

Republique du Cameroun  
Paix-travail-patrie

Ministere de l'Enseignement Superieur

Universite de Yaounde 1

Ecole Nationale Supérieure  
Polytechnique

Republic of Cameroon  
Peace-work-fatherland

Ministry of Higher Education

University of Yaounde 1

National Advance School of  
Engineering

## DOCUMENTATION DU PROJET DE VIRTUALISATION

### **THEME: Migration des machines virtuelles dans un environnement heterogene**

#### **Redige par :**

Fomekong Tchoffo Frank Borel  
Azafack Temgoua Rosane  
Akoa Teddy William  
Nyongo Ghislain  
Geena Ayuk Nalowa  
Dongmo Vincess  
Kouogan Mike

#### **Encadreurs :**

Pr Alain Tchana  
Ing Fonyuy Caleb

**Annee Academique: 2022-2023**

# Table of Contents

I. Resume.....	4
1. Probleme.....	4
2. Solution.....	4
II. Prerequis.....	6
1.Liste des prerequis.....	6
2.Notations utilisees.....	6
III. Guide d'installation.....	7
IV. Configuration de la machine virtuelle.....	8
A. Conrafiguration de l'accès au reseau.....	8
Connexion du domU à internet.....	10
B. Configuration du service pour les tests suites.....	12
C. Configuration sur le dom0.....	14
V. Guide d'utilisation.....	15
ANNEXES.....	16
Installer et configure xen , xen-tools.....	16

## Table of Figures

Figure 1: probleme.....	4
Figure 2: Solution.....	5
Figure 3: repository.....	7
Figure 4: reseau.....	8
Figure 5: ip domU.....	9
Figure 6: internat interface dom0.....	11
Figure 7: myvm.cfg.....	14
Figure 8: myvm.cfg.....	14
Figure 9: app result.....	15
Figure 10: features list.....	15
Figure 11: contenu de myvm.cfg.....	17
Figure 12: contenu 2 de myvm.cfg.....	18

# I. Resume

## 1. Probleme

Echec de la migration d'une machine virtuelle dans un environnement hétérogène lorsque l'ensemble des features du processeur visible par la machine virtuelle sur la machine hôte de départ est différent de l'ensemble des features présentées à la machine virtuelle par la machine hôte de destination.

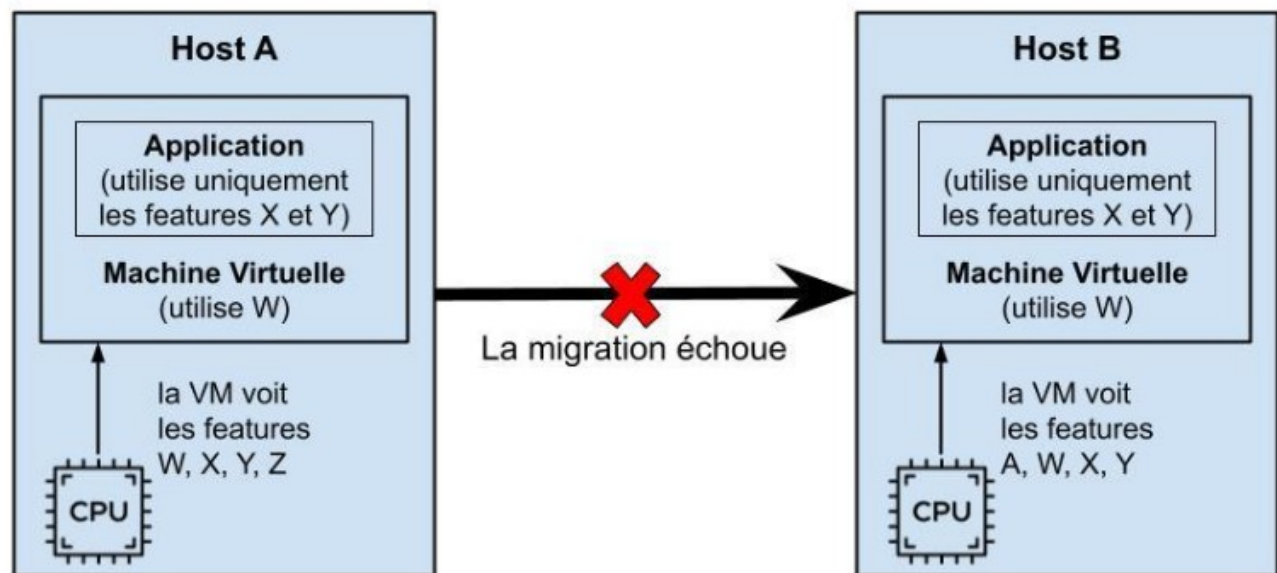


Figure 1: probleme

## 2. Solution

Créer un logiciel qui s'exécutera sur la machine hôte de départ qui prend en entrée une application et retourne l'ensemble des features du processeur utilisées lors du fonctionnement de la machine virtuelle et de l'application. Ces features seront ensuite utilisées pour la configuration des features du processeur présentées à la machine virtuelle par la machine hôte de destination. Ceci aidera à considérablement réduire le nombre d'échecs de migrations entre les environnements hétérogènes.

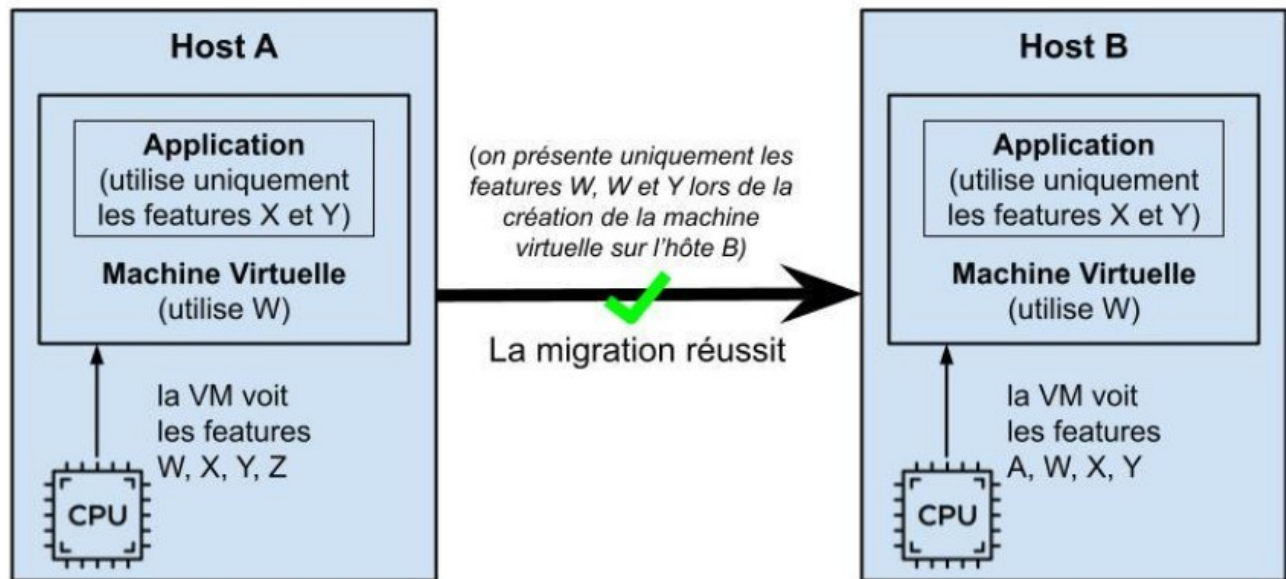


Figure 2: Solution

## II. Prerequis

### 1. Liste des prerequis

Pour pouvoir faire marcher l'application proposee comme solution, il faut:

- Avoir un processeur intel
- un systeme d'exploitation ubuntu 20.04
- Avoir au moins 10GO de libre sur le disque dur
- Avoir au moins 8GO de RAM
- Avoir un hyperviseur xen installe sur cette machine, ainsi que l'outil xen-tools
- Avoir python3,github installes
- installer et configurer nfs
- avoir une VM creee avec xen-tools possedant une OS ubuntu 18.04

### 2. Notations utilisees

- Commandes a utiliser executer dans un terminal se presentent sous la forme **command**

Dans le cas contraire, il s'agira d'un texte a renseigner dans l'editeur nano

- les textes ou variable sont placee entre **[] exple :** [ parametre ] veuillez remplacer [parametre ] par la valeur du parametre indique entre crochets

### III. Guide d'installation

-Ouvrir un terminal tant que superutilisateur en tapant "Ctrl+Alt+t", et tapez la commande

**sudo su**

- Cloner le repository github suivant: <https://github.com/FrankFomekong/vitualisation.git>

vous pourrez utiliser la commande : **git clone**

**https://github.com/FrankFomekong/vitualisation.git**

- se positionner au niveau du repository clone en tapand la commande

**cd vitualisation**

- Rassurez vous d'avoir ces fichiers en tapant la commande **ls**

vous aurez le resultat suivant :

```
root@borel-HP-Stream-Notebook-PC-13:/media/borel/SD/ProjetVirtualisation/MainApplication# ls
count.py      genFile.py   launcher.py  loop.sh      mapping.txt  myvm.cfg      saveFlags.py  tmprun.sh
configCPUID  finalFeature.txt  install    launchVm.sh  main.sh      match.py      readTestSuiteResult.py  tmp
root@borel-HP-Stream-Notebook-PC-13:/media/borel/SD/ProjetVirtualisation/MainApplication#
```

Figure 3: repository

- se positionner dans le repertoire install en tapant la commande

**cd install**

- puis taper la commande **source install.sh**

- patienter jusqu'a la fin de l'installation

## IV. Configuration de la machine virtuelle

### A. Configuration de l'accès au réseau

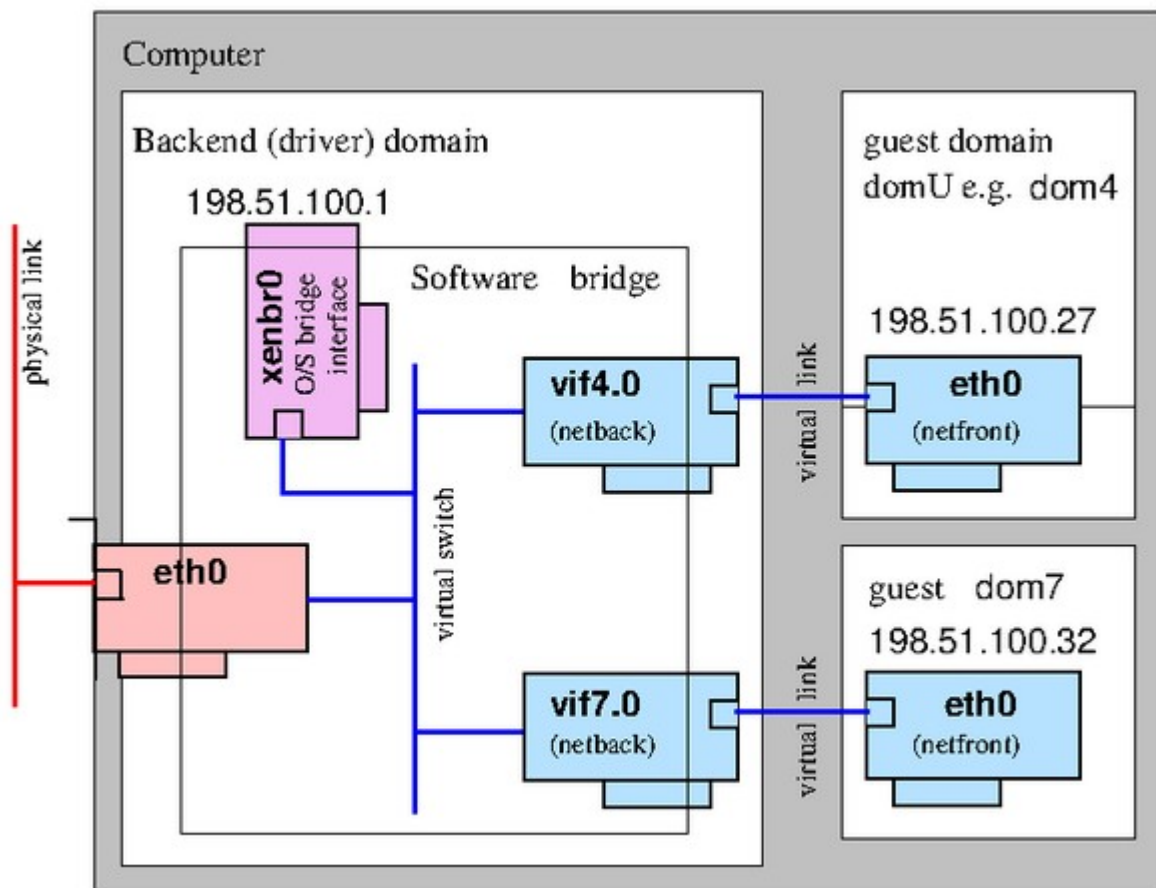


Figure 4: reseau

L'objectif ici est de créer une connection entre l'hôte (dom0) et les machines virtuelles (domU), de sorte que le ping de l'un à l'autre marche. Les paramètres de notre environnement sont les suivants:

- nom de la machine virtuelle: myvm
- adresse du réseau LAN partagé par le dom0 et le domU: 192.168.1.0
- adresse du pont xenbr0 dans le réseau LAN partagé par le dom0 et le domU: 192.168.1.101
- adresse de l'interface eth0 du domU: 192.168.1.102



## 1. Configuration du niveau du dom0

1. définir le mode de connexion entre le dom0 et le domU:

Nous avons choisi le mode **bridge** (script = vif-bridge). Dans le fichier de configuration de la machine virtuelle (`/etc/xen/myvm.cfg`), au niveau du paramètre vif, ajouter les paramètres **script** et **bridge** comme dans l'exemple suivant:

```
vif = ['ip=10.0.0.1, mac=00:16:3E:8B:54:20, script=vif-bridge,
bridge=xenbr0' ]
```

1. définir le pont utilisé par le domU (déjà fait à l'étape 1.a)

Ceci se fait au niveau du paramètre vif du fichier `/etc/xen/myvm.cfg` (bridge=xenbr0).

2. créer le bridge qui sera utilisé pour la connexion du domU au dom0

```
brctl addbr xenbr0
```

3. attribuer une adresse IP au bridge xenbr0 dans le LAN partagé par le dom0 et le domU

```
ifconfig xenbr0 192.168.1.101/24
```

4. activer le bridge xenbr0

```
ifconfig xenbr0 up
```

## 2. Configuration au niveau du domU (machine virtuelle)

Après avoir démarré la VM, exécuter les commandes suivantes:

1. Consulter les interfaces présentes

```
ip a
```

résultat:

```
root@myvm:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:16:3e:8b:54:20 brd ff:ff:ff:ff:ff:ff
```

Figure 5: ip domU

on observe qu'on a deux interfaces: **lo** et **eth0**. Nous allons utiliser l'interface **eth0** pour la configuration de la connexion au dom0.

## 2. Activer l'interface **eth0**

```
ip link set eth0 up
```

## 3. Attribuer une adresse ip à l'interface **eth0** dans le réseau lan partagé par le dom0 et le domU

```
ip addr add 192.168.1.102/24 dev eth0
```

## 4. définir l'interface eth0 comme passerelle vers le réseau LAN partagé par le dom0 et le domU

```
ip route add 192.168.1.0/24 dev eth0
```

Pour faciliter l'exécution de ces différentes étapes, les commandes peuvent être copiées dans un fichier qui sera exécuté sous forme de script bash.

# Connexion du domU à internet

Pour pouvoir configurer l'environnement pour l'exécution de l'application que nous avons choisie sur notre machine virtuelle, et également mettre en place une connexion ssh avec le dom0, il sera nécessaire que notre machine virtuelle ait accès à internet pour permettre le téléchargement et l'installation des paquets. Pour cela, nous devons mettre en place un NAT (Network Address Translation) au niveau du dom0 pour transmettre tous les paquets destinés au réseau extérieur provenant du LAN interne partagé par le dom0 et le domU. Pour cela nous nous sommes principalement inspirés de [ce site](#).

Nous avons suivi les étapes suivantes:

## 1. Configuration au niveau du dom0

### 1. Vérifier l'interface ethernet nous donnant accès à internet

```
ifconfig
```

résultat:

```

eno1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether d0:bf:9c:87:dc:11 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enx0c5b8f279a64: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.8.100 netmask 255.255.255.0 broadcast 192.168.8.255
    inet6 fe80::9c91:cf95:813d:83ff prefixlen 64 scopeid 0x20<link>
    ether 0c:5b:8f:27:9a:64 txqueuelen 1000 (Ethernet)
    RX packets 100277 bytes 102096865 (102.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 38532 bytes 9818225 (9.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Boucle locale)
    RX packets 9027 bytes 837934 (837.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0

```

Figure 6: internat interface dom0

Dans notre cas l'interface ethernet nous donnant accès à internet est **enx0c5b8f279a64**

## 2. NAT configuration with IP tables

```

iptables --flush
iptables --table nat --flush
iptables --table nat --delete-chain
iptables --table nat --append POSTROUTING --out-interface enx0c5b8f279a64 -
j MASQUERADE
iptables --append FORWARD --in-interface xenbr0 -j ACCEPT

```

## 3. Activer le forwarding des paquets au niveau du noyau du système:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

## 4. Appliquer la configuration

```
apt-get install iptables-persistent
```

Ce paquet est utilisé pour la persistance des configurations effectuées sur iptables. Lors de l'installation il sera demandé si l'on souhaite enregistrer la configuration courante d'ipv4 et ipv6 (répondre oui).

## 2. Configuration au niveau du domU

1. définir la passerelle par défaut pour les paquets destinés à internet

```
ip route add default via 192.168.1.101 dev eth0
```

2. Indiquer l'adresse du serveur DNS

Dans notre cas on va utiliser le serveur DNS de google (8.8.8.8). Aller dans le fichier

`/etc/resolv.conf` et le modifier comme suit:

```
nameserver 8.8.8.8
```

3. Tester la connexion à internet

```
ping google.com
```

Pour faciliter l'exécution de ces différentes étapes, les commandes peuvent être copiées dans un fichier qui sera exécuté sous forme de script bash.

## B. Configuration du service pour les tests suites

Une fois que la VM démarre et mise en réseau( avec un accès internet):

- installer l'éditeur nano en utilisant la commande `sudo apt-get install nano`
- installons les dépendances de postgresql en tapant successivement les commandes :

```
apt update
```

```
apt-get install libncurses5-dev libreadline6 libreadline6-dev zlib1g zlib1g-dev
```

```
apt-get install flex bison libssl-dev pkg-config make
```

- Ensuite installer et configurer nfs, sur la vm, tapez successivement les commandes

```
apt install nfs-common
```

```
mkdir -p mntnfs_clientshare
```

```
mount [ip du dom0]:/mnt/nfs_clientshare
```

- créer un nouvel utilisateur:

```
adduser [nom de votre nouvel utilisateur]
```

ensuite renseigner les informations relatives à l'utilisateur

**NB: il ne doit pas appartenir au groupe root. De plus retenir le [nom d'utilisateur] que vous allez renseigner il sera utilisé plus bas.**

- puis creer le script pour le service qui recupera les features et lancera les tests suite au demarrage,pour cela :

- tapez la commande **nano Network.sh** ,puis inserer les lignes ci apres:

*ip link set eth0 up*

*ip addr add 192.168.1.102/24 dev eth0*

*ip route add 192.168.1.0/24 dev eth0*

*ip route add default via 192.168.1.101 dev eth0*

*mount \$1:/mnt/nfs\_share /mnt/nfs\_clientshare*

*/mnt/nfs\_clientshare/main >features.txt*

*cd mntnfs\_clientshare/postgresql-12.0 && sudo -u [Nom de l'utilisateur cree recemment]  
make check >test.txt*

- Puis tapez les combinaisons de touche “CTRL+S” et “CTRL+X”

- puis tapez la commande *chmod +x Network.sh*

- Ensuite taper la commande **nano /etc/systemd/system/networkxen.service**

et inserer les lines suivantes :

**Unit]**

**Description=Create xenbr0 interface systemd service.**

**[Service]**

**Type=simple**

**ExecStart=/bin/bash /root/Network.sh [adresse ip du dom0]**

**[Install]**

**WantedBy=multi-user.target**

- Puis taper les combinaisons de touche “CTRL+S” et “CTRL+X”

- puis tapez les commandes suivante pour enregistrer et activer le service

**systemctl daemon-reload**

**systemctl enable networkxen.service**

## C. Configuration sur le dom0

A la racine du repository github clonee precedemment:

- tapez dans le terminal, la commande `nano myvm.cfg`
- verifiez les parametres suivants et les ajuster selon votre environnement:
  - ➔ emplacement des disques durs virtuels(disk,swap)

```
disk      = [
            'file:/home/xen/domains/myvm/disk.img,xvda2,rw',
            'file:/home/xen/domains/myvm/swap.img,xvda1,rw',
            ]
```

Figure 7: myvm.cfg

- ➔ nombre cpu et la quantite de RAM en Mbts

```
vcpus      = '5'
memory     = '3072'
```

Figure 8: myvm.cfg

- **Puis taper les combinaisons de touche “CTRL+S” et “CTRL+X”**

## V. Guide d'utilisation

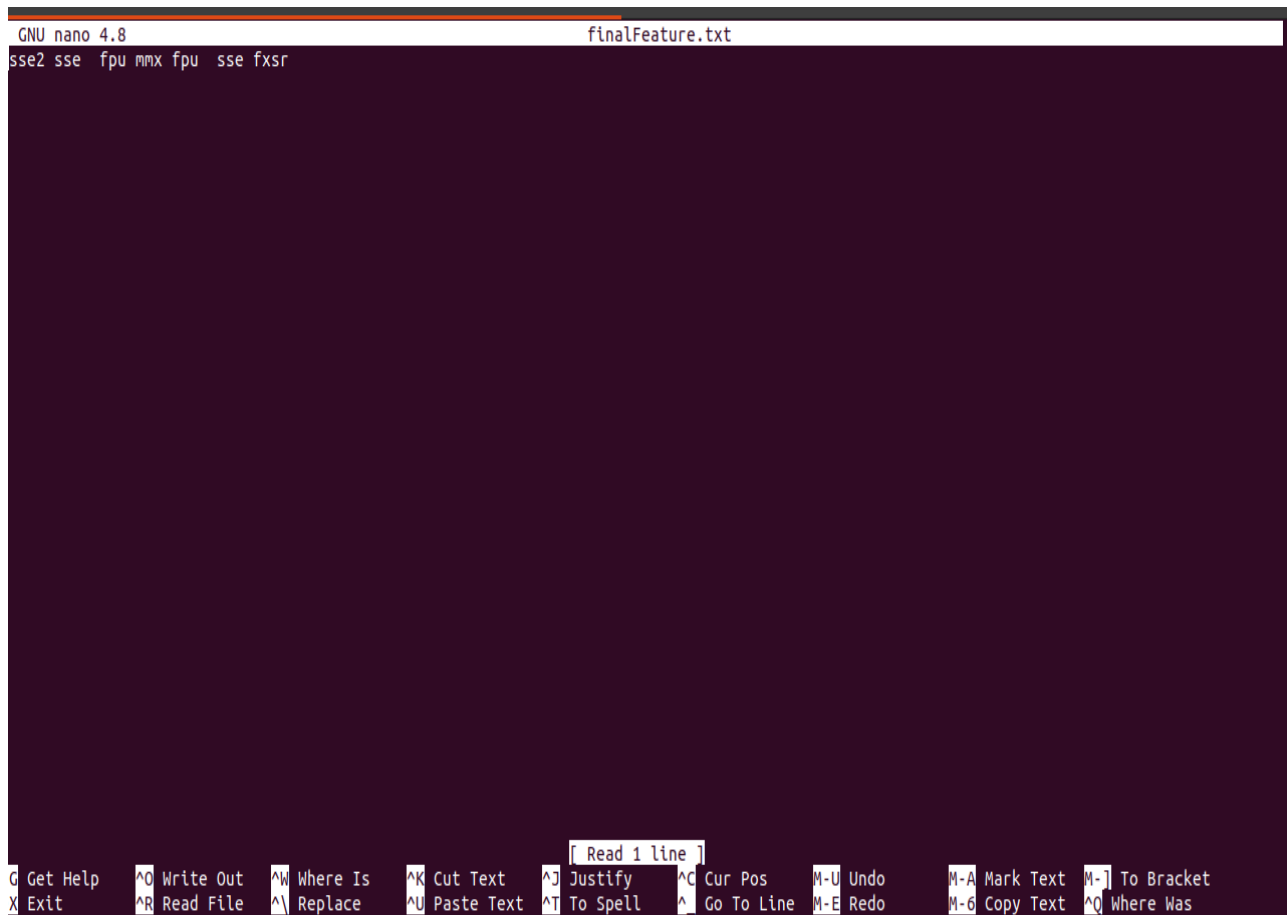
Dans le repertoire github (virtualisation) clonee precedemment, pour lancer le programme:

- taper dans le terminal la commande `source main.sh`
- Puis pateintez que le programme finisse en vous affichant le message

```
===== execution terminee consulter la liste des features necessaires trouvee dans le fichier finalFeature.txt =====
```

Figure 9: app result

- puis pour consulter le resultat du programme,taper la commande `nano finalFeature.txt`
- Vous aurez ceci (les features suivantes sont obtenus dans notre cas)



```
GNU nano 4.8 finalFeature.txt
sse2 sse fpv mmx fpv sse fxsr
```

[ Read 1 line ]

G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket  
X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^\_ Go To Line M-E Redo M-G Copy Text ^Q Where Was

Figure 10: features list

# ANNEXES

## Installer et configure xen , xen-tools

- ➔ executer les commandes ci apres en tant que root, vous pouvez taper **sudo bash**
- ➔ mettre a jour vos packages  
**apt-get update**
- ➔ chercher le package de l'hyperviseur xen  
**apt-cache search xen | grep hypervisor**
- ➔ chercher le package le plus recent compatible avec votre OS. Dans notre cas il s'agit de **xen-hypervisor-4.11-amd64**  
**apt-get install xen-hypervisor-4.11-amd64**
- ➔ configurer le grub pour pouvoir demarrer depuis l'hyperviseur xen  
**nano /etc/default/grub**  
  
GRUB\_DEFAULT=0  
  
#GRUB\_TIMEOUT\_STYLE=hidden  
  
GRUB\_TIMEOUT=10
- ➔ Mettre a jour le grub pour prendre en compte nos modifications  
**update-grub**
- ➔ redemarrer et selectionner "boot on xen-hypervisor" au demarage,tapez la commande  
**reboot**  
  
pour redemarer
- ➔ Verifier si Xen fonctionne,taper la commande  
**xl list**



- Installer `xen-tools`: un ensemble d'outils pour manipuler les VM

```
apt-get install xen-tools
```

- Verifier si on a les lignes suivantes dans `/etc/xen-tools/xen-tools.conf`

```
#####VM images will be installed in /home/xen-images
```

```
dir = /home/xen-images
```

```
#####xen-create-image (see below) will use this image to download and install a Linux  
distribution file system, that will be used by your VM.
```

```
install-method = debootstrap
```

- Créer une image (le disque de votre VM) et sauvegarder les mots de passes obtenus de la commande ci apres:

```
xen-create-image --hostname myvm --force --ip 10.0.0.2 --vcpus 10 --pygrub --dist bionic
```

- le fichier de configuration `/etc/xen/myvm.cfg` contiendra ces informations.

```
#  
# Configuration file for the Xen instance myvm, created  
# by xen-tools 4.8 on Fri Sep 16 06:52:23 2022.  
#  
#  
# Kernel + memory size  
#  
bootloader = 'pygrub'  
vcpus      = '5'  
memory     = '3072'  
#  
# Disk device(s).  
#  
root       = '/dev/xvda2 ro'  
disk       = [  
              'file:/home/xen-images/domains/myvm/disk.img,xvda2,rw',  
              'file:/home/xen-images/domains/myvm/swap.img,xvda1,rw',  
            ]  
#  
# Physical volumes  
#|  
#  
# Hostname  
#  
name       = 'myvm'
```

Figure 11: contenu de `myvm.cfg`

```
#
# Hostname
#
name      = 'myvm'

#
# Networking
#
dhcp      = 'dhcp'
vif       = ['ip=10.0.0.1, mac=00:16:3E:0F:E2:AB, script=vif-bridge, bridge=xenbr0' ]

#
# Behaviour
#
on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

*Figure 12: contenu 2 de myvm.cfg*