problèmes de ralentissement ou de blocages de tâches pour les fonctions qui lui sont demandés tout en permettant au Pc de garder une grande quantité des ses processeurs. Pour la quantité de stockage, nous avons simplement suivis les recommandations qui nous avaient été indiquées en allouant le maximum d’espace possible - 2Go pour le disque de stockage et les deux Gigas restants pour le disque d’échanges. Enfin pour l'environnement de travail, après consultation de plusieurs sites et avis sur des forums, il semblait ressortir que pour un équilibre fonctionnalités, ergonomie, et allocation de ressource, Xfce était l'environnement de travail le plus adapté à notre projet.

Nous avons fait le choix final pour les droits des groupes pour des droits au niveau 750, c'est-à-dire RWX.R-X.--- . Soit tous les droits pour le propriétaire, le droit d’écriture et de lecture pour les membres de son groupe et aucun droit pour les autres.

Script rappel.bash :

```
#!/bin/bash
#script vérifiant les emprunts datant de plus d'un moins et les affichant
# Parcourir les emprunts
while IFS= read -r ligne; do
    # Récupération de la date d'emprunt
    dateEmprunt=$(echo "$ligne" | cut -d ";" -f 4)
    jouremprunt=$(echo $dateEmprunt | cut -d ' ' -f 1)
```

```
    moisemprunt=$(echo $dateEmprunt | cut -d ' ' -f 2)
    anneemprunt=$(echo $dateEmprunt | cut -d ' ' -f 3)

    dateCourante=$(date '+%-d %-m %Y %X')
    #Transformation de la date courante dans ce format
jour_mois_annee_heure:minute:seconde
    dateCourante=$(echo $dateCourante | sed 's/ /_/g')

    # Conversion de la date d'emprunt en format timestamp
    dateEmpruntTimestamp=$(date -d
"$anneemprunt-$moisemprunt-$jouremprunt" +%s 2>/dev/null)

    #Utilisation de comptemps.bash pour comparer les dates
    bash comptemps.bash $jouremprunt $moisemprunt $anneemprunt
$dateCourante > /dev/null

    # Récupération du code de retour
    codeRetour=$?

    # Si la date d'emprunt est supérieure à un mois donc code retour = 1
ou 2
    if [ $codeRetour -eq 1 ] || [ $codeRetour -eq 2 ]; then
        echo "$ligne"
    fi

done < "emprunts.txt"
```

Cette partie de code est utile et intéressante car elle correspond à la partie du programme qui sert à renvoyer les messages de rappel qui sont, dans ce système, la clé de l'interaction avec les usagers. En effet, ce bout de programme va servir à récupérer la date à laquelle a été emprunté chaque livre se trouvant dans le fichier emprunt à l'aide d'une boussole for qui va lire chaque ligne du fichier. Une fois cela fait le programme convertit la date et fait appel à comptemps pour comparer les différentes dates. Le programme récupère le code retour de comptemps et l'utilise pour afficher les lignes qui doivent être affichés.

Script rend.bash :

```
#!/bin/bash
```

```
#Script qui prend en compte 2 arguments et qui vérifie leur validité
#Puis contrôle si l'emprunt est bien enregistré dans le fichier
#Si l'emprunt est enregistré, le script supprime la ligne dans le fichier
#Et modifie le fichier de stock en ajoutant la ligne de l'exemplaire

#Récupération des numéros d'adhérent et de livre et d'exemplaire
idAdh=$(grep "$1;" membres.txt | cut -d ";" -f 1)
idLivre=$(grep "$2;" livres.txt | cut -d ";" -f 1)

#Récupération de l'exemplaire
#Il doit vérifier si l'exemplaire est disponible et prendre le premier
qu'il est
idexemplaire=$(grep "$idLivre;[^;];[^;]" exemplaires.txt | grep "non" |
head -1 | cut -d ";" -f 2)

#Suppression de la ligne dans le fichier emprunts.txt
sed -i "/$idAdh;$idexemplaire/d" emprunts.txt

#Ajout de la ligne dans le fichier stock.txt
sed -i "s/$idLivre;$idexemplaire;non/$idLivre;$idexemplaire;oui/"
exemplaires.txt
echo "Emprunt rendu avec succès"
exit 0
```

Ce bout de programme de rend sert à la mise à jour des fichiers de stockages, ici le programme récupère les informations nécessaires au bon format du fichier et supprime la première ligne correspondante dans emprunts et l'ajoute dans stock.

Script inscrire.bash :

```
#!/bin/bash

# Inscrire un adhérent
# $1: option -a
# $2: nom
# $3: prénom
# $4: adresse

# Ajouter un livre à la bibliothèque
# $1: option -l
# $2: nombre exemplaires
# $3: titre
```

```
# $4: Auteur

#Vérif si l'option est -a ou -l
if [ $1 = "-a" ]
then
    echo "Inscrire un adhérent"
    echo "Nom: $2"
    echo "Prénom: $3"
    echo "Adresse: $4"
elif [ $1 = "-l" ]
then
    echo "Ajouter un livre"
    echo "Nombre d'exemplaires: $2"
    echo "Titre: $3"
    echo "Auteur: $4"
else
    echo "Option inconnue"
    exit 1
fi

#########################################################################
#######
#Vérif si un id est existant sinon commencer à 1 sinon incrémenter de 1
(MEMBRES)
if [ ! -e membres.txt ]
then
    ID=1
else
    ID=$(tail -1 membres.txt | cut -d ";" -f 1)
    ID=$(($ID + 1))
fi

#Vérif si un id est existant sinon commencer à 1 sinon incrémenter de 1
(Livres)
if [ ! -e livres.txt ]
then
    IDli=1
else
    IDli=$(tail -1 livres.txt | cut -d ";" -f 1)
    IDli=$(($IDli + 1))
```

```
fi
#########################################################################
#######

#########################################
#Inscription dans le fichier texte membres.txt
if [ $1 = "-a" ]
then
    echo -e "$ID;$2;$3;$4" >> membres.txt
elif [ $1 = "-l" ]
then
    echo -e "$IDli;$3;$4" >> livres.txt
fi

#Inscription des exemplaires dans le fichier texte exemplaires.txt
if [ $1 = "-l" ]
then
    for i in $(seq 1 $2)
    do
        echo -e "$IDli;$i;oui" >> exemplaires.txt
    done
fi
#########################################
```

Cet extrait de programme montre les différentes options que le programme inscrire prends en compte et comment il exécute différentes suites d'instructions en fonction du choix de l'utilisateur

Script demande.bash :

```
#!/bin/bash
#Script qui a 2 arguments : Nom adhérent et titre du livre
#Vérif si le nombre d'arguments est bon
# $1 : nom adhérent
# $2 : titre du livre

#Récupération des numéros d'adhérent et de livre et d'exemplaire
idAdh=$(grep "$1;" membres.txt | cut -d ";" -f 1)
idLivre=$(grep "$2;" livres.txt | cut -d ";" -f 1)

#Récupération de l'exemplaire
```

```
#Il doit vérifier si l'exemplaire est disponible et prendre le premier
qu'il est
idexemplaire=$(grep "$idLivre;[^;];[^;]" exemplaires.txt | grep "oui" |
head -1 | cut -d ";" -f 2)

#Vérif si l'emprunt est déjà enregistré
if [ $(grep -c "$idAdh;$idLivre;$idexemplaire" emprunts.txt) -ne 0 ]
then
    echo "Il n'y a plus d'exemplaire disponible"
    exit 1
else
    echo "L'emprunt n'est pas enregistré"
fi

#Enregistrement de l'emprunt
echo -e "$idAdh;$idLivre;$idexemplaire;$date" >> emprunts.txt

#Modification de la disponibilité du livre
sed -i "s/$idLivre;$idexemplaire;oui/$idLivre;$idexemplaire;non/"
exemplaires.txt
```

Cet extrait de demande reprend le même principe que rend mais dans le sens opposé, il récupère donc les informations sur l'adhérent et le livre et vérifie si le livre est disponible ou déjà emprunté. Selon le cas, soit elle sort du programme, soit elle enregistre l'emprunt et modifie le fichier de disponibilité des livres, exemplaires

Script comptemps.bash :

```
#!/bin/bash
#Script comparant la date courante à une date donnée en argument

#$1 Jour dans le mois
#$2 Mois
#$3 Année
#$4 Date courante (format : jour mois année heure:minute:seconde)

# Récupération des arguments
jour=$1
mois=$2
annee=$3
dateCourante=$4
```

```
# Récupération de la date courante
jourCourant=$(echo $dateCourante | cut -d '_' -f 1)
moisCourant=$(echo $dateCourante | cut -d '_' -f 2)
anneeCourant=$(echo $dateCourante | cut -d '_' -f 3)

# Conversion des dates en format timestamp pour comparaison
dateArgTimestamp=$(date -d "$annee-$mois-$jour" +%s 2>/dev/null)
dateCouranteTimestamp=$(date -d "$anneeCourant-$moisCourant-$jourCourant"
+%s 2>/dev/null)

# Vérification de la validité des dates
if [ -z "$dateArgTimestamp" ] || [ -z "$dateCouranteTimestamp" ]; then
    echo "La date fournie n'est pas valide"
    exit 1
fi
```

Cet extrait de comptemps est le principe même du programme, c'est-à-dire la comparaison de date et la conversion de format de dates. Le programme récupère donc la date courante et convertit celle qu'il reçoit, puis les compare et renvoie le code de sortie en fonction de la raison.