Travail Pratique 1 (LINUX)
Travail pouvant être effectué en équipe de 1, 2, 3 ou 4 personnes
Vitual Machine System (VMS) et ressources partagées

En vous **inspirant** des utilitaires de gestion d'un VMS simple et d'une liste chaînée qui vous sont fournis dans le dossier **/exempleVMS** dans le répertoire des exemples sur DMILinux (voir le document cour1sif1015-A22-Update-Environnement.ppt accessible sur le lien : http://dmiftp.uqtr.ca/Public/FMeunier/sif1015/Documentations/Cours1-A22/) ainsi que des notions de gestion du partage des ressources présentées en classe, vous devez implanter un programme permettant la gestion de machines virtuelles (VMS) orientées Conteneur (voir fig. 1) pour l'éxécution d'exécutables (fichiers .olc3). Ce VMS devra pouvoir être géré de façon concurrente. Donc, plusieurs transactions pourraient être effectuées en parallèles. Ce programme devrait comporter les spécifications suivantes:

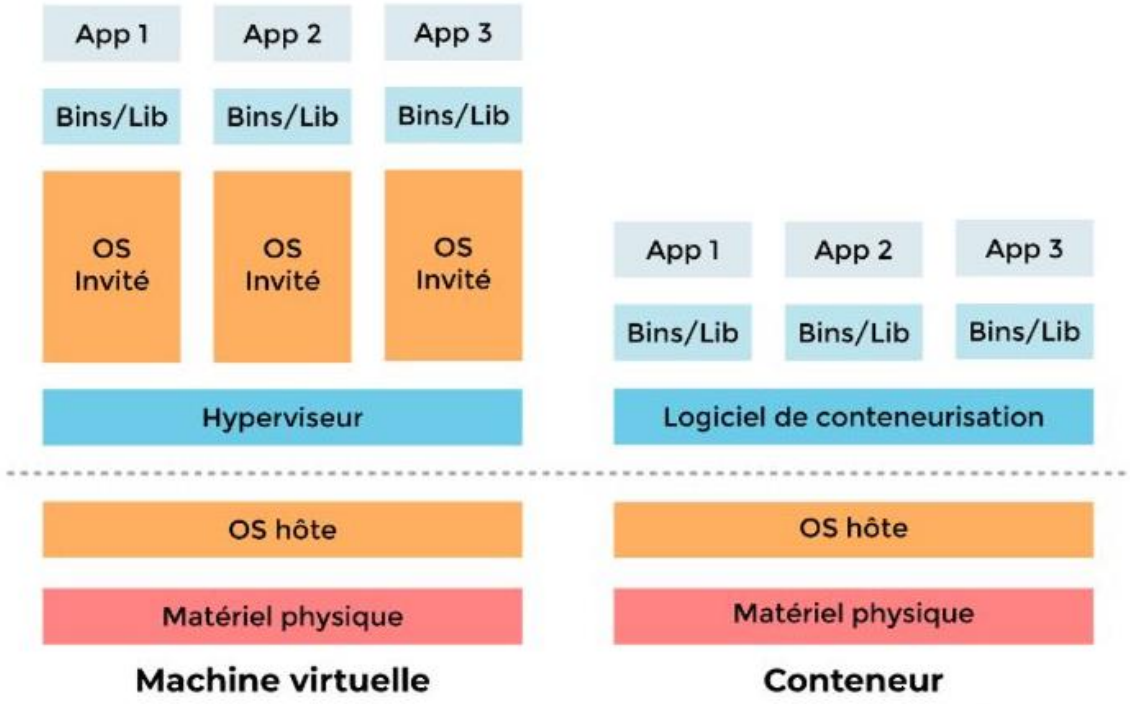

Figure 1. Schématisation de la comparaison entre les notions de Machine Virtuelle (ex : VMWARE) et Conteneur (ex : Docker).

a) Au départ, votre programme (main()) fait la lecture d'un fichier de transactions, dont le nom de fichier est passé en ligne de commande par l'usager. Le programme (main()) fait l'ouverture du fichier de transactions et fait la lecture de chaque transaction. Un **processus/thread est ensuite démarré pour traiter chaque transaction**. Chaque transaction contenue dans un fichier de transactions est sur une ligne individuelle du fichier de transactions (ex : trans2.txt). Par exemple :

A
L 1-4
A

```
L 1-4
A
L 1-4
E 2
L 1-4
A
L 1-4
X 1 Mult-Numbers.olc3
X 2 Mult-Numbers.olc3
```

b) Chaque ligne d'un fichier de transactions commence par une lettre qui représente le type de transaction à effectuer. **A** pour l'ajout d'une VM, **E** pour éliminer une VM dont le numéro est donné comme argument, **L** pour l'affichage à la console des informations de VM (ex : affichage des informations sur les VM de 1 à 4). **X** pour l'exécution (interptétation) du code binaire contenue dans le fichier .olc3 dont le nom est spécifié comme argument.

c) Votre programme (main()) crée une liste chaînée devant contenir chaque noeudVM. Chaque nœud de cette liste correspond à une structure de la forme :

```
struct noeudVM{
        struct infoVM       VM;
        struct noeudVM      *suivant;};
```

La structure infoVM correspond:

```
struct infoVM{
        int             noVM;
        unsigned char   busy;
        unsigned short *    ptrDebutVM;
        unsigned short *    offsetDebutCode; // region memoire ReadOnly
        unsigned short *    offsetFinCode;};
```

d) Étant donné que la liste chaînée de VM est une structure pouvant être partagée par plusieurs processus/thread pouvant ainsi causer des problèmes d'intégrité au niveau des informations (attributs d'une VM) contenues dans les nœuds de la liste chaînée. Il faut donc éviter que l'accès à chaque nœud de cette structure se fasse de façon simultanée par plusieurs threads, **l'accès à chaque nœud de la liste chaînée doit être restreint à un seul thread à la fois**. De plus, **l'accès au pointeur de tête et de queue de la liste chaînée, du nombre de VM actives, devra aussi être en exclusion mutuelle**. **L'accès à la console** pour l'affichage des informations (transactions de type L) par plusieurs threads doit aussi être fait en exclusion mutuelle. L'accès au système de fichier pourrait aussi avoir à être géré en exclusion mutuelle.

Contraintes de la partie 1 (10 points):

Le rapport de ce travail doit être remis le **18 octobre 2022.** La correction du travail est basée sur les critères suivants :

Fonctionnement des programmes 50 %

Programmes (code du main(), codes des procédures, fichier en-tête, makefile)

Style de programmation 50 %

(documentation, efficacité de la solution, validation, originalité de la solution)

Documents à remettre :

Vous devez déposer un manuel d'utilisateur de votre application sur le portail de cours, incluant votre *Username, Password* sur le serveur *DMILinux* ainsi que le chemin pour accéder à votre dossier TP1, le nom du fichier exécutable et le nom du fichier de transactions utilisé pour tester votre application.

Avis aux étudiants qui travaillent en VIRTUAL BOX, voir à me transmettre un rapport contenant les résultats de l'éxécution du fichier de transactions trans2.txt.