RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ AMMAR TELIDJI LAGHOUAT

Faculté des sciences

Département de mathématiques et informatique

Spécialité : Informatique

Mémoire en vue de l'obtention du Diplôme de Master de recherche en informatique

Option :Réseaux et applications systémes répartis

Thème :

Virtualisation avec UML

Présentée Par : Mechraoui imane

Hegga meriem

Soutenu devant le jury composé de :

Mlle Abdelhafidi Zohra	MAA	Président	Université de Laghouat
Mr Oubbatti Sami	MAB	Examinateur	Université de Laghouat
Mr Ameur Mohamed el amine	MAA	Examinateur	Université de Laghouat
Mlle Belabbaci Amel	MAA	Encadreur	université de Laghouat

Année Universitaire : 2015-2016

Remerciement

Je remercie ***Dieu*** *tout Puissant de m'avoir permis de mener à terme ce projet qui est pour moi le point de départ d'une merveilleuse aventure, celle de la recherche, source de remise en cause permanente et de perfectionnement perpétuelle, qui a bénéficié des conseils scientifiques des uns, de l'appui moral et du soutien financier des autres.*

Je remercie mon encadreur ***M Amel belabbaci*** *, et mes professeurs de la base jusqu'au sommet.*

Enfin un grand merci à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Dédicaces : H.Meriem

Je rends grâce à ***ALLAH*** *de m'avoir donné le courage et la volonté. Ainsi que la conscience d'avoir pu terminer mes études.*

Je dédie ce modeste travail....

A ma très chère et douce mère, A mon très cher père à qui m'adresse au ciel les vœux les plus ardents pour la conservation de leur santé et de leur vie.

.

A mes soeurs ***sarah*** *,* ***safaa*** *,* ***hadjer*** *,* ***amina*** *..... Je vous souhaite beaucoup de bonheur et de réussite. .*

A ma petite chère nièce yasmine .

A toute ma famille... grands et petits..

A mon binôme Imane et sa famille.

A tous les enseignants et étudiants de département mathématiques et informatique..

A tous mes collègues..

A tous mes chers amis(es) surtout boukhari sarah ,naima daradji..

A tous ceux qui m'aiment..

A tous ceux que j'aime..

Dédicaces : M.Iman

Je dédie ce travail :

A mes chers parents qui m'ont aidés et soutenus. Mes cheres sœurs ***soumia*** *et* ***maroua*** *et mon cher frère* ***taher***

.

Auxquelles je souhaite beaucoup de succès et de réussite.

A mes chères amies :

O.Samira , G.fatna ,S.khadija,B.saraha .

A mon meilleure amie et mon binôme H. meriem.

Résumé

La virtualisation présente de nombreuses réponses à des problèmes qui se posent aujourd'hui que ce soit au niveau logiciel avec une nécessité de sécurisation, ou au niveau énergétique avec la forte augmentation du coût de l'énergie électrique
à travers ce projet on a étudié la techniques de virtualisation User mode linux ainsi que leur spécificité. ce qui nous a permis de :

• Montrer son rôle pour apprendre l' administration de system linux

• Création d'un laboratoire virtuel en interconnectant Plusieurs machines virtuelles au moyen d'interface TAP Et Switch VDE

• Partage des fichiers entre machines distance utilisant le logiciel « samba »

• Compiler un noyau linux pour avoir un nouveau noyau
Mot clés : Virtualisation, user mode linux, Interface TAP , Switch VDE ,Samba , Noyau .

Abstract

The virtualisation presents numerous answers to problems that arise today whether it is for the software level with a necessity of reassurance, or for the energy level with the strong increase of the electric energy cost
Through this project we studied techniques of virtualisation To use mode Linux as well as their specificity. What allowed us of :

• Show his role to learn the administration of system Linux

• Creation of a virtual laboratory by interconnecting Several virtual machines by means of interface TAP and Switch VDE

• Division of the files between machines outstrip using the software "samba"

• Compile a pit Linux to have a new pit(core)
Keywords : Virtualisation, to use mode Linux, Interfaces TAP, Switch VDE, Samba, Pit .

Sommaire

Table des figures

Introduction générale

La virtualisation et une technologie qui permet d'exécuter simultanément plusieurs systèmes d'exploitation et plusieurs applications sur un méme ordinateurs.
Plusieurs logiciels de virtualisation existentont cité VMware,Virtual box,Xen,User Mode Linux(UML) qui sont diffèrent en matière de performances et de fonctionnalité.parmi ces logiciels on est va étudier par la virtualisation UML.
Plusieurs simulateurs (émulateurs travaillent de la méme maniére qu'UML (création de laboratoire) comme Netkit,Vnuml,et Marionnette[5] .
C'est la raison pour laquelle, on souhaite étudier UML en détail.
UML rend possible la création de maquettes pédagogiques virtuelles de réseaux, en interconnectant Plusieurs machines virtuelles au moyen d'interfaces ethertap. Un micro-ordinateur de faible puissance (exemple : Pentium III) peut supportait un réseau virtuel composé d'un routeur firewall interconnectant trois sous réseaux de deux postes virtuels chacun Une des machines virtuelles configurée en routeur firewall assurait Le lien entre la machine réelle et l'ensemble du réseau virtuel.

dike jeff est le contributeur du kernel UML depuis 1999 et assure encore aujourd'hui la responsabilité et le développement du projet UML, travailler avec uml c'est just un habby pour lui il a reçu beaucoup d'aide de « Dartmouth ISTS » pour ce travail concernant les pots de miel (honeypot). Il a aussi eu un support occasionnel d'entreprises qui voulaient que des fonctionnalités soient ajoutées ou que des bugs soient corrigés, selon Jeff UML va devenir véritable composant d'un OS. Il pourra être combiné arbitrairement avec le noyau de l'hôte ou avec d'autres applications.
Il pourra fournir au noyau le contrôle de ressources qui n'existent pas physiquement. Combiné à d'autres applications, il fournira de nouveaux types de fonctionnalités, aussi bien en termes de qualité, d'efficacité que d'évolutivité
Le but de ce mémoire et d'étudier UML et ses applications d'utilisation.il est composé de trois chapitre

Le premier chapitre est la présentation d ' UML
Le second chapitre est l'utilisation d'uml dans environnement d'éducation
Le troisième chapitre est la création des laboratoires virtuels ave UML

Chapitre 1

UML

1.1 Introduction

Dans ce chapitre, on va présenter UML(User Mode Linux) : son objectif , son architecture, ses avantages et ses inconvénients. On va voir également comment préparer et utiliser une machine virtuelle UML .

1.2 Définition UML (User Mode Linux)

UML est une machine virtuelle Linux qui fonctionne sur Linux pour être exécuté dans l'espace utilisateur comme n'importe quel processus. Il permet d'exécuter plusieurs machines virtuelles et chaque machine peut étre liée par un kernel spécifique [1]. Chaque machine virtuel UML a une console qui permet d'exploiter la machine virtuelle en ligne de commande, et une mémoire, qui est un bout de la mémoire de la machine hébergeuse,et un système de fichiers sauvegardé dans un fichier de système de la machine hébergeuse.

1.3 Objectif de UML

on utilise UML pour :

• Concevoir, déployer, sécuriser et maintenir des plateformes virtuelles hétérogènes supportant des systèmes d'exploitation.

• Se familiariser avec l'administration Linux, Il faut disposer d'un matériel suffisant.

• Comprendre et apprécier objectivement les diverses techniques de virtualisation (hyperviseur, émulation, ...).

• Construire de réseaux virtuels.

1.4 Architecture de UML

l'architecture de UML fournit un environnement autonome identique à celui d'une machine Linux, Chaque processus crée par une machine virtuelle est d'abord intercepté et traité par le kernel UML puis passe au kernel de la machine physique [2].

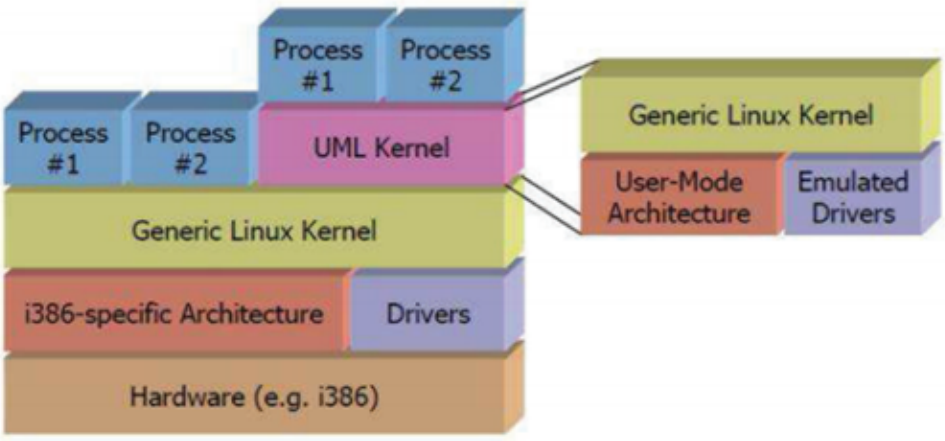

FIGURE 1.1 – Architecture de UML [3] .

1.5 préparation d'um machine virtuelle UML

l'installation de UML, peut etre faite comme n'imporet qu'elle application sous linux,ensuite on doit lier la machine UML avec un noyau linux. On peut connecter la machine UML avec le noyau de système comme on peut le connecter avec n'importe quel autre noyau. La première solution est facile mais elle présente un risque pour les débutants. En effet, si on utilise le noyau système, tous changement dans la machine virtuelle. On peut utiliser le noyau de système local c'est vrai qu'un a une machine virtuelle dans une machine physique, mais cette utilisation a permet à tout ce qui est passé

dans la machine virtuel fais des changement mémé sur la machine réelle donc c'est risqué d'utiliser le noyau .
Après son installation, UML peut être lancé avec la commande linux, si notre machine est lié avec le système, on indique en plus le chemin vers le fichier d'initialisation[4]

Remarque : à ne pas confondre avec les commande linux linux 32 et linux 64.

1.5.1 Résultat de Fichiers COW et Rootfs :

UML permet à plusieurs machines virtuelles de se partager un root-fs commun en Lecture seule. Les opérations d'écriture de chacune des machines virtuelles sont déportées dans des fichiers séparés propres à chaque machine virtuelle. Ces fichiers ne contenant que les différences avec le root-fs, sont dénommés fichiers COW (abréviation de Copy On Write).
UML a introduit les fichiers Copy On Write, Plusieurs VM se partagent un rootfs commun accessible en lecture seule, Chaque machine virtuelle dispose d'un espace de débordement dit Copy On Write dans lequel sont déportées les surcharges apportées au rootfs de référence [6].

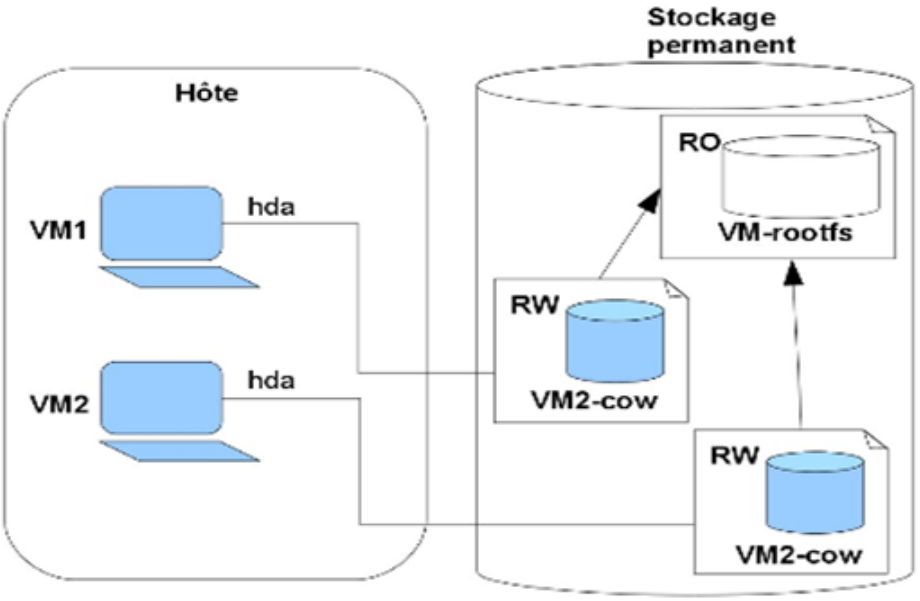

FIGURE 1.2 – Fichiers Rootfs et Cow [7] .

1.6 Avantages de UML

De maniére génerale :

•Il permet de mettre en place un réseau complètement virtuel de machines Linux, pouvant communiquer entre elles.

• Les tests de topologies lourdes d'un point de vue physique peuvent donc être testé .

•Si UML se bloque, le noyau hôte est toujours reste en bon état.

•L'exécution de Linux sur une autre distribution Linux, sans avoir besoin d'un accès privilégié

•Si UML est connecté avec un noyau autre que celui du systéme,il permet en plus de :

•Un utilisateur sera root sur un user Mode Linux, mais pas sur le système hôte.

•Si un User Mode Linux plante, le système hôte n'est pas affecté.

•Il permet aussi de tester différents paramètres noyaux sans se soucier des conséquences.

•Il permet de tester différentes configurations ou compilations du noyau sans avoir à l'installer et redémarrer la machine.

1.7 Inconvionnemnt UML

le seul défaut remarqué jusqu' à présent,et qu ' il est
• très lent, plutôt conçu pour des tests fonctionnels que pour la performance

1.8 Application de UML

UML peut être utilisé dans plusieurs applications, parmi ces applications on cite :
• **Débogage du noyau linux**
Les développeurs du noyau et les projets ont commencé à utiliser UML comme principal la plate-forme de développement, et démarrer leurs travaux sur machines physiques plus tard aprés débogage.
• **Réalisation de réseaux virtuels**
UML set largement utilisé pour créer des réseaux virtuels(des laboratoire) dans le but de faire des simulations
• **Utilisation dans un environnement d'éducation**
UML est très utile pour apprendre l'administration du système linux et ses commandes,surtout lorsqu'il est installé et connecté avec un autre noyau autre que le noyau du systéme.

1.9 UML et la sécurité

UML a des caractéristiques de sécurité idéales pour le contrôle et l'emprisonnement des pirates, UML peut enregistrer tout le trafic de terminal à l'hôte d'une manière qui est invisible et impossible à interférer avec l'intérieur de l'UML contrairement Xen et VMware.
En ce qui concerne la sécurité, la première chose à considérer est de savoir si les UML accueilleront les utilisateurs potentiellement hostiles, qui ne savent pas beaucoup sur ses clients. Il est aussi peut-être le cas avec un UML qui est exposé à l'Internet, auquel cas l'utilisateur hostile serait quelqu'un qui a rompu en elle. Pour UML sur un intranet dont les propriétaires ont des comptes sur l'hôte, ce qui a probablement ne serait pas un problème .
UML offre une sécurité suffisante pour être utilisé comme un honeypot, il empêchant le honeypot d'être utilisé comme une base pour de nouvelles attaques exigent que ce soit facile à arrêter ou retiré du réseau rapidement, mais il ne

peut pas être utilisé comme une plate-forme pour attaquer d'autres systèmes [8] .

1.10 Conclusion

UML est un environnement léger et facile à installer et à utiliser, il est livré avec un ensemble de technologies réseau prêts à être utiliser, il ne demande pas un matériel très performent, avec plusieurs machine on peut simuler un réseau complexe.

Chapitre 2

L'utilité de UML dans l'environnement d'éducation

2.1 Introduction

Dans ce chapitre, on va voir l'utilité de UML, dans l'environnement d'éducation : comment UML est utile pour l'apprentissage du système linux (les commandes linux et la compilation du noyau) ainsi que son administration.

2.2 Utilité d'Uml dans l'enseignements

Pour apprendre le system linux et son administration on a besoin d'essayer des commandes et des scripts , de pratiquer et parfois de risquer qui peuvent endommager notre matériels et système, UML permet d'éviter ce risque car c'est un environnement virtuel et rien ce qu'on fait sur la machine virtuelle UML peut changer ou endommager la machine réelle ou son logiciel .
(bien entendu, lorsqu'on est lié a un autre kernel autre que celui de notre systeme).
La moindre des choses l'administration du système exige que les étudiants aient des privilèges root sur les machines qu'ils apprennent sur tout les travaux seront sur des machines physiques sur un réseau physique et ca peut détruire complètement le logiciel de système ou endommager le matériel.
Tous ces problèmes ont des solutions dans un environnement physique mais ca prend la planification, la configuration, le temps et les ressources qui ne sont tout simplement pas nécessaires lors de l'utilisation d'un environnement UML. [10]

2.3 Utilisation UML pour apprendre le noyau linux

UML est aussi couramment utilisé pour apprendre la programmation au niveau du noyau Pour les étudiants, UML est un environnement parfait pour apprendre. Programmation du noyau fournit un authentique noyau pour modifier, avec les outils de développement et de débogage qui devrait déjà être familier. En outre, le matériel au-dessous de ce noyau est virtualisé et donc mieux comporté que le matériel physique. C'est à dire Les échecs seront causés par un logiciel buggé, pas par des dispositifs de mauvaise conduite.
Ainsi, les étudiants peuvent se concentrer sur le débogage du code plutôt que le diagnostic de matériel cassé Enfin il existe quelque établissements d'enseignement qui font des cours d'administration Linux en utilisant UML Certaines sociétés commerciales offrent même des cours d'administration système sur l'Internet en utilisant UML. Chaque élève se voit attribuer une UML personnelle, qui est accessible sur Internet, et l'utilise pour compléter le cours.

2.4 L exécution des commandes dangereuse en Uml

Si UML est connecté avec un kernel autre que celui du système, on peut essayer des commandes dangereuses sous UML sans endommager notre système.

La commande fdisk

Est un outil de base pour réaliser des opérations sur les tables de partitions des disques durs, elle permet de manipuler les tables de partitions. Il permet de créer, de supprimer, de lister les partitions sur un disque dur. Voyons la syntaxe des différentes opérations

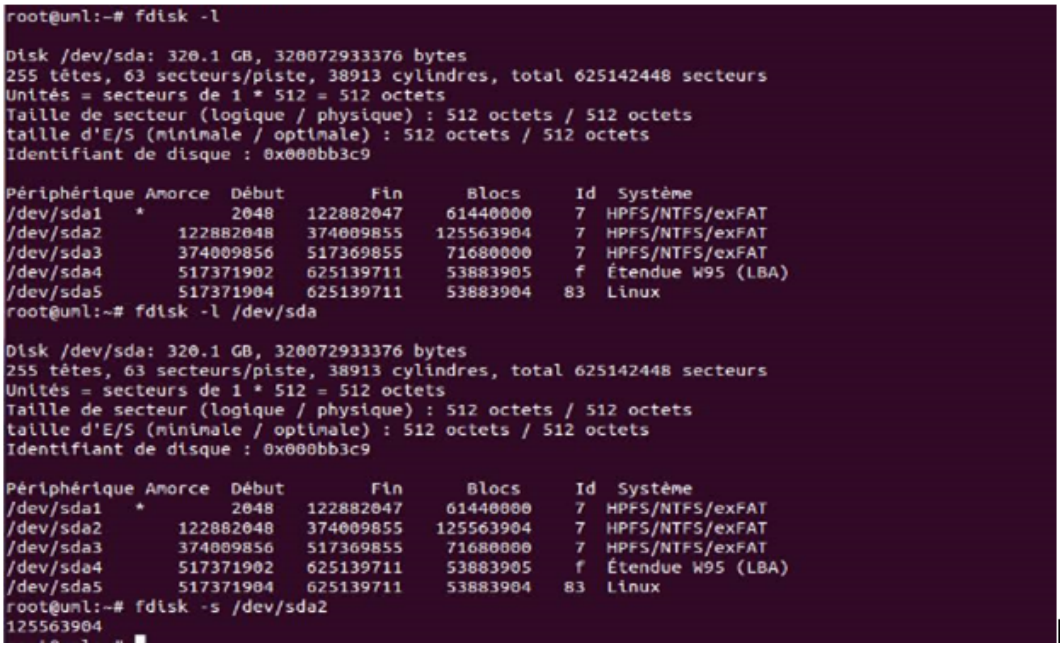

FIGURE 2.1 – Commande fdisk sous user mode linux .

La commande dd :

dd permet de copier tout ou partie d'un disque par blocs d'octets, indépendamment de la structure du contenu du disque en fichiers et en répertoires, dd permet de faire beaucoup de choses intéressantes, mais elle peut aussi être dangereuse [12]

Syntaxe : dd if=<source> of=<cible> bs=<taille des blocs> skip= seek= conv=<conversion>

source représente les données à copier, cible est l'endroit où les copier ; ça se comprend : if correspond à l'input file et of correspond à l'output file

La commande hdparm :
Est un utilitaire logiciel pour obtenir ou positionner les paramètres de disque dur. Il s'utilise en mode console ou par le biais d'un fichier de configuration
Syntaxe : hdparm [drapeau] [périphérique]

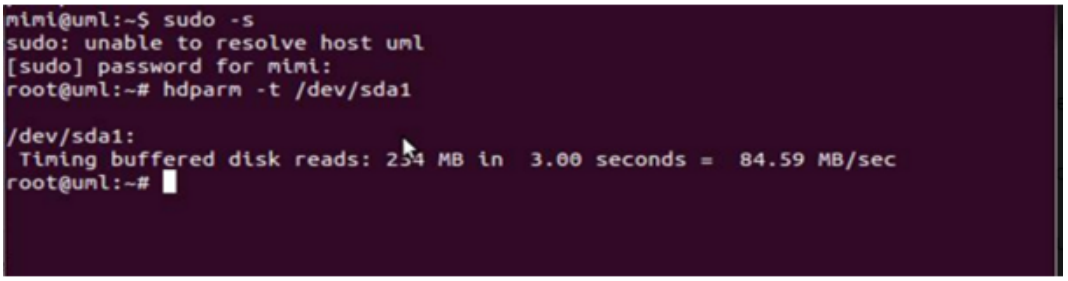

FIGURE 2.2 – Commande hdparm sous user mode linux .

La commande mkfs.ext3 /dev/sda :
Mkfs est une commande utilisée pour formater les disques durs. Ici elle effacera les données du disque /dev/sda.
mkfs.ext3 /dev/sda

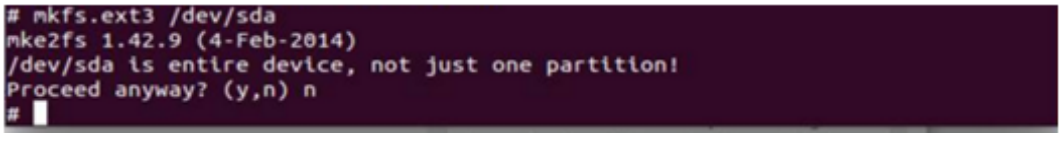

FIGURE 2.3 – Commande mkfs sous user mode linux .

La commande rm -rf / :
Cette commande efface tout simplement tous les fichiers contenus sur votre disque dur

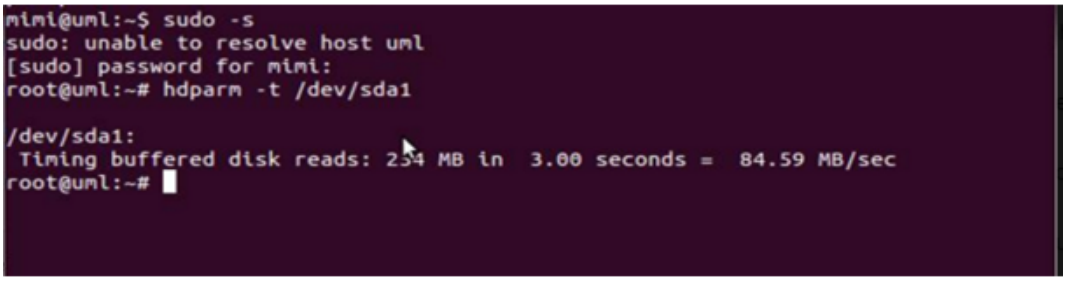

FIGURE 2.4 – Commande rm-rf/ sous user mode linux .

La commande rm –rf* :

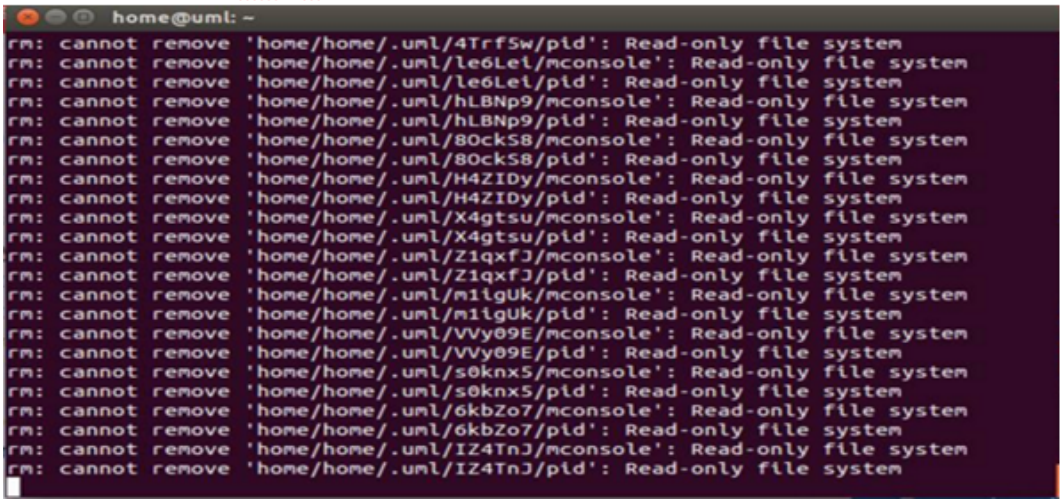

FIGURE 2.5 – Commande rm-rf* sous user mode linux .

La commande :() :| :; :
Cette commande bash, pleine de caractères bizarres, est en fait une bombe fork. Si vous la lancez, votre système sera contraint de créer des processus jusqu'à saturation de la mémoire. Et ceci provoquera donc le plantage de l'ordinateur [11].

2.5 Remarque et discussions

Pour essayer ces commandes sans endommager notre system ,UML doit etre lie a un kernel autre que celui du système . En effet on donne notre exemple, lorsque on a executé la commande rm -rf* qui permet d'effacer tous les fichiers de disque dur sous UML tous ca marche bien mais lors de redémarrage d'ordinateur c'était la surprise , le système est endommager on conclus que cette commande a toucher la machine car notre UML est lie avec le kernel de systeme .

2.6 Compilation du noyau linux

2.6.1 Définition d'un noyau (kernel)

le noyau d'un système d'exploitation est le logiciel qui assure la communication entre les logiciels et le matériel Son rôle central impose par ailleurs des performances élevées. Cela fait du noyau la partie la plus critique d'un système d'exploitation et rend sa conception et sa programmation particulièrement délicates. Plusieurs techniques sont mises en œuvre pour simplifier la programmation des noyaux tout en garantissant de bonnes performances[14].

FIGURE 2.6 – Noyau linux et ces fonctionnalité .

Les noyaux ont comme fonction de base d'assurer le chargement et l'exécution des processus, de gérer les entrées/sorties et de proposer une interface entre l'espace noyau et les programmes de l'espace utilisateur. À de rares exceptions, les noyaux ne sont pas limités à leurs fonctionnalités de base. On trouve généralement dans les noyaux les fonctions des micro-noyaux [15] prenons le gestionnaire de mémoire et l' ordonnanceur, ainsi que des fonctions de communication inter-processus [16].

2.7 Objectif de la compilation du noyau

La compilation du noyau est de construire un kernel qui est en adéquation avec le matériel. Les kernels livrés avec les distributions sont en général assez

gros, car ils intègrent une grande quantité de drivers pour pouvoir fonctionner sur le plus grand nombre de machines possibles. Le problème est que ces drivers consomment beaucoup de mémoire, autant laisser la mémoire disponible pour les applications ,la compilation du noyau permet aussi de faire reconnaître ses propres périphériques comme un lecteur ZIP, une carte son ou un graveur de CD.
On peut citer la quelques raison, de compilation du noyau :

- 1. Comprendre comment fonctionne le noyau Linux.
- 2. Faire fonctionner un matériel qui n'est pas pris en charge par de noyau actuel.
- 3. Appliquer un correctif.
- 4. On peut utiliser une distribution qui oblige de compiler le noyau
- 5. Activer des options qui ne sont pas incluses dans le noyau par défaut
- 6. Gérer des périphériques spéciaux, ou des conflits de périphériques dans les noyaux par défaut [16]

2.8 Les étapes de complication d'un noyau :

2.8.1 Exemple installation de noyau (vanille) avec une distribution debian :

Tout d'abord, téléchargez le paquetage kernel : sudo apt-get install linux-tree Pour le processus de compilation, on a besoin des paquets suivants :

• sudo apt-get update
• build-essential
• kernel-package
• apt-get install gcc
• l libncurses5
• l libncurses5-dev
• libqt3-mt-dev

• kernel

Maintenant, on peut commencer à personnaliser1 la configuration du noyau par les commandes :
• make xconfig (on menuconfig) Puis, après avoir terminé la personnalisation, nous devons commencer le processus de compilation
• sudo make-kpkg clean
• sudo make-kpkg –append-to-version=-custom kerne-image modulesimage
Il est possible de personnaliser le nom de noyau en lui ajoutant un champ

–append-to-version Pour cela, avant de lancer la compilation, on va éditer le fichier Makefile qui se trouve à la racine des sources et renseigner ce champ maintenant on a un paquet .deb dans / usr / src prêt à être installé comme tout autre paquet par un simple doubleclic.
Grub sera mis à jour automatiquement. Donc,on execute simplement la commande suivente :

• sudo dpkg -i kernel-image-2.6.12-custom$_1$0.00.$Custom_i$386.deb[17]

2.9 la compilation avec UML :

Il existe d'autre application (machine virtuel) capable de faire compiler un noyau linux. Cependant, ils ne sont pas aussi efficaces que UML. Par exemple, "chroot" pose le probléme de la securité. Lorsque le mot de passe Root est trouvé, la machine virtuelle ne rempli alors plus sa fonctionnalité de "prison".[9]

2.10 Un débogage du noyau après la compilation :

Au cour de compilation il peut y avoir création des bugs c'est pour cette raison il est préférable de faire un débogage après une compilation pour avoir un system sans anomalies
GDB :
Le logiciel gdb est un logiciel GNU permettant de déboguer les programmes C (et C++). Il permet de répondre aux questions suivantes :
• à quel endroit s'arrête le programme en cas de terminaison incorrecte, notamment en cas d'erreur de segmentation ?
• Quelles sont les valeurs des variables du programme à un moment donné de l'exécution ?
• Quelle est la valeur d'une expression donnée à un moment précis de l'exécution ?
Gdb permet donc de lancer le programme, d'arrêter l'exécution à un endroit précis, d'examiner et de modifier les variables au cours de l'exécution et aussi d'exécuter le programme pas-à-pas.
Exemple :

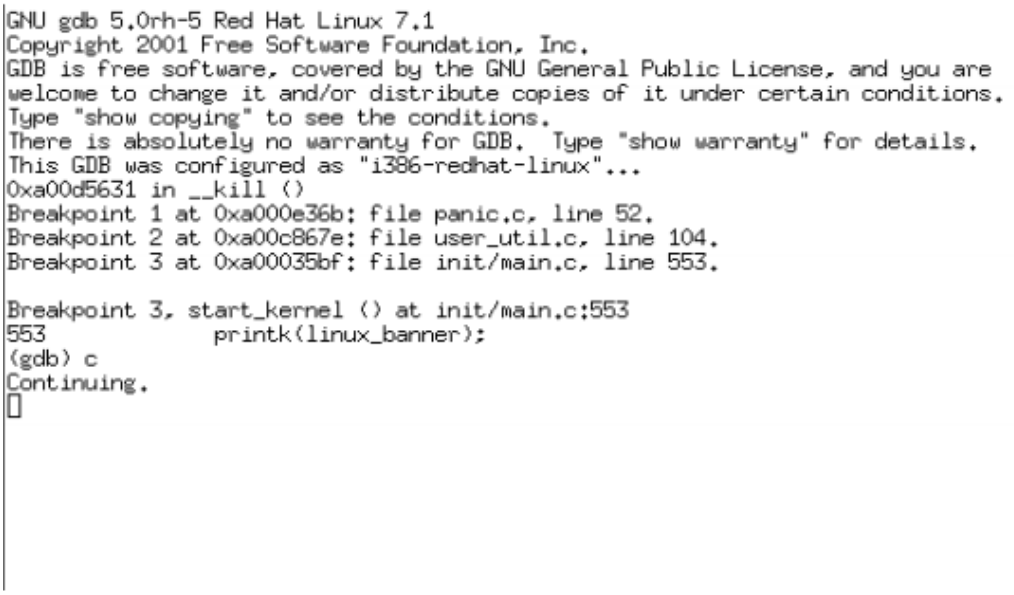

FIGURE 2.7 – Exemple de débogage avec GDB .

KDB :
Debogueur fonctionnant sur la machine a deboguer (dans le handler du clavier). Permet de :

- Interrompre l'execution du noyau ;
- Mettre des breakpoints (execution, lecture/ecriture) ;
- Exécuter le code en mode pas a pas ;
- Connaitre la pile d'appel (tache courante, toutes les taches) ;
- Examiner/modifier la mémoire (adresses, listes chainées etc) ;

• Examiner/modifier les registres des processeurs.
• Débogueur local a la machine (pas de matériel supplémentaire).
• Débogueur au niveau assembleur

Exemple :

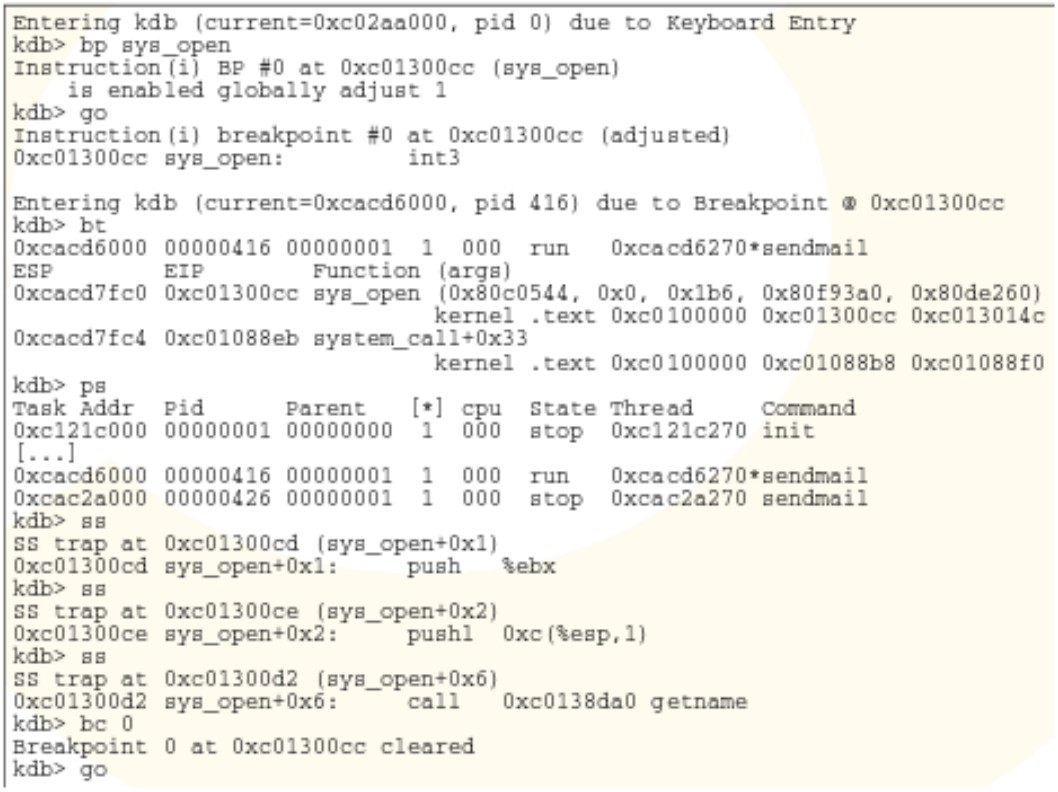

FIGURE 2.8 – Exemple de débogage avec KDB .

KGDB :

Permet de :

- Interrompre l'execution du noyau ;
- Mettre des breakpoints (execution, lecture/ecriture) ;
- Executer le code en mode pas a pas ;
- Connaıtre la pile d'appel (tache courante, toutes les taches) ;
  Examiner/modifier la memoire (adresses, variables, listes chaınees etc) ;
- Toute fonctionnalite de gdb standard ...
- Debogueur au niveau source.
- Deux machines necessaires

Exemple :

```
Program received signal SIGTRAP, Trace/breakpoint trap.
breakpoint () at gdbstub.c:1177
1177    }
(gdb) break sys_open
Breakpoint 2 at 0xc0131ef7: file open.c, line 783.
(gdb) cont
Continuing.

Breakpoint 2, sys_open (filename=0x80c0544 "/proc/loadavg", flags=0, mode=438)
    at open.c:783
783             tmp = getname(filename);
(gdb) list
778             int fd, error;
779
780     #if BITS_PER_LONG != 32
781             flags |= O_LARGEFILE;
782     #endif
783             tmp = getname(filename);
784             fd = PTR_ERR(tmp);
785             if (!IS_ERR(tmp)) {
786                     fd = get_unused_fd();
787                     if (fd >= 0) {
(gdb) next
...
786                     fd = get_unused_fd();
(gdb) next
787                     if (fd >= 0) {
(gdb) print fd
$1 = 5
(gdb) where
#0  sys_open (filename=0x80c0544 "/proc/loadavg", flags=0, mode=438)
    at open.c:787
#1  0xc0107323 in system_call () at af_packet.c:1891
(gdb) clear sys_open
Deleted breakpoint 2
(gdb) cont
Continuing.
```

FIGURE 2.9 – Exemple de débogage avec KGDB .

UML : User Mode Linux

Virtualisation du noyau Linux.

• Visible comme une architecture supportee par Linux : um.

• Processus standard permettant l'utilisation "normale" de gdb ,gprof etc.

• Facilite de debogage.

• Pas de debogage materiel.

Exemple :

```
$ ./linux debug
GNU gdb Red Hat Linux (5.2.1-4)
...
(gdb) break sys_open
Breakpoint 4 at 0xa003d384: file open.c, line 808.
(gdb) cont
Continuing.

Breakpoint 4, sys_open (filename=0xa01cbcca "/dev/console", flags=2, mode=0)
    at open.c:808
808             tmp = getname(filename);
(gdb) list
803             int fd, error;
804
805     #if BITS_PER_LONG != 32
806             flags |= O_LARGEFILE;
807     #endif
808             tmp = getname(filename);
809             fd = PTR_ERR(tmp);
810             if (!IS_ERR(tmp)) {
811                     fd = get_unused_fd();
812                     if (fd >= 0) {
(gdb) next
...
811                     fd = get_unused_fd();
(gdb) next
812                     if (fd >= 0) {
(gdb) print fd
$2 = 0
(gdb) cont
Continuing.
```

FIGURE 2.10 – Exemple de débogage avec UML .

LKCD : Linux Kernel Crash Dump

Permet de

- Sauvegarder l'état du noyau en cas de crash.
- Pilotable aussi par séquences SysRq.
- Image mémoire sauvegardée en swap, puis lors du reboot automatiquement dans/var/log/dumps.
- Analyse post-mortem grâce a l'outil lcrash
- Pas de surcharge, utilisable en production.
- Sauvegarde de l'etat, pas des événements

Exemple :

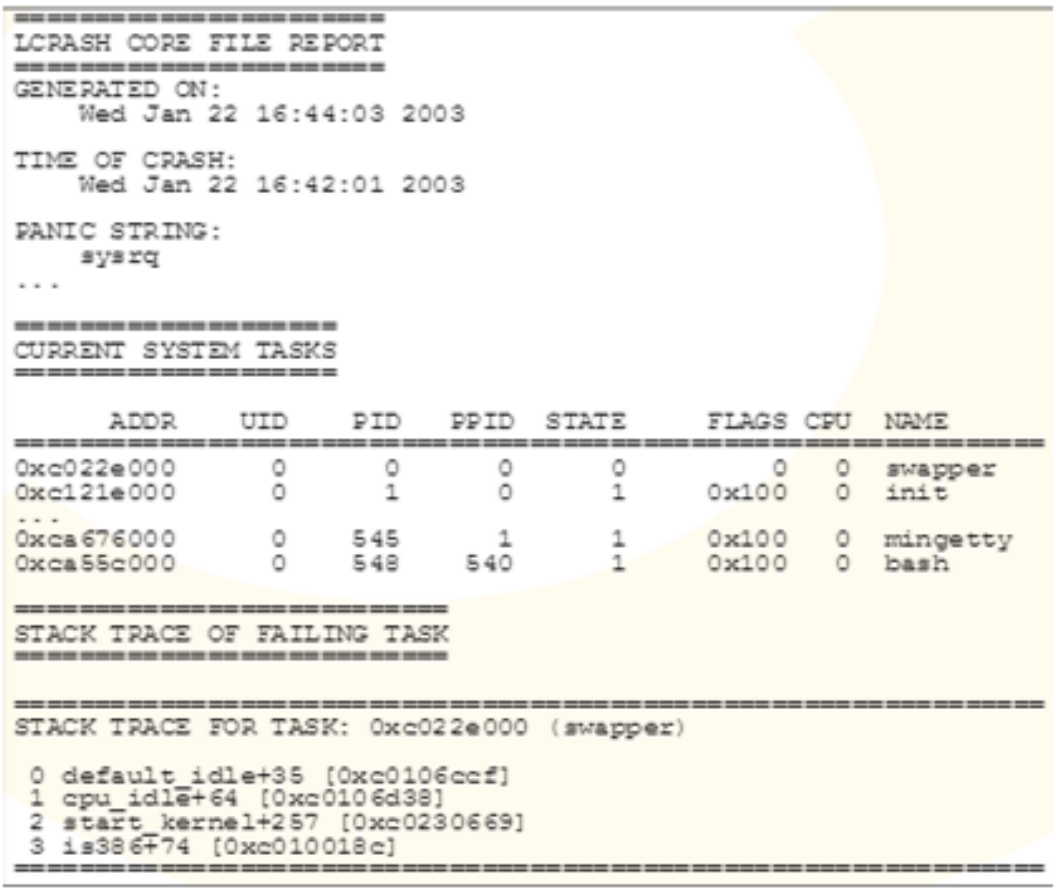

FIGURE 2.11 – Exemple de débogage avec LKCD .

2.11 Les opérations de compilation

2.11.1 La première compilation d'un noyau : (sans utiliser UML)

La première étape : :

1. téléchargement et installation des paquets nécessaires pour commencer la compilation
apt-get install build-essential
Le paquet build-essential contient des références à de nombreux paquets nécessaires pour la construction de logiciels en général Pour bénéficier d'une interface de configuration on va installer des librairies supplémentaires
sudo apt-get install libncurses-dev (Dépendances pour l'interface menucon-

fig)
sudo apt-get install libncurses5-dev (Dépendances pour l'interface nconfig)
sudo apt-get install libqt3-mt-dev (Dépendances pour l'interface xconfig)
Puis l'instalation de
sudo apt-get install kernel-package
sudo apt-get install gcc

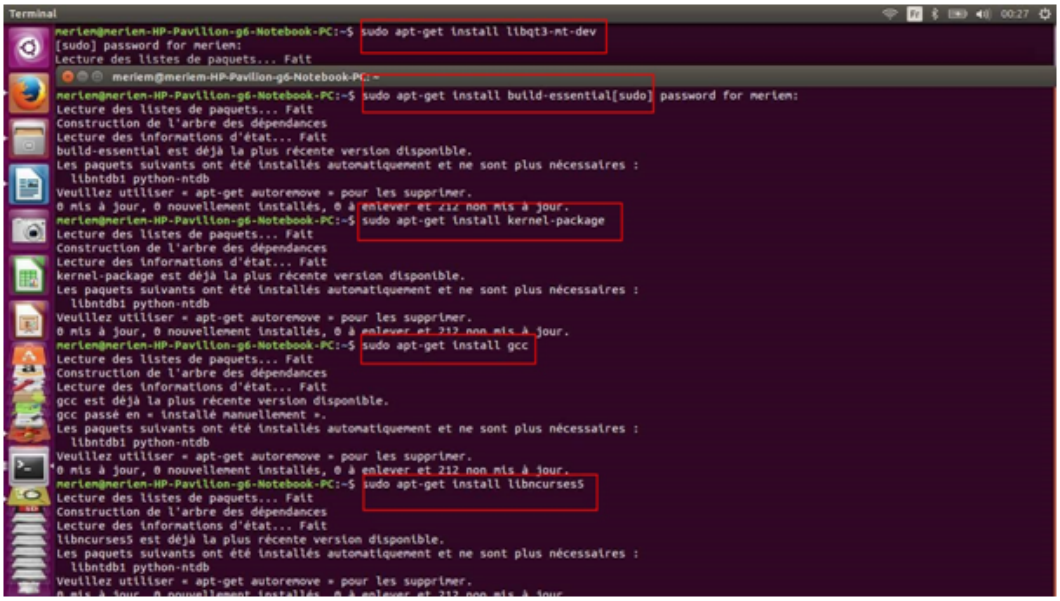

FIGURE 2.12 – installation de différents paquets nécessaire pour la compilation .

Deuxième étape :

2. Installation des sources :

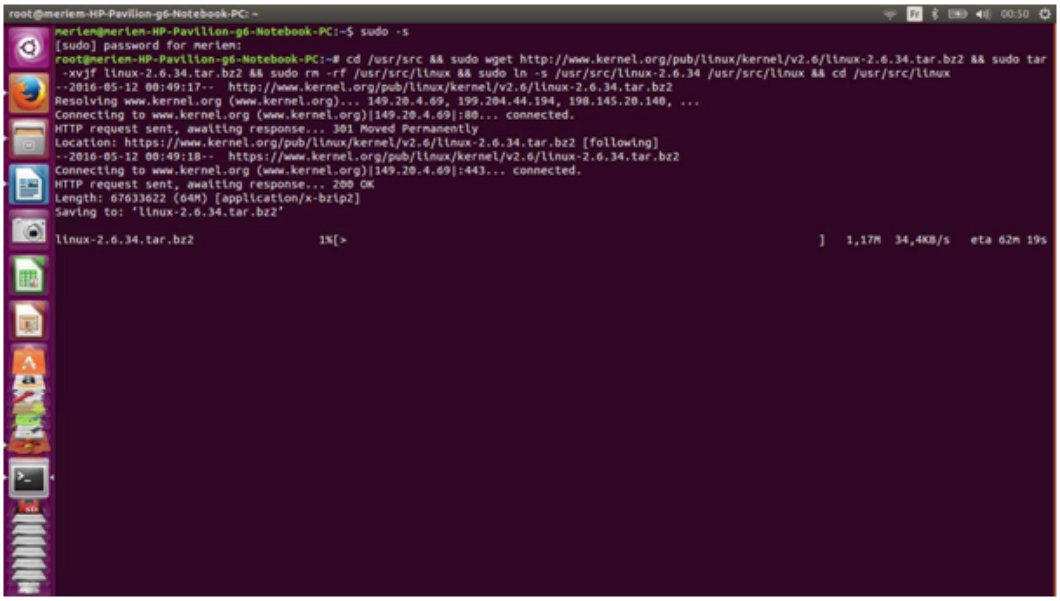

FIGURE 2.13 – Lancement d' installation des sources .

Après lancement de l'installation un crash est apparus sur les terminales avec une écriture illisible et infinie provoquant le blocage d'ordinateur donc je fus dans l'obligation de le redémarrer depuis la touche .

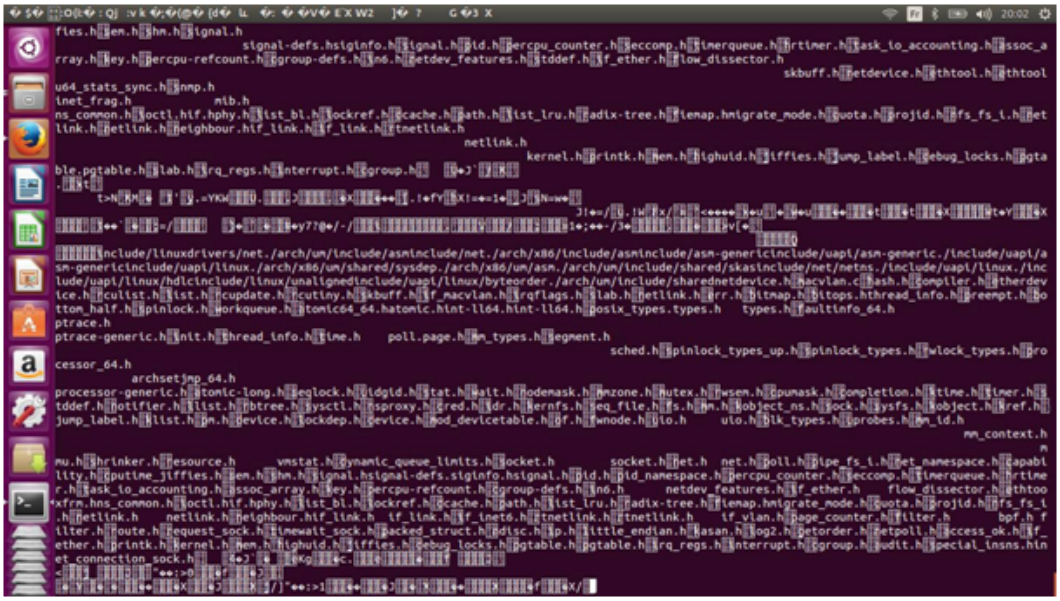

FIGURE 2.14 – Le crash .

FIGURE 2.15 – Le crash .

Le crash mit impossible d'accéder a la session malgré plusieurs tentatives

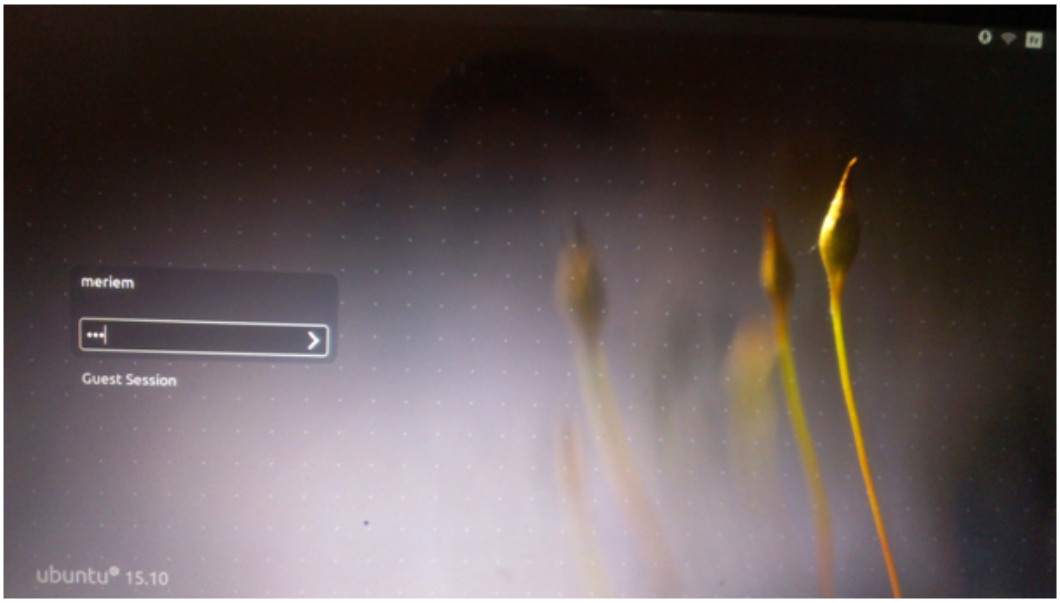

FIGURE 2.16 – Session inaccessible n .

Discussion :

Apres l analyse de cause il s'avère qu'il ya deux erreurs :
La première erreur c'est la compilation de noyau en tant que super utilisateur, cette façon est déconseillée en compilation.
La deuxième erreur est la cause qui a empêcher le système pour bien fonctionner lorsque la dernière manipulation a été testée avec le noyau 2.6.34 et notre version du noyau 2.4.27 ignorant que cette manipulation faisant un risque au système, donc on peut dire que la compilation de noyau demande beaucoup d'attention et pour le faire il faut lire beaucoup de documentation.

2.11.2 La deuxième compilation du noyau utilisant UML :

On doit suivre les étapes suivantes :

1. Utiliser la commande make mrproper pour supprimer toutes les compilations précédentes Dans le readame fourni avec le noyau on trouve que il est conseille de faire make mrproper avant make menuconfig pour avoir un arbre de compilation tout propre

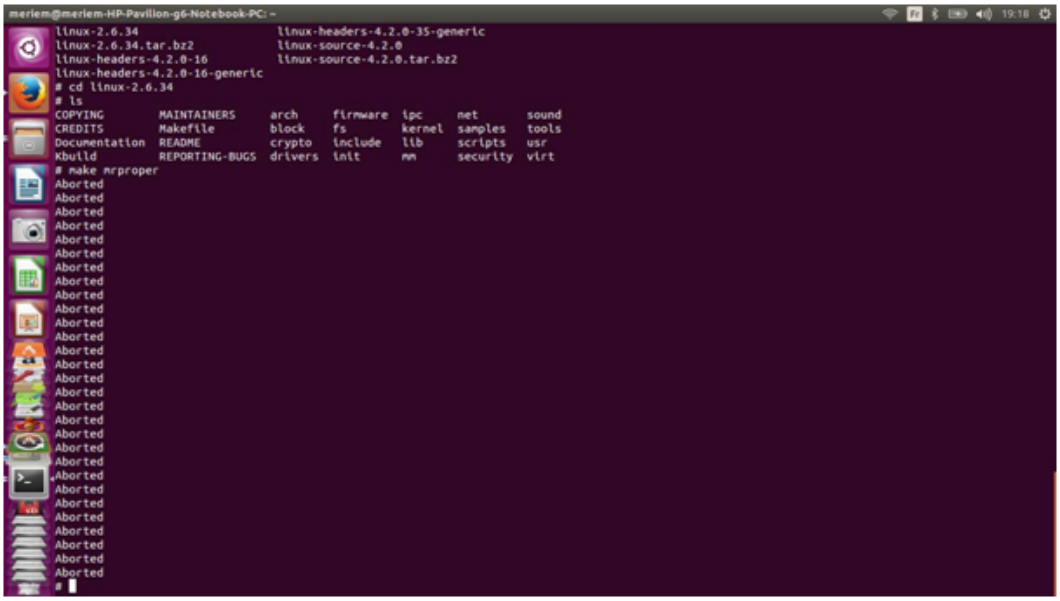

FIGURE 2.17 – éliminer l'ancienne configuration .

2. Voila le lancement de cette opération

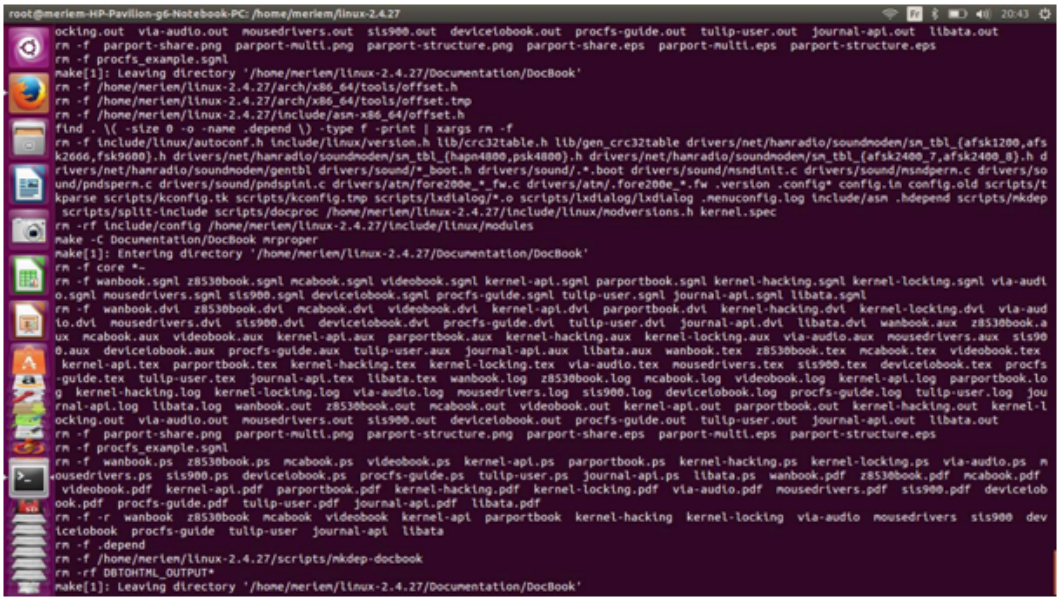

FIGURE 2.18 – lancement d'opération de nettoyage .

3. Premièrement il faut lancer la commande make defconfig pour crée une configuration basée sur les valeurs par défaut de l'architecture de l'ordinateur Cette étape fournie un bon point de départ pour un noyau jamais configuré

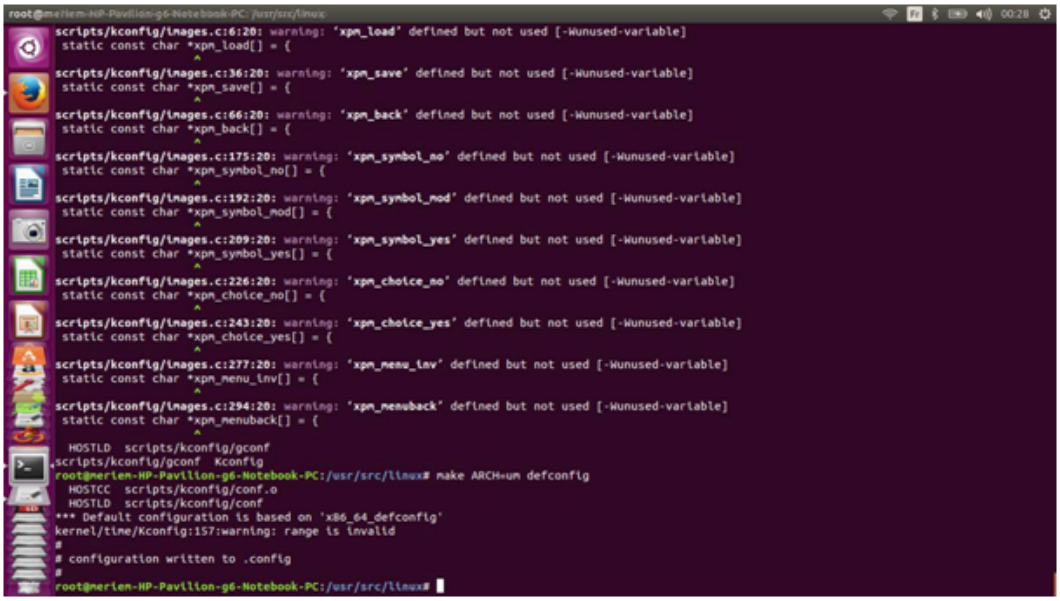

FIGURE 2.19 – Rendre la configuration par defaut .

4. âpres on va revernir pour le choix d'option de configuration et pour cela on va exécuter la commande make menuconfig ARCH=um qui permet de naviguer dans l'interface et activer les options qui conviennent avec le matériel
Le choix des options de configuration couvrent un ensemble de fonctionnalités très large ,la description profonde des options est inutile car on ne peut pas utiliser tous les gestionnaires de peripherique et toutes les fonctionnalités de linux avec un meme ordinateur , Il s'agit donc de répondre aux questions appropriées selon la configuration
la moindre erreur de configuration peut endomager le système .

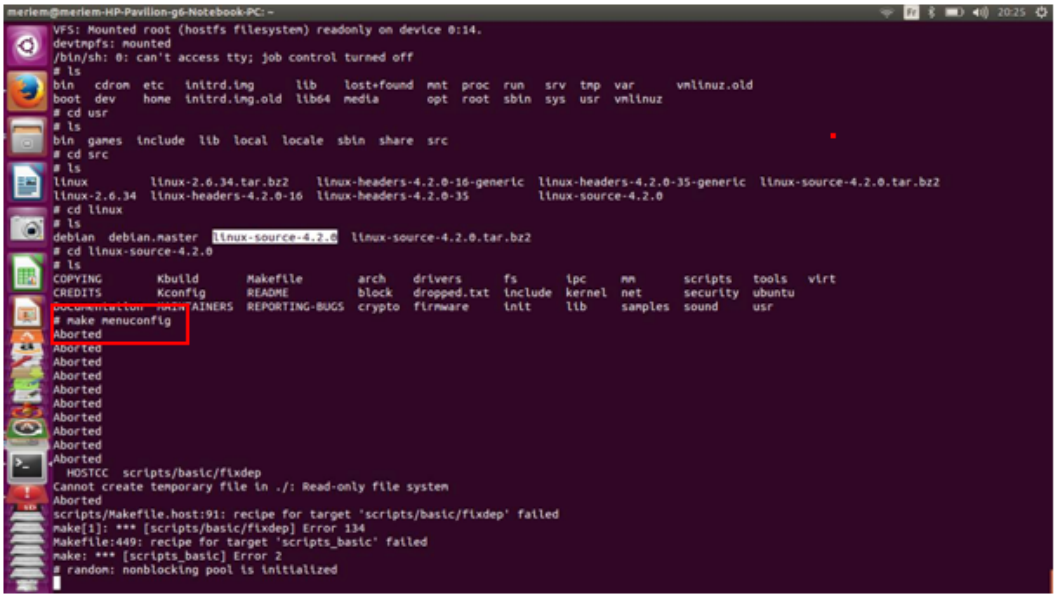

FIGURE 2.20 – Lancement d'interface de configuration .

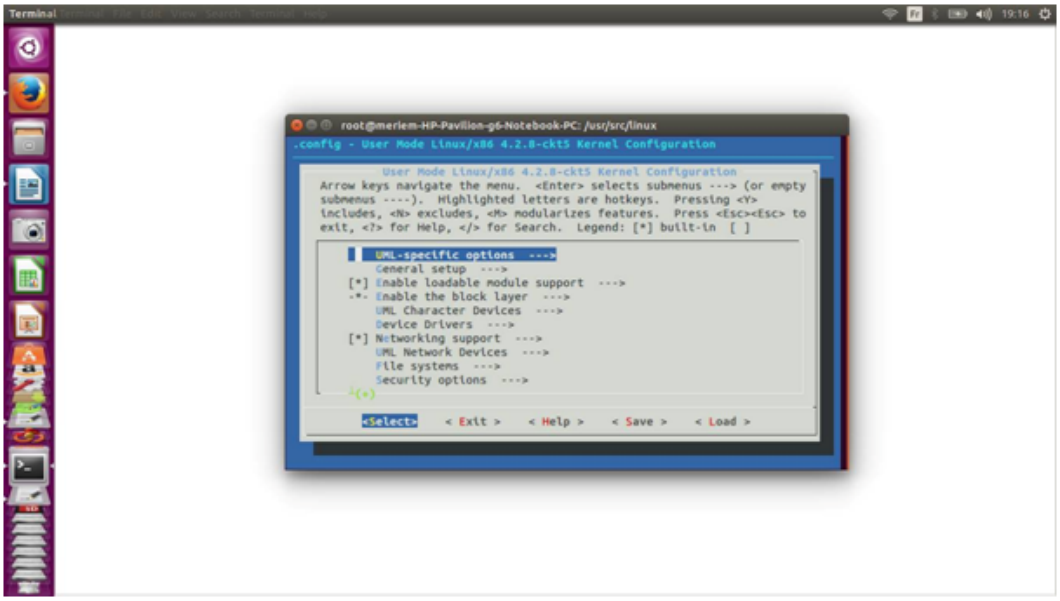

FIGURE 2.21 – l'interface menuconfig .

5. On va sélectionner Géneral setups utilisant Les touches fléchées on déplace dans le menu vers l'option « Initial RAM filesystem and RAM disk

(initramfs/initrd) support » pour l activern
Cette option permet d'activer la gestion des systèmes de fichiers virtuels en mémoire que les gestionnaires de chargement peuvent utiliser pour charger le système de fichiers racine lors du démarrage sans disque. Elle a une dépendance avec la compilation de noyau et si on désactive UML la compilation sera échouée

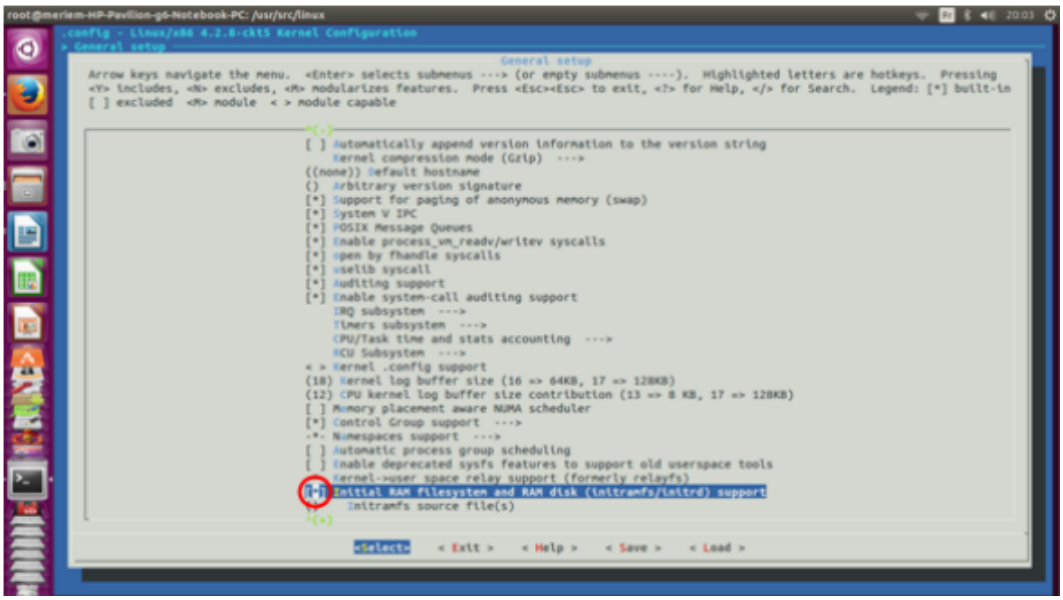

FIGURE 2.22 – l'option « Initial RAM filesystem and RAM disk (initramfs/initrd) support » est activée .

6. ensuite on passe a l'option kenel hacking et on active l'option « Kernel debugging »
Cette option permet d'accéder à une série d'options relatives au débogage du noyau. Ces options ne sont pas réellement utilise pour un utilisateur et ne sont présentes que pour les développeurs de gestionnaires de périphériques ou les développeurs du noyau

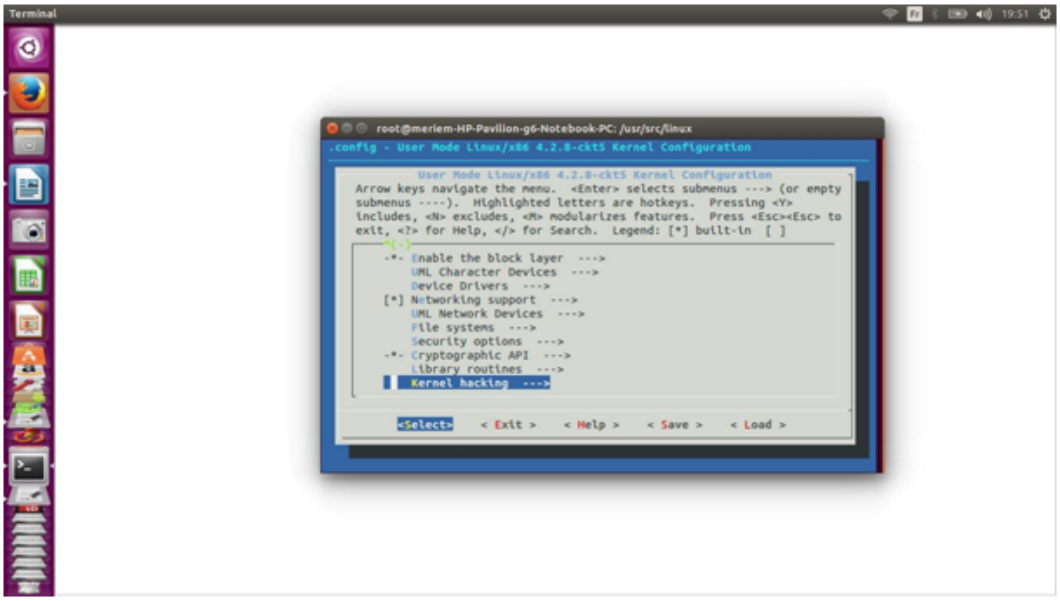

FIGURE 2.23 – Vérification d'existence « kenel hacking » .

7. la construction du noyau : Une fois la configuration du noyau réalisée, la compilation peut être lancée. Pour cela, il suffit d'executé la commande make ARCH=um 2>1 | tee build.out dans le répertoire /usr/src/linux.

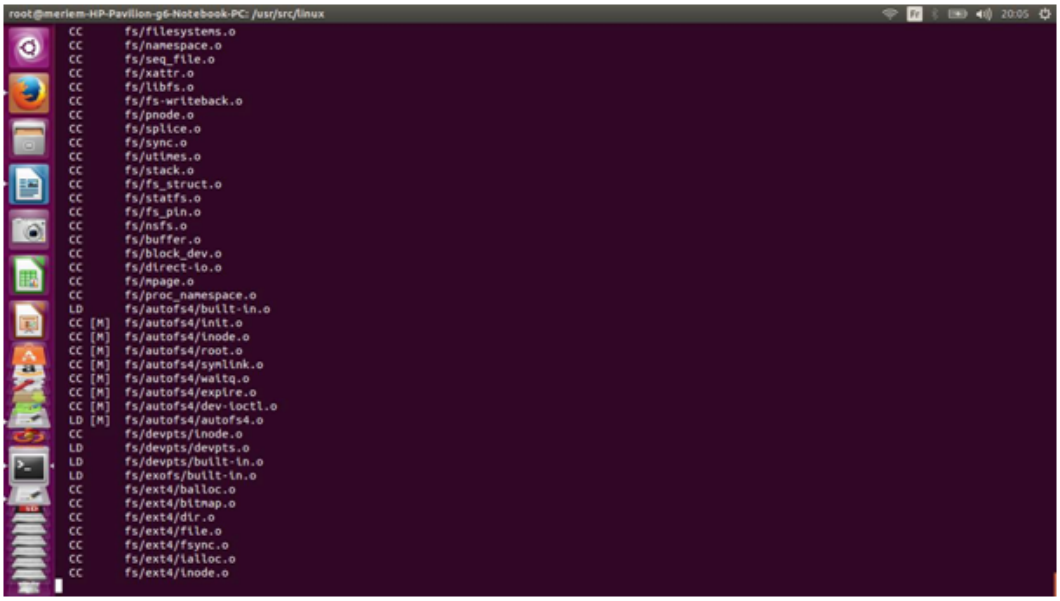

FIGURE 2.24 – lancement de compilation du noyau .

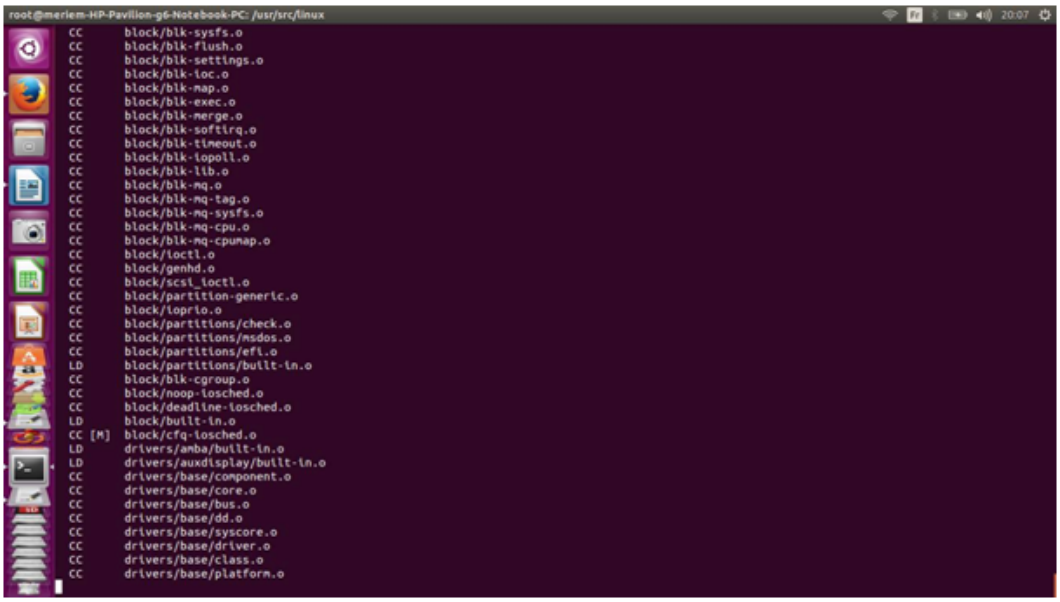

FIGURE 2.25 – lancement de compilation du noyau .

cette opération prendre avec nous

FIGURE 2.26 – Le temps consacré par la compilation .

8. Vérifier si on a les fichiers linux vmlinux : le noyau est installé dans le répertoire /usr/src/linux Il porte souvent le nom de vmlinux

```
  LD [M]  fs/isofs/isofs.ko
root@meriem-HP-Pavilion-g6-Notebook-PC:/usr/src/linux# ls -l linux vmlinux
-rwxr-xr-x 2 root root 40767392 mai   14 20:09 linux
-rwxr-xr-x 2 root root 40767392 mai   14 20:09 vmlinux
root@meriem-HP-Pavilion-g6-Notebook-PC:/usr/src/linux# file
```

FIGURE 2.27 – Vérification de fichier vmlinux .

On va lancer UML :

```
root@meriem-HP-Pavilion-g6-Notebook-PC:/usr/src/linux-source-4.2.6# ./linux ubd0=~/Debian-Squeeze-AMD64-root_fs
```

FIGURE 2.28 – Lancement de machine UML.

On va Passer au Free terminal pour voir ce qu'on a comme processus :

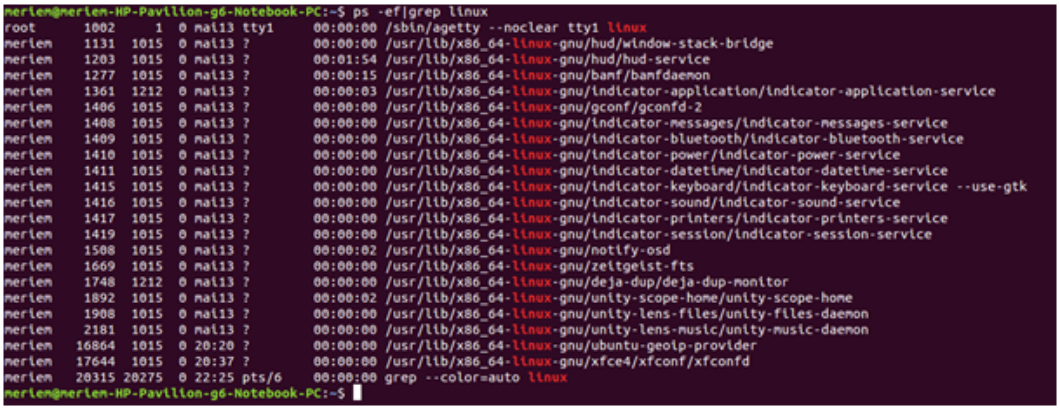

FIGURE 2.29 – Débogage de noyau utilisant gdb et Uml .

9. Débogage de noyau utilisant gdb et Uml
Executé la commande Gdb linux ;
Ensuite la commande List sys$_{clonepourcreunbreakpoint(bouclefork)}$;

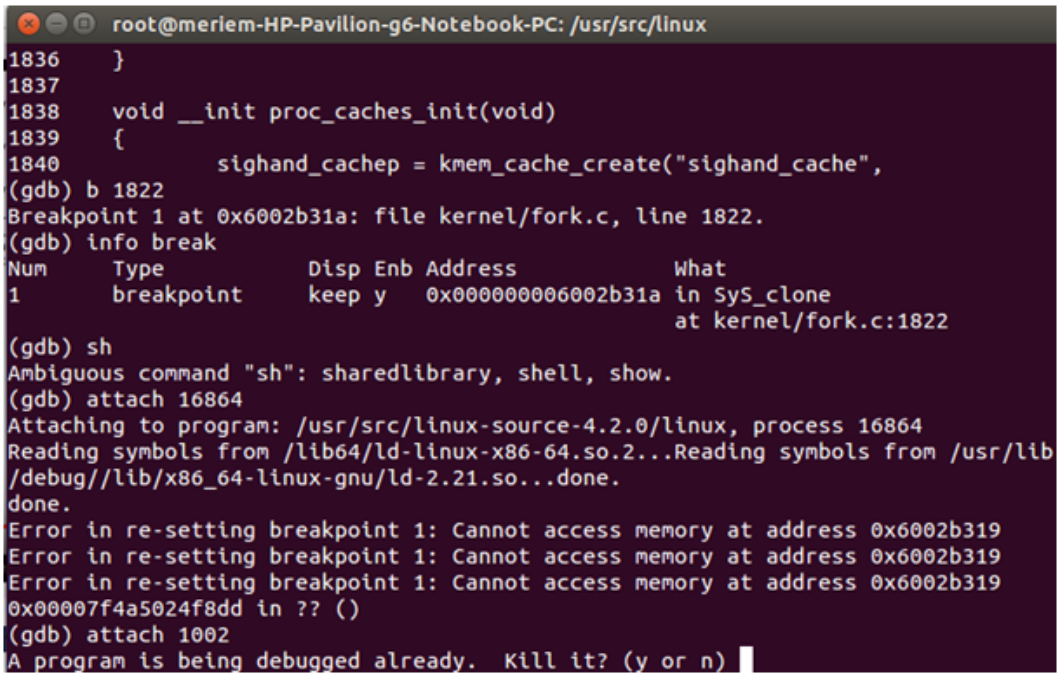

FIGURE 2.30 – Execution UML sous GDB et création de break point .

2.11.3 Résultat :

Le résultat de cette compilation est un nouveau noyau suivie par la date de compilation et d'autre information et aussi si on a observé le processus de démarrage on trouve qu'il ya pas d'erreurs

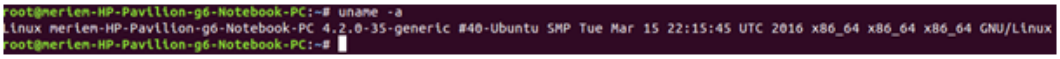

FIGURE 2.31 – Version de noyau noyau .

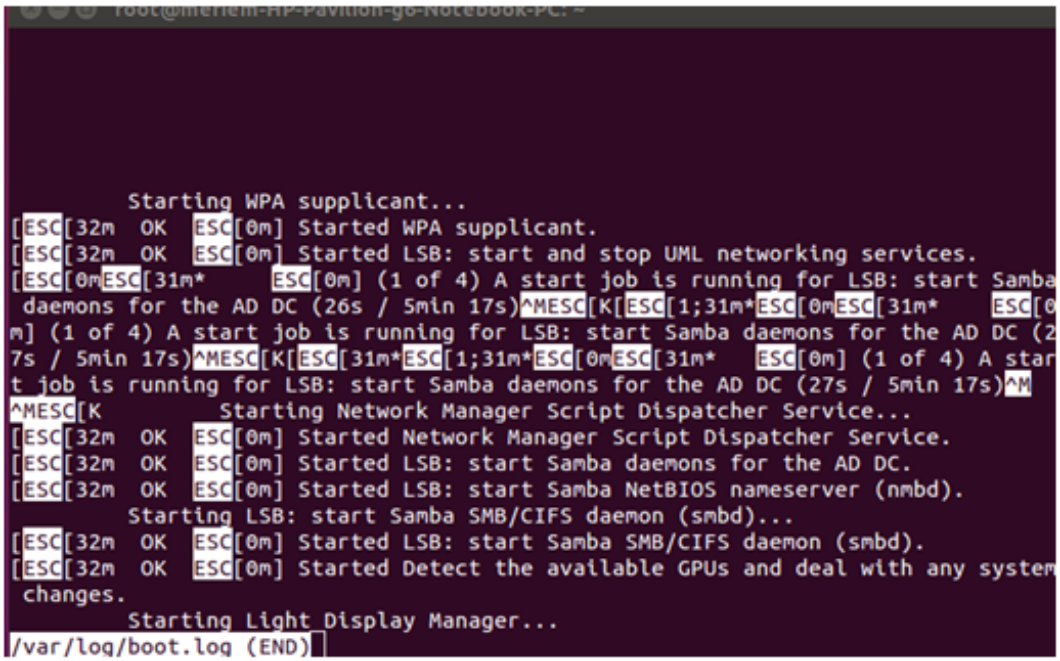

FIGURE 2.32 – Ouverture de fichier log .

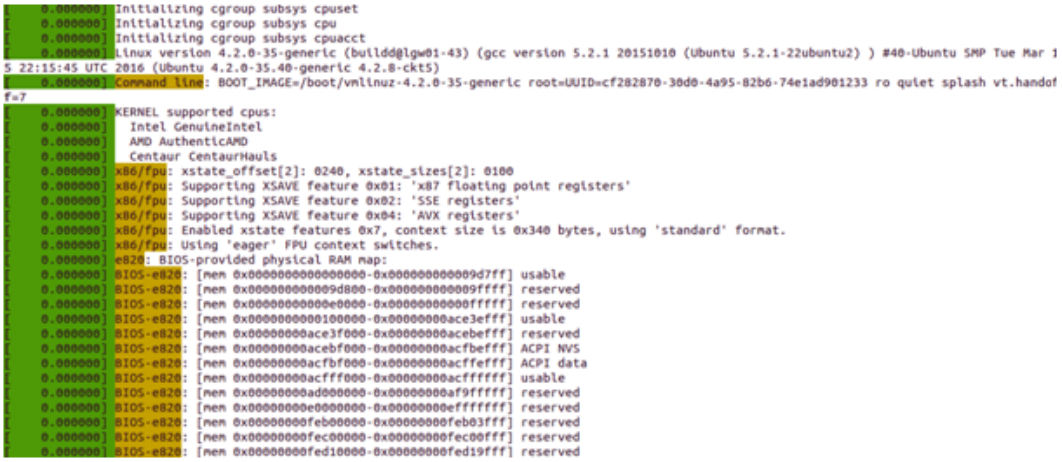

FIGURE 2.33 – Disparition des erreurs après la compilation .

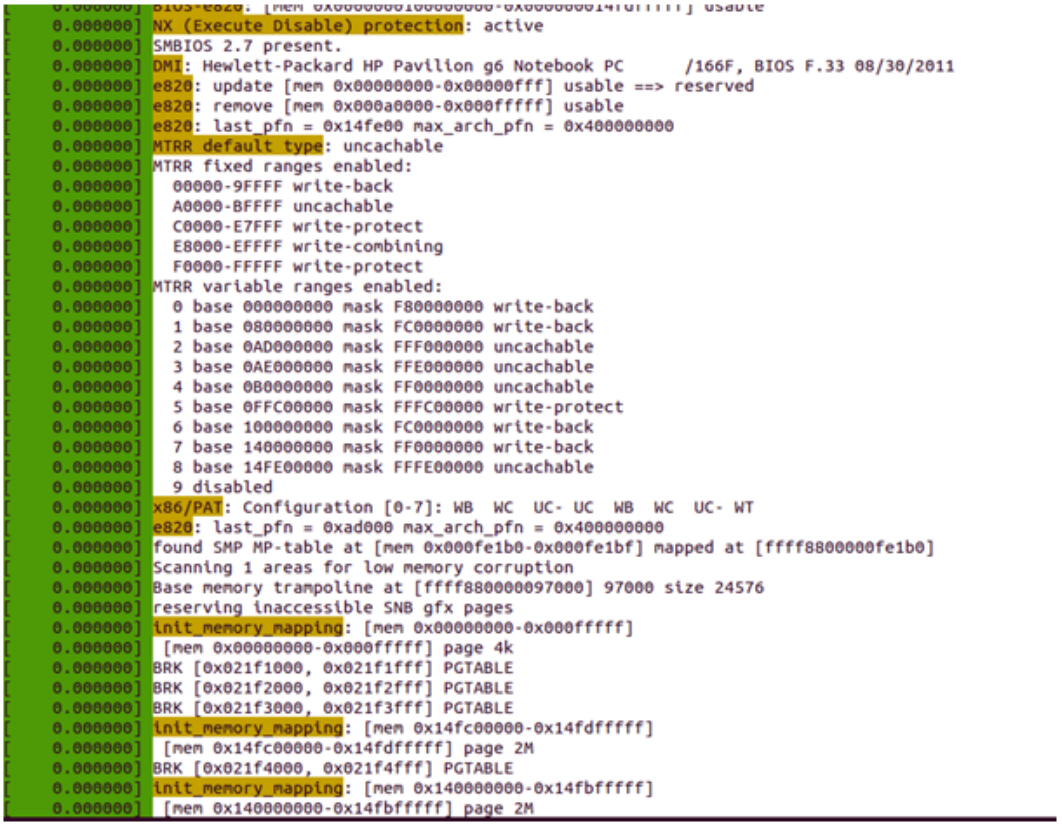

FIGURE 2.34 – Un ficher log sans erreur .

2.12 Conclusion

Les commandes dangereuses reste dangereuse mais le but c'est protéger la machine, Avec UML on peut atteindre cette protection La compilation du noyau n'est une tâche facile. Il permet de répondre correctement aux questions de configuration. L'erreur ici n'est pas autorisée.

Chapitre 3

Création d'un laboratoire virtuel

3.1 Introduction

Une des objectifs de notre étude est la réalisation d'un laboratoire virtuel avec UML. Ce point est très important, car de nombreux simulateurs et émulateurs (Netkit et Vnuml par exemple) travaillent avec le principe de laboratoires virtuels .
Dans ce chapitre, on va présenter deux manières différentes pour créer un laboratoire : La première lorsque tous le laboratoire existe dans la même machine (cas utilisé par les simulateurs et émulateurs).
Le deuxième cas lorsqu'une machine UML est connecté a' un réseaux réel.

3.2 Creation d'un laboratoire avec user mode linux dans une seul machine

On va expliquer dans ce qui suit les étapes à suivre pour réaliser un réseau virtuel dans une machine Après avoir lancé UML, on suit les étapes suivantes :
Etape 1 :
Configuration user mode linux
Si la machine uml n'est pas liée avec un kernel spécial mais elle utilise le kernel de system on utilise /bin/sh comme init pour montrer les informations de noyau On normaliser un peu environnement Par de modification du nom d'hôte de route de urs est lui nommé R1, on va monter les fichiers de system pour les rendre accessible et avec la commande tmpfs on va securisés l'espace allouer au moment du montage.

```
# hostname -b R1
# export TERM=xterm
# export PATH=/usr/local/bin:/usr/bin:/bin:/sbin:/usr/local/sbin:/usr/sbin
# mount -t proc proc /proc
# mount -t sysfs sysfs /sys
# mount -t tmpfs tmpfs /var/run -o rw,nosuid,nodev
# mount -t tmpfs tmpfs /var/log -o rw,nosuid,nodev
```

FIGURE 3.1 – Normaliser l'environnement user mode linux .

Le répertoire proc : est un répertoire utilisé par le noyau pour envoyer des informations aux différents processus d'où le nom /proc [18].
Le répertoire sys : est un répertoire virtuel fourni par le noyau Linux qui exporte des informations sur les différents sous-systèmes du noyau, des dispositifs matériels, et les pilotes de périphériques associés à partir du modèle de périphérique du noyau vers l'espace utilisateur [19]. On monte également /lib/modules permet de placer les modules et /home via hostfs, en lecture/écriture.

```
#  mount -o bind /usr/lib/uml/modules /lib/modules
mount: warning: /lib/modules seems to be mounted read-only.
# mount -t hostfs hostfs /home/home/bernat/mylab -o /home/home/bernat/mylab
#
```

FIGURE 3.2 – Monter le home via hostfs .

A l'intérieur de UML ont à un droits particuliére sur le système La commandes mount et remount nécessitent des privilèges d'utilisateur root pour effectuer des changements des privilèges spécifiques à ces accès
On démarré le système en init=/bin/sh, getty est un programme qui gère la phase d'entree en session sur un système Unix. Il est indispensable si vous souhaitez vous loguer sur votre machine

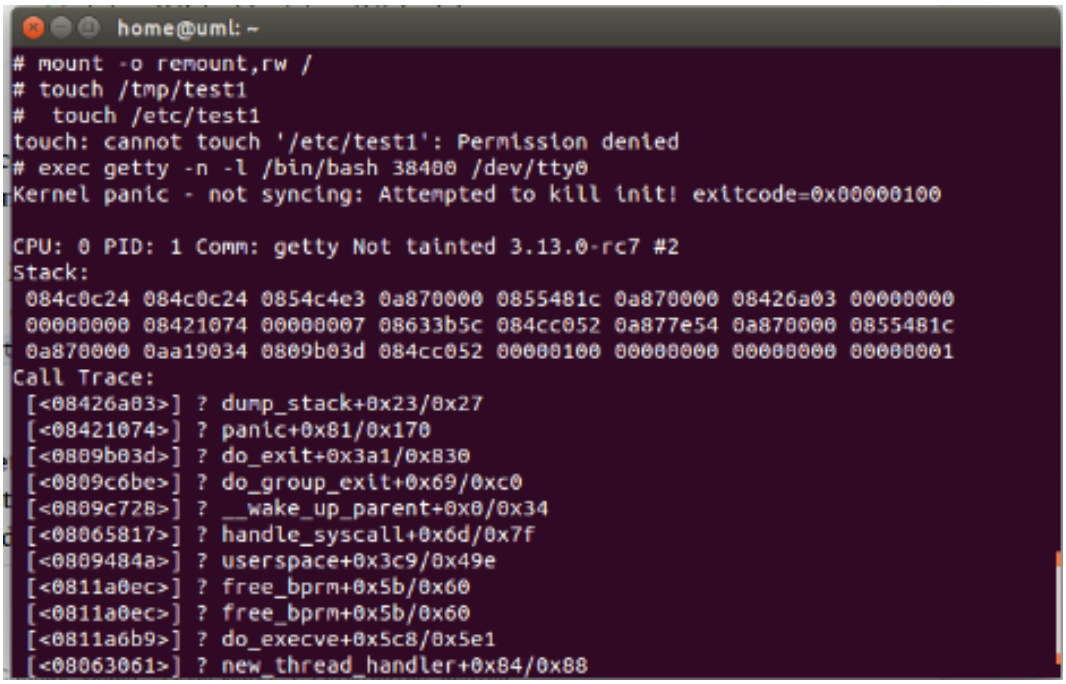

FIGURE 3.3 – Utiliser des priviléges d'accès UML .

Etape 2 : choix d'une interface réseau Après avoir démarée la machine UML, on va ajouter des interfaces réseau virtuelle comme les interfaces TAP ou les switchs VDE

Communication par TAP

Elle est utilisée pour injecter des trames ethernet, Tout ce que le noyau envoie dans cette interface est reçu par l'application Le noyau considère cette interface comme une simple interface Ethernet. Donc on va faire de bridger. On va créer deux interfaces et les lier entre elle par un câble virtuel,ensuite on demande les deux machine UML la premier machine UML par la commande :Tap tap-R1 br-R1R2

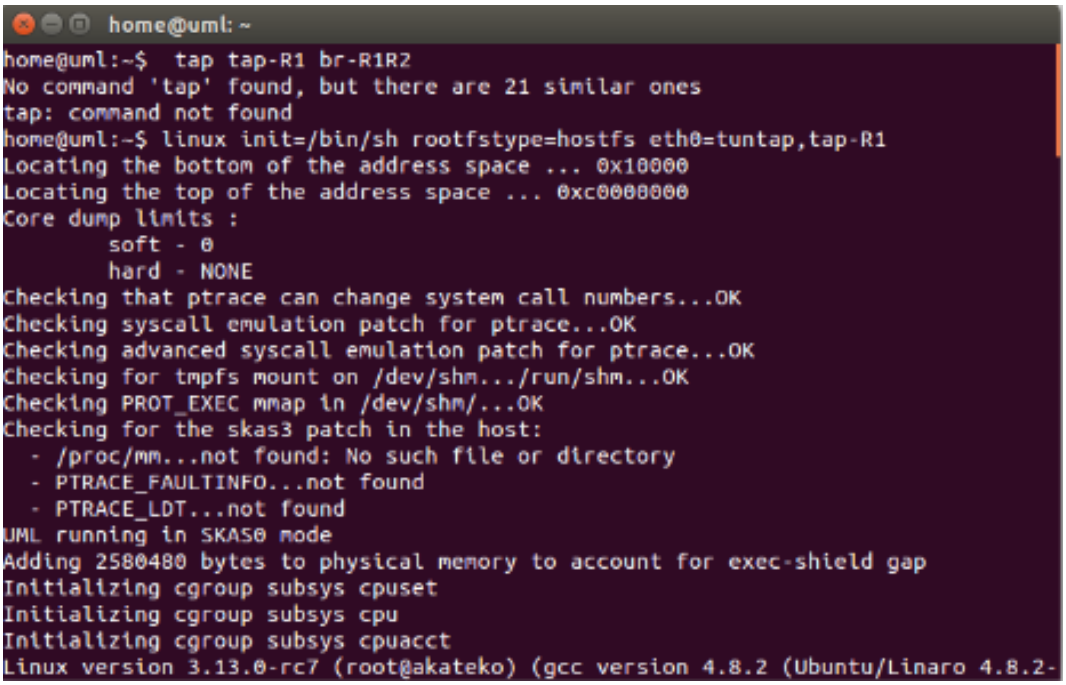

FIGURE 3.4 – Interface tap R1L .

Et la deuxième machine UML par la commande :Tap tap -R2 br-R1R2

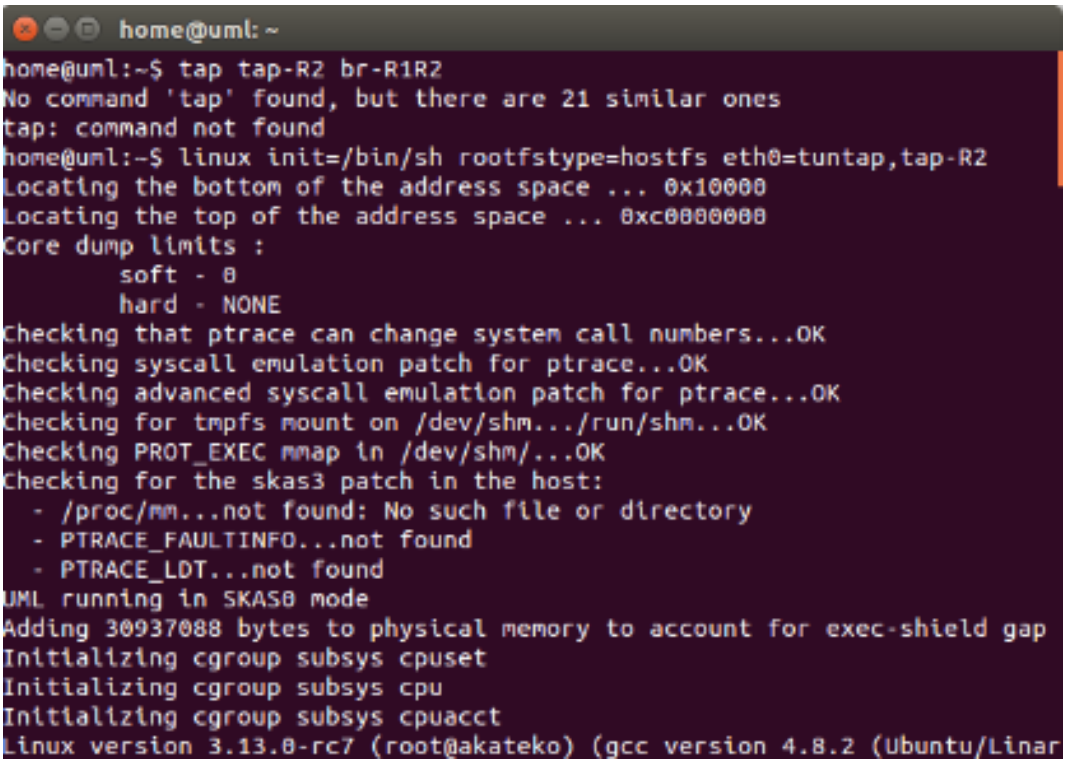

FIGURE 3.5 – Interface tap R2 .

communication avec un switch vde

Le switch vde est un commutateur virtuel fourni avec l'architecture de réseau vde. Switch vde peut interconnecter plusieurs périphériques virtuels de réseaux multiples vde-switches qui peuvent être reliés entre eux avec vde câbles. Pour utiliser un switch vde ,On va installer le paquet vde2. Il est alors possible de lancer le switch avec la commande vde switch.

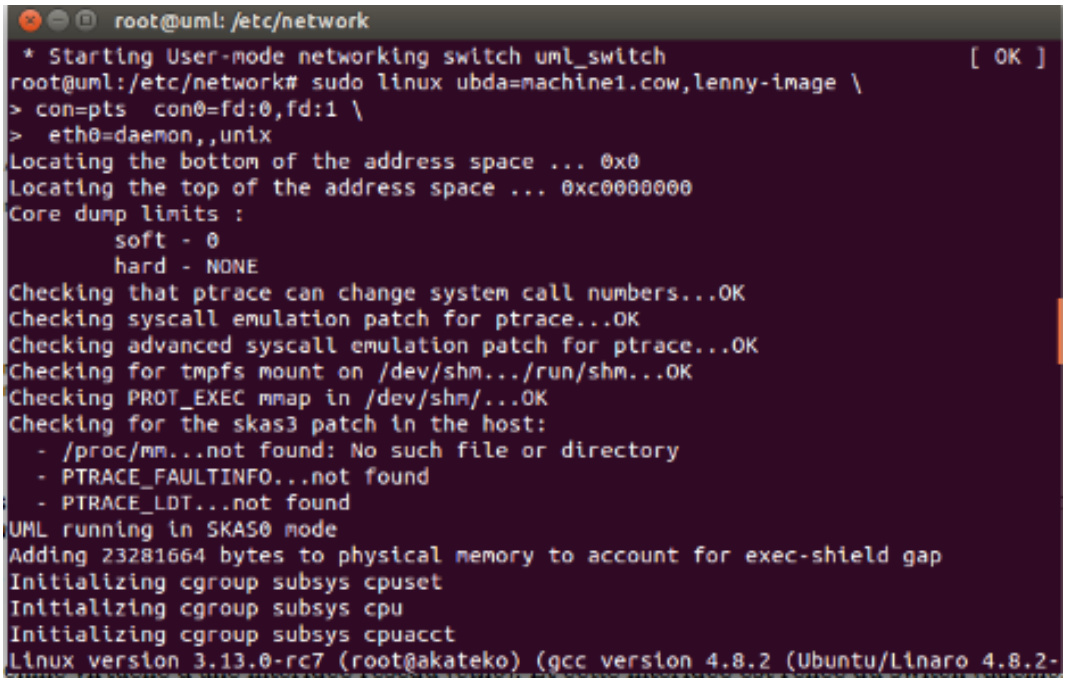

FIGURE 3.6 – lancement de user mode switchs .

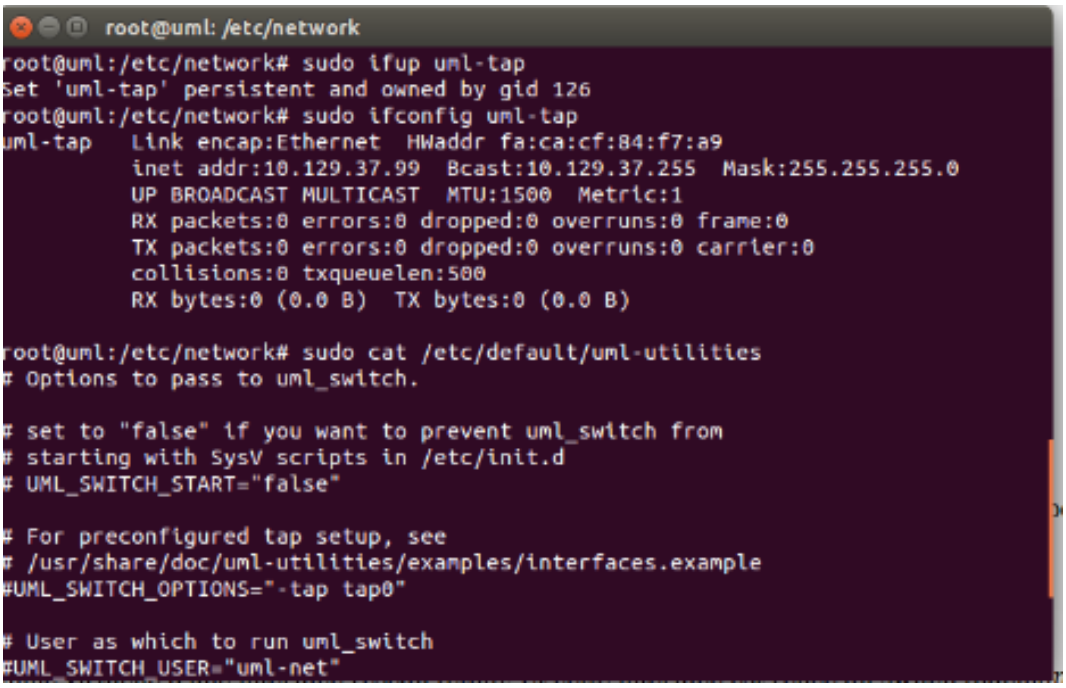

FIGURE 3.7 – Configuration la machine 1.

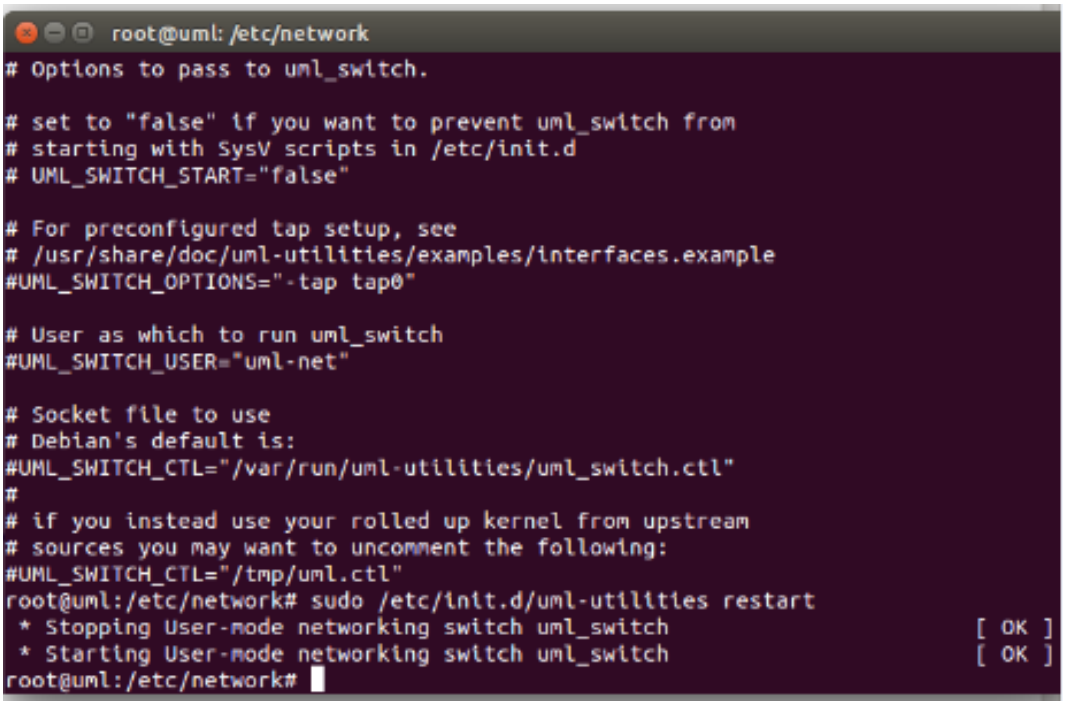

FIGURE 3.8 – Configuration de machine 1 switch .

On va faire communiquer deux UML à travers ce switch. La premiére machine par les commandes :
linux init=/bin/sh rootfstype=hostfs eth0=vde
ip link set up dev eth
ip addr add 192.168.0.1/24 dev eth0

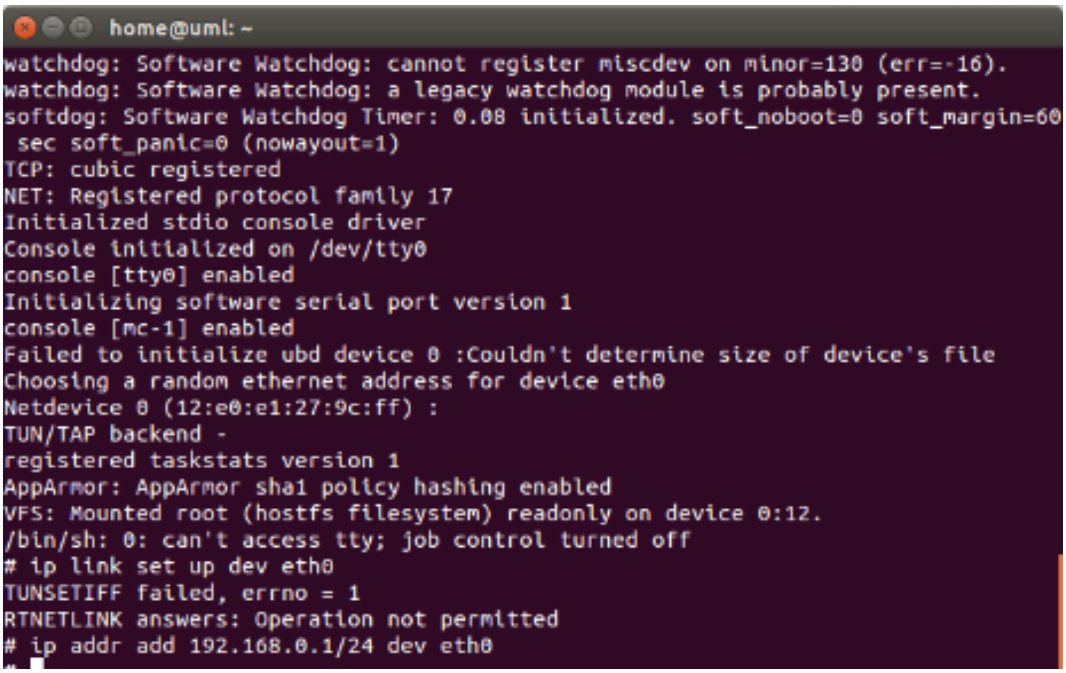

FIGURE 3.9 – Communiquer machine1 UML avec switchh .

Et la deuxiéme machine par les commandes : :
linux init=/bin/sh rootfstype=hostfs eth0=vde
ip link set up dev eth0
ip addr add 192.168.0.2/24 dev eth0
ping 192.168.0.1

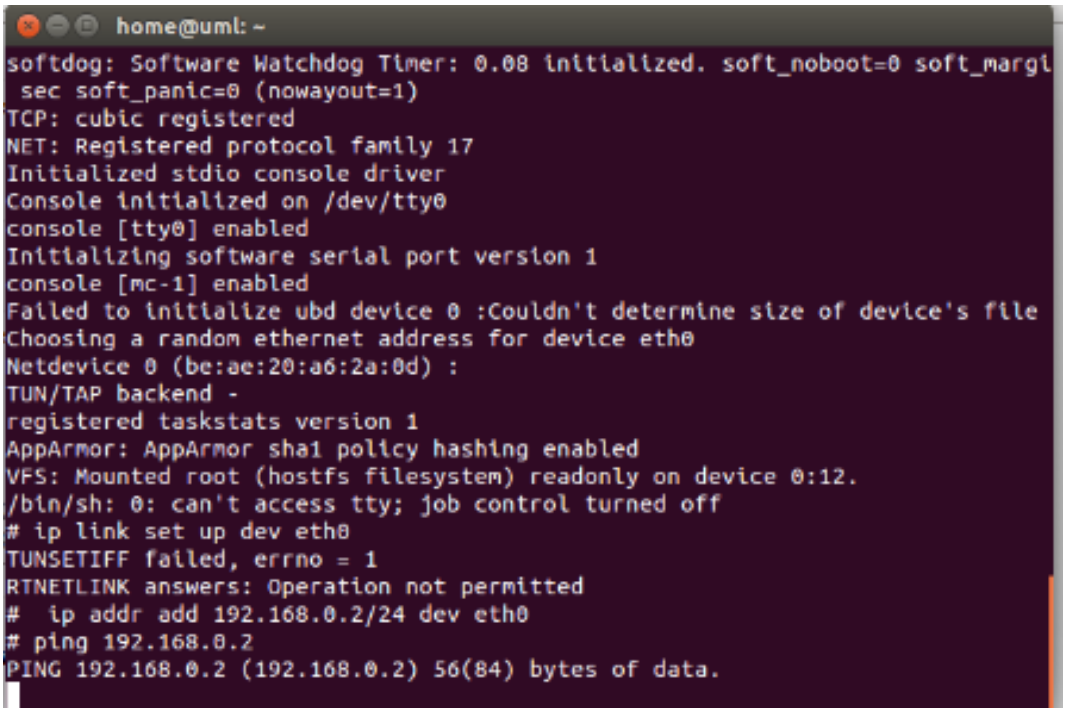

FIGURE 3.10 – Communiquer machine2 UML avec switch .

Etape 3 :

Le laboratoire

Dans le labo nous allons mettre deux sites distants, Chaque site utilise OSPF pour le routage interne. BGP sera utilisé pour échanger les routes entre les deux sites. Nous n'allons utiliser que des switch VDE pour la partie réseau.

3.2.1 Configuration des UML

On va configurer la machine user mode linux à l'aide d'un scripts suivants :

```
Case $$ in
        # Inside UML. Three states :
        TMP=$(mktemp -d)
        trap "rm -rf $TMP" EXIT
        check_dependencies
        setup_screen
        # Setup switches
        setup_switch site1
        setup_switch site101
        setup_switch internet
        # Start VM
Start_vm R1 eth0=vde, $TMP/switch-site1.sock
Start_vm R2 eth0=vde, $TMP/switch-site101.sock
Start_vm V1 eth0=vde, $TMP/switch-site1.sock eth1=vde,$TMP/switch-internet.sock
Start_vm V2 eth0=vde, $TMP/switch-site1.sock eth1=vde, $TMP/switch-internet.sock
        start_vm V3 eth0=vde,$TMP/switch-site101.sock eth1=vde,$TMP/switch-internet.sock
        start_vm V4 eth0=vde,$TMP/switch-site101.sock eth1=vde,$TMP/switch-internet.sock
        start_vm I1 eth0=vde,$TMP/switch-internet.sock
        display_help
        cleanup
        screen -X quit
Esac
```

On va démarrer le lab. Les répertoires au nom des machines contiennent les fichiers de configuration pour Quagga (un démon de routage) et racoon (un démon IKE pour IPsec)
On va activer les démons Quagga correspondant aux protocoles qui on peut configure Zebra et ospf et ripd

Quagga est installé sur un hôte physique et agit comme un routeur dédié. Il met à jour la table de routage du noyau. Il modifie l'interface d'adressage IP, définit les routes statiques et permet le routage dynamique.

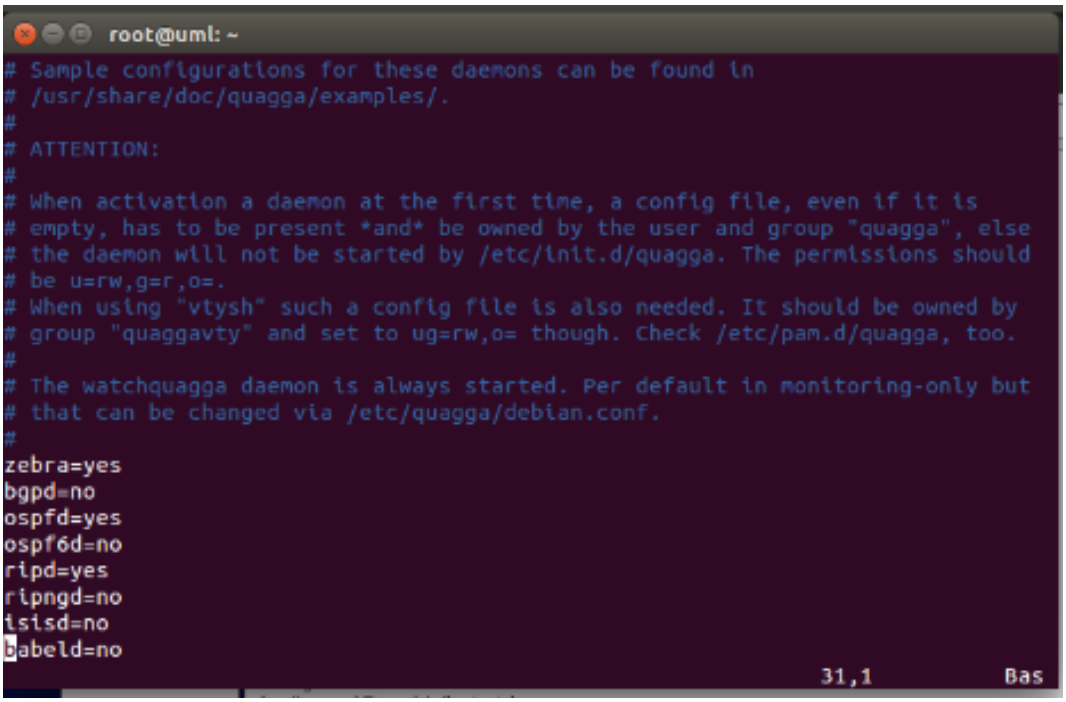

FIGURE 3.11 – Configuration fichier quagga .

Le démon de zèbre est une couche d'abstraction entre le noyau et les processus de routage en cours d'exécution. Chaque protocole de routage a son propre démon spécifique [21]

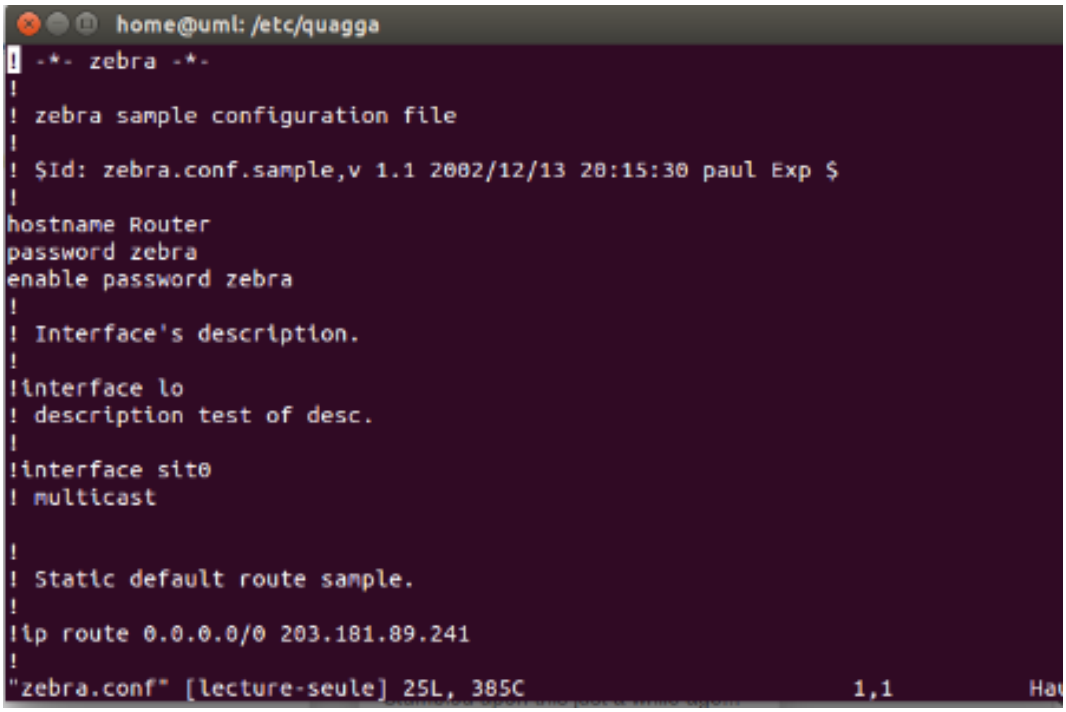

FIGURE 3.12 – Configuration de fichier zebra. .

Le protocole OSPF a été développé en raison d'un besoin dans la communauté Internet d'introduire un protocole IGP interne non propriétaire à fonctionnalités élevées pour la famille de protocoles TCP/IP [22].

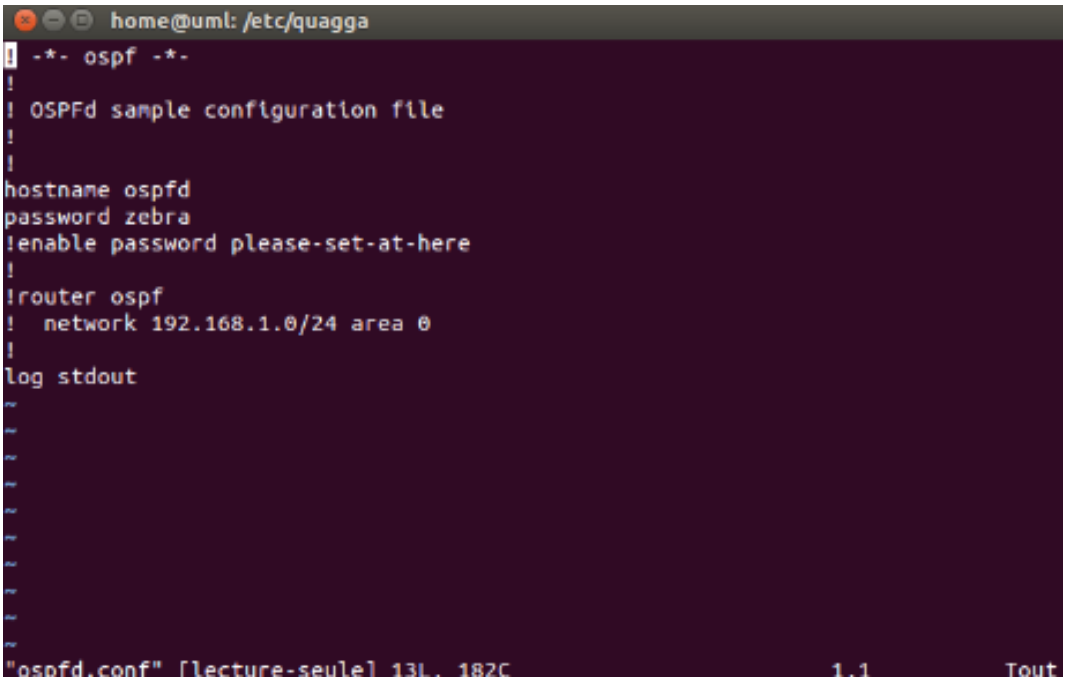

FIGURE 3.13 – Configuration de fichier ospf .

Le Routing Information Protocol (RIP) est l'un des plus anciens protocoles de routage à vecteur de distance qui emploient le nombre de sauts comme métrique de routage. RIP empêche les boucles de routage en mettant en place une limite sur le nombre de sauts autorisés dans un chemin de la source à la destination [23].

Préparation du laboratoire : on suivant les étapes précedentes on va créer un lab avec 6 machines
Pour chaque machine à lancer, on précise quelles sont les interfaces réseau que l'on veut mettre en place. Pour R1 et R2, on utilisera une interface dummy0 pour le réseau interne. Pour la machine V1et V2etV3et V4 On lui destinée à une seule interface.
Pour la machine R1 : en configurer la machine R1

```
echo 192.168.15.1 $HOSTNAME >> /etc/hosts

modprobe dummy

modprobe dummy numdummies=3

ifconfig dummy0 $HOSTNAME

ifconfig dummy 0 :0 192.168.15.1

route add $ HOSTNAME dummy0

route add 192.168.15.1 dummy0 :0

route -n
```

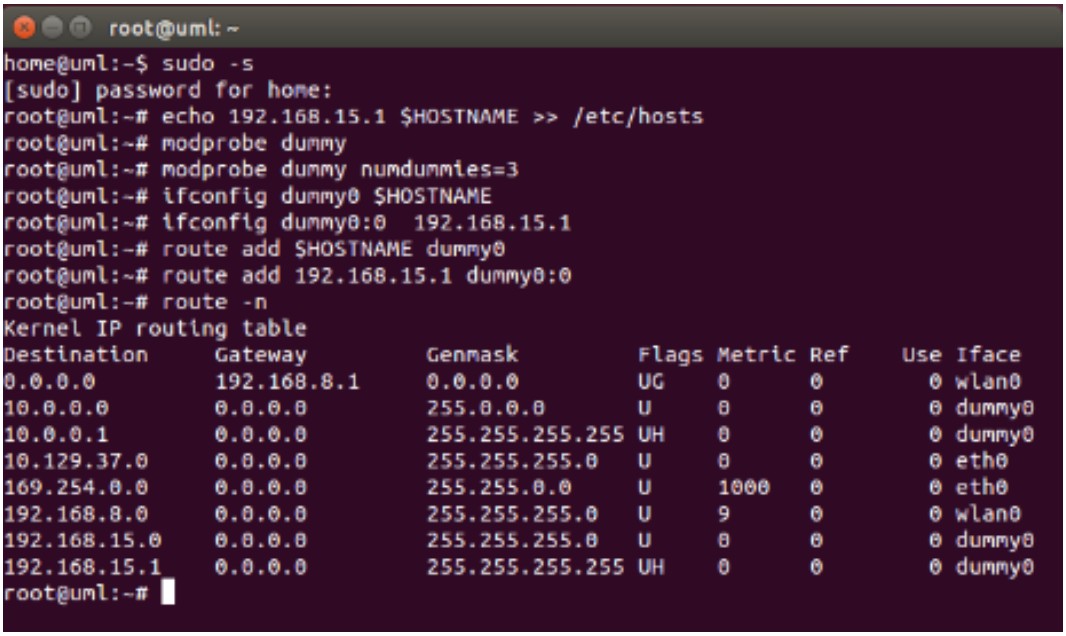

FIGURE 3.14 – Configuration machine R1 .

Pour la machine R2 :

Echo 192.168.115.1 $ HOSTNAME >> /etc/hosts

Modprobe dummy

Ip addr add 192.168.115.1/24 dev dummy0

Route –n

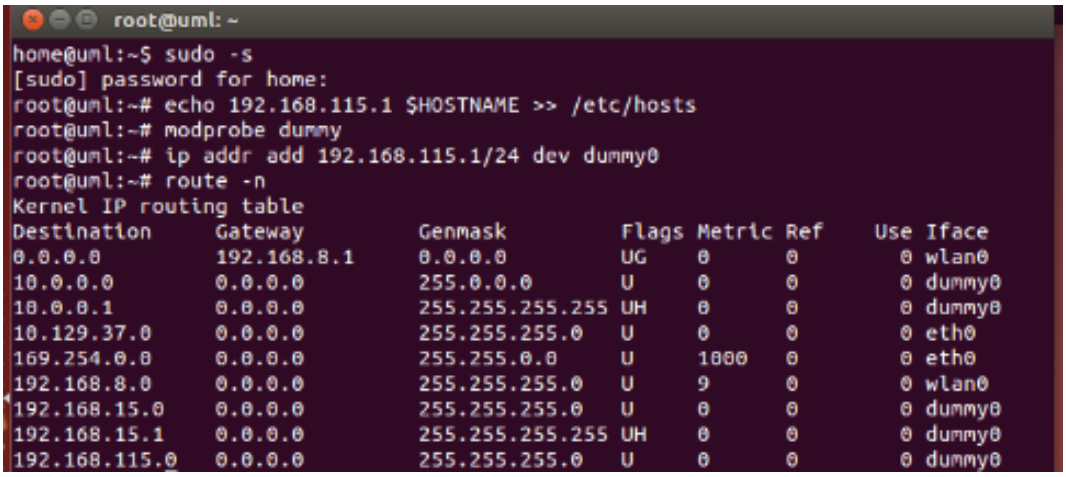

FIGURE 3.15 – Configuration machine R2 .

On utilisera une interface eth pour le réseau interne. On ne demande donc qu'une seule interface pour la machine V1, V2, V3, V4

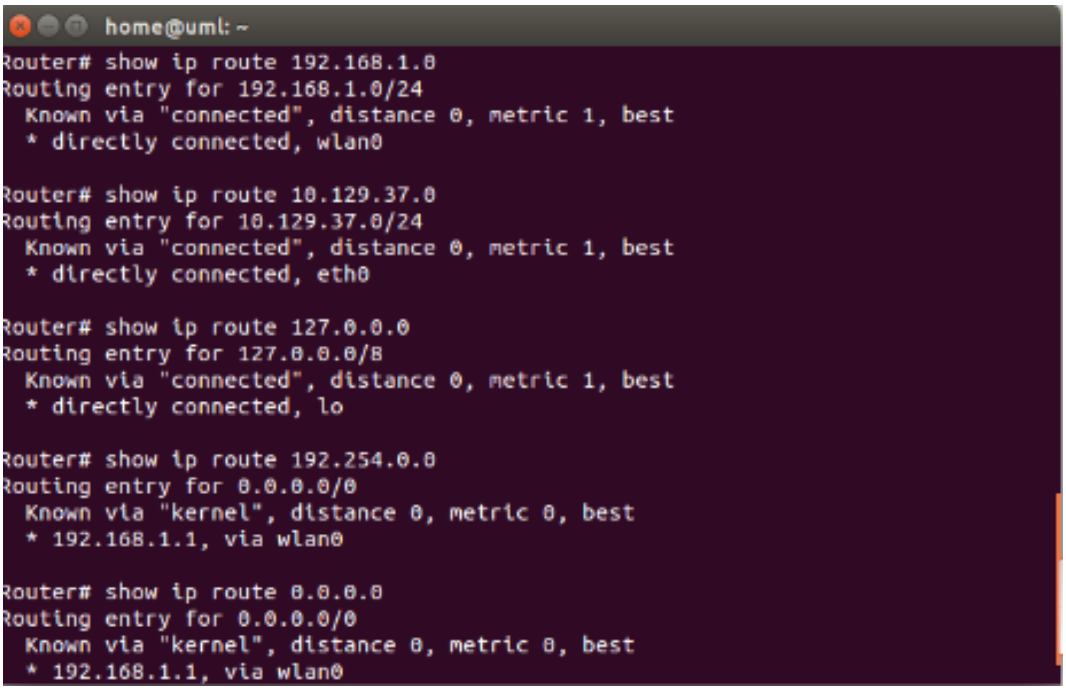

FIGURE 3.16 – Configuration interface ethernet .

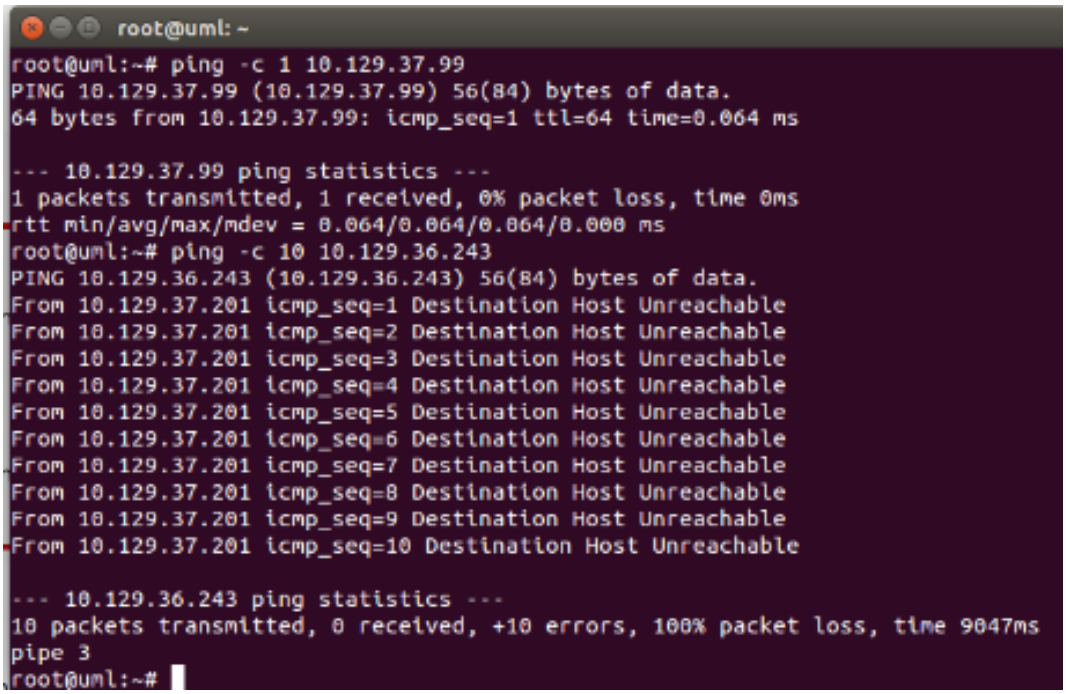

FIGURE 3.17 – Connexion des machine virtuelles..

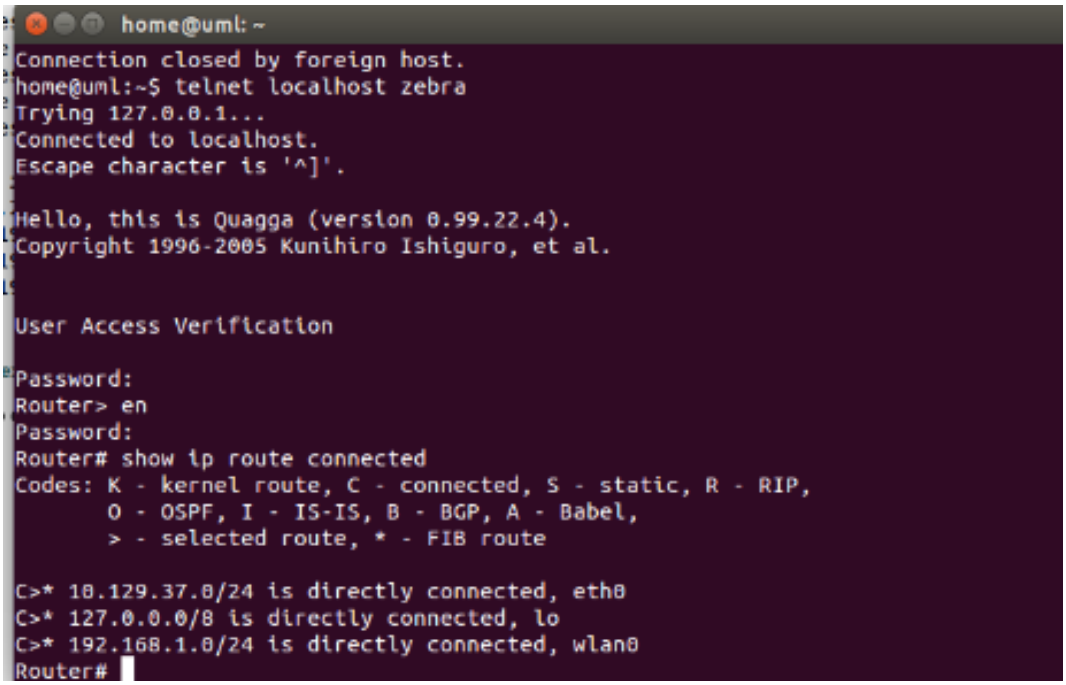

FIGURE 3.18 – Configuration de table de routage R1.

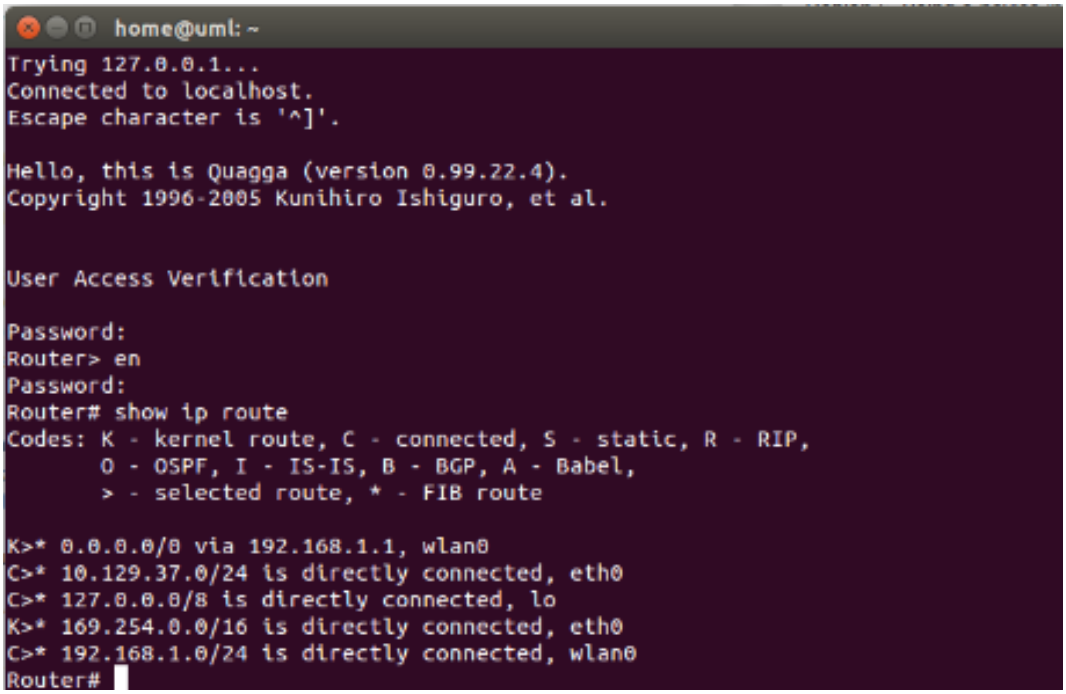

FIGURE 3.19 – Configuration table de routage de R3.

```
<pre><span></span><span class="go">R1# ping -I 192.168.15.1 192.168.115.1</span>
<span class="go">PING 192.168.115.1 (192.168.115.1) from 192.168.15.1 : 56(84) bytes of data.</span>
<span class="go">Warning: time of day goes back (-459129us), taking countermeasures.</span>
<span class="go">64 bytes from 192.168.115.1: icmp_req=1 ttl=62 time=1.10 ms</span>
</pre>
```

FIGURE 3.20 – Pinge de R2 depuis R1.

3.3 Creation d'une laboratoire virtuel avec des machines distinctes

Samba est un logiciel libre il est utilisé pour le partage de ressources (fichiers, imprimantes ...) à travers le réseau entre les postes Linux., nous allons voir comment installer et configurer le logiciel Samba.

3.3.1 Configuration du PC1 :

On va installez Samba ainsi que le client smbclient et le monteur de dossier smbfs et on Sauvegarde le fichier smb.conf en smb.conf.old afin de protéger le fichier original on va Créez le fichier de partage avec leafpad Ce fichier se décompose en deux parties : la section générale appelée "global" et la section de partage s'appelle : DypDesk-Home.

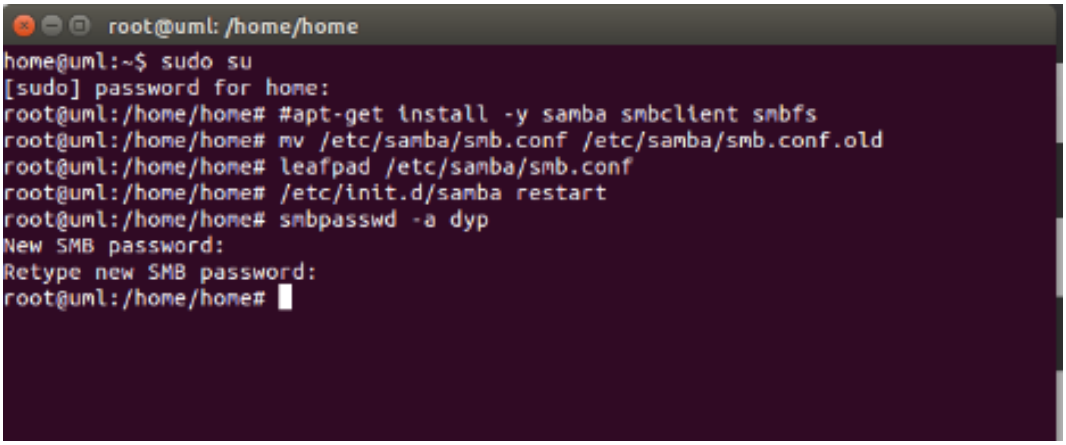

FIGURE 3.21 – Configuration PC1.

Fichier de partage avec leafpad

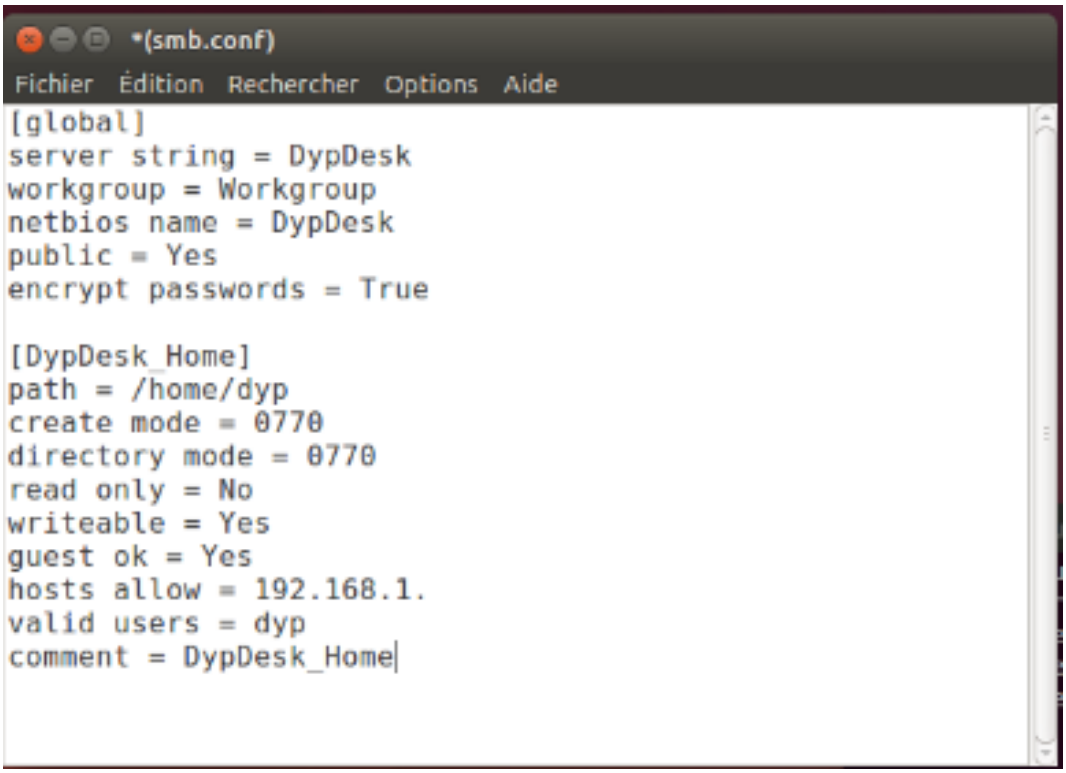

FIGURE 3.22 – Configuration fichier leafpad.

3.3.2 Configuration de pc 2

On va faire la même étape que le pc1 mais on Crée le fichier de partage avecmousepad le fichier de partage avec leafpad Ce fichier se décompose en deux parties : la section générale appelée "global" et la section de partage s'appelle : DypLap.Home

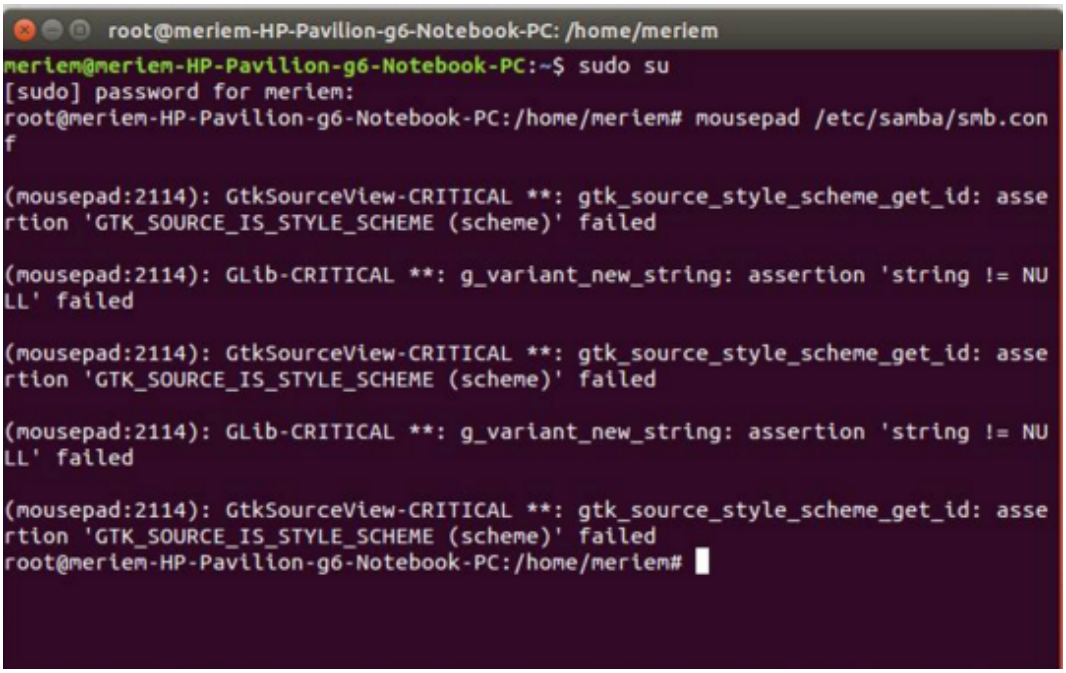

FIGURE 3.23 – Configuration PC2.

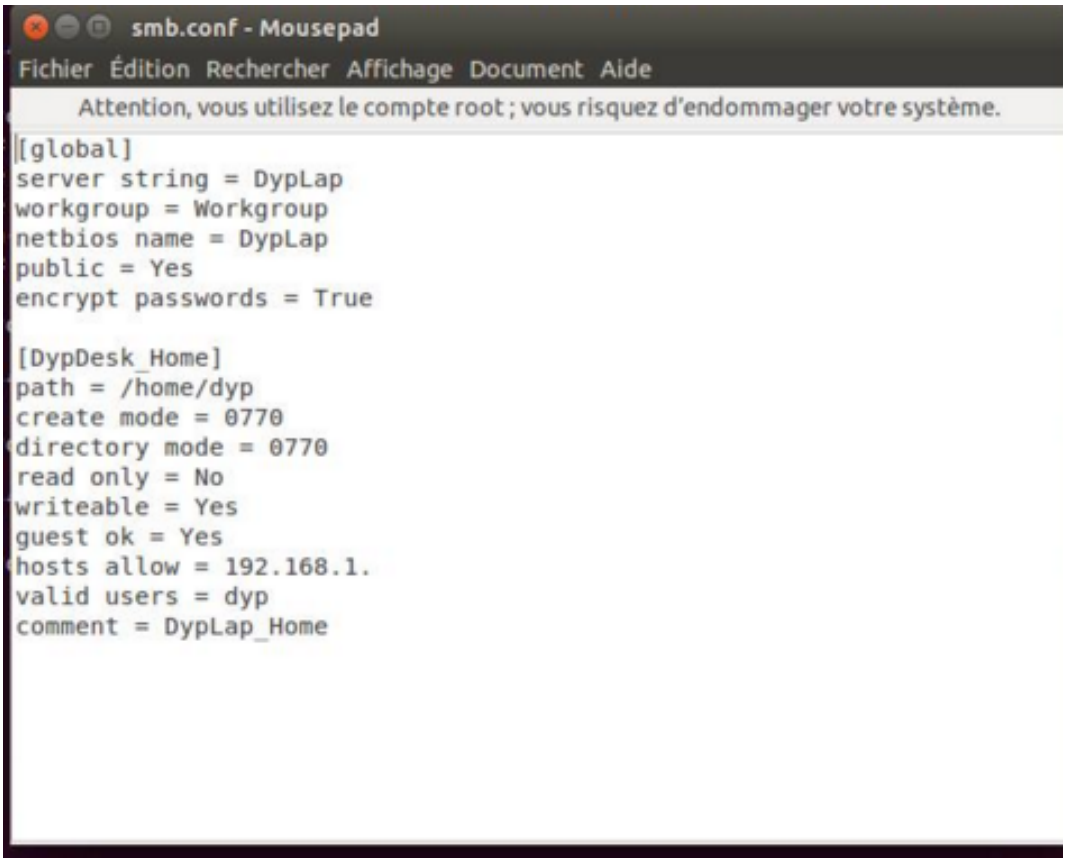

FIGURE 3.24 – Configuration fichier leafpad.

Nous avons configuré pc1 a l'aide de GADMIN-SAMBA

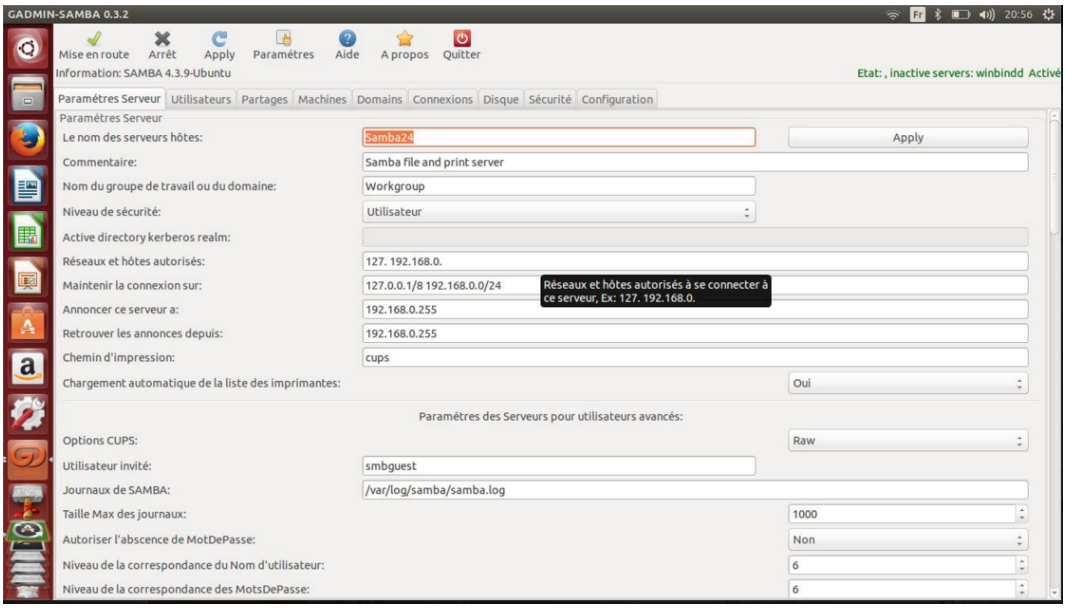

FIGURE 3.25 – Configuration pc1 avec Gsamba.

3.4 Connexion d'une machine virtuelle user mode linux au réseaux physique

User Mode Linux vous permet de créer des machines virtuelles Linux qui tournent en mode utilisateur sur votre machine réelle. Si pouvoir lancer une machine de manière isolée peut être utile, à un moment ou un autre, j'ai connecter une machine virtuelle User Mode Linux au réseau physique
La première étape pour relier une machine virtuelle UML au réseau va être de créer un tunnel TUN/TAP

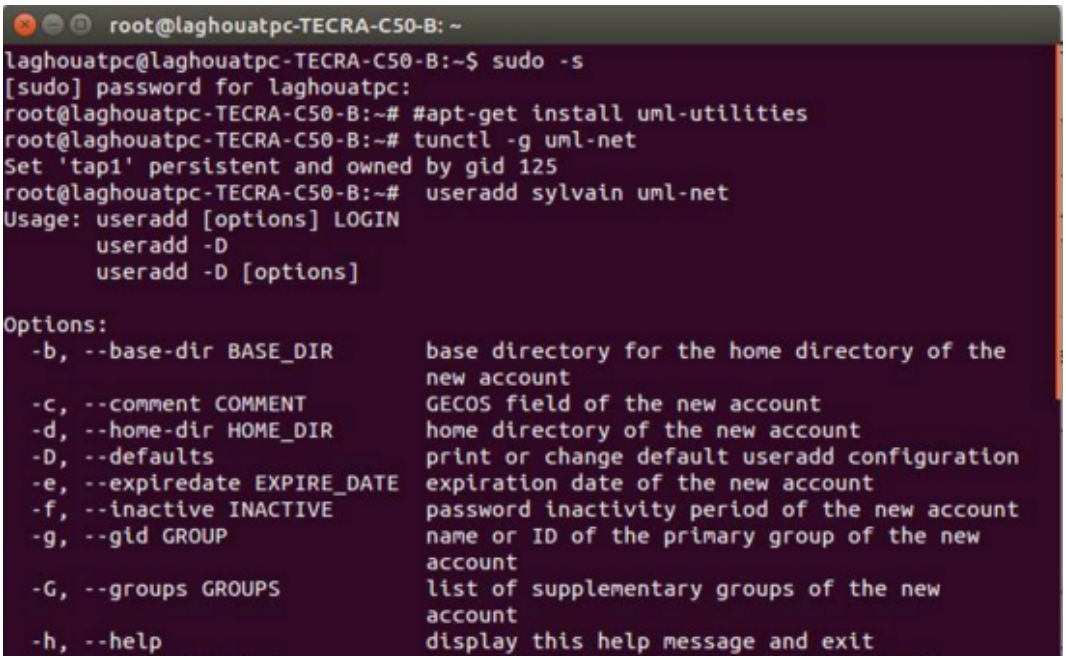

FIGURE 3.26 – Créer un tunnel.

On va Monter un pont Une machine Linux peut être configurée pour servir de pont. Cela permet d'interconnecter au niveau Ethernet plusieurs interfaces réseau. Le pont br0 interconnecte les réseaux reliés aux interfaces eth0, tap0 et tap1. Les trames Ethernet peuvent donc être acheminées entre ces interfaces exactement comme entre les ports d'un switch. Au niveau IP, seul le pont possède une adresse.

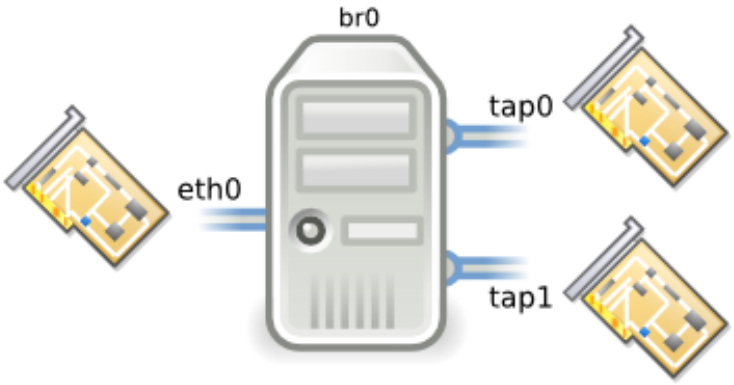

FIGURE 3.27 – Pont Linux.

Le pont lui-même aura une adresse IP. Et non les interfaces qui lui seront connectées. Il faut reconfigurer L'interface physique (eth0) et le tunnel (tap0)

```
laghouatpc@laghouatpc-TECRA-C50-B: ~
laghouatpc@laghouatpc-TECRA-C50-B:~$ #ifconfig eth0 0.0.0.0 promisc up
laghouatpc@laghouatpc-TECRA-C50-B:~$ #ifconfig tap0 0.0.0.0 promisc up
laghouatpc@laghouatpc-TECRA-C50-B:~$ #brctl addbr br0
laghouatpc@laghouatpc-TECRA-C50-B:~$ # brctl stp br0 off
laghouatpc@laghouatpc-TECRA-C50-B:~$ #brctl addif br0 eth1
laghouatpc@laghouatpc-TECRA-C50-B:~$ #brctl addif br0 eth1
laghouatpc@laghouatpc-TECRA-C50-B:~$  ifconfig br0 192.168.8.147 netmask 255.255
.255.0
SIOCSIFADDR: Operation not permitted
br0: ERROR while getting interface flags: No such device
SIOCSIFNETMASK: Operation not permitted
laghouatpc@laghouatpc-TECRA-C50-B:~$ # ifconfig br0 192.168.8.147 netmask 255.25
5.255.0
laghouatpc@laghouatpc-TECRA-C50-B:~$ route add default gw 192.168.8.254
SIOCADDRT: Operation not permitted
laghouatpc@laghouatpc-TECRA-C50-B:~$ route add default gw 192.168.8.254
SIOCADDRT: Operation not permitted
laghouatpc@laghouatpc-TECRA-C50-B:~$ #route add default gw 192.168.8.254
laghouatpc@laghouatpc-TECRA-C50-B:~$ linux ubda=lenny-image \
> con=pts  con0=fd:0,fd:1 \
>  umid=uml \
> eth0=tuntap,tap0
Locating the bottom of the address space ... 0x10000
Locating the top of the address space ... 0xc0000000
```

FIGURE 3.28 – Reconfigurer interface physique et le tunnel.

La machine virtuelle avec l'interface eth0 sera reliée au tunnel tap0 de

l'hôte. Dans le monde réel, cela correspondrait à l'ajout d'une nouvelle carte réseau et au branchement d'un câble entre carte et switch

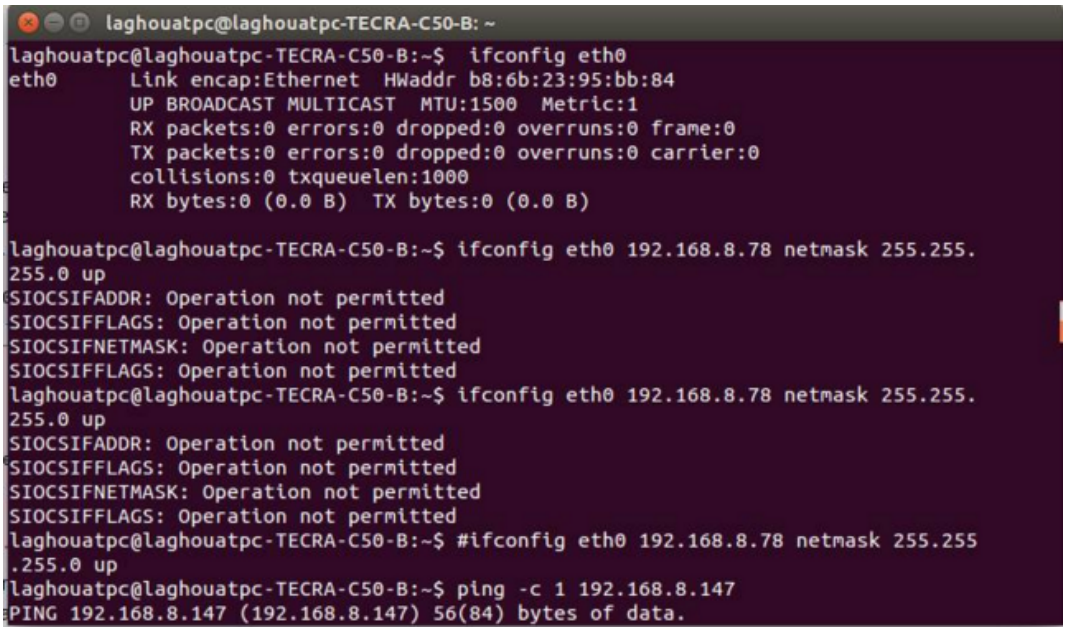

FIGURE 3.29 – liaison interface eth0 par un tunnel.

À partir de maintenant, l'hôte est accessible tout comme les autres machines situées sur le même sous-réseau.

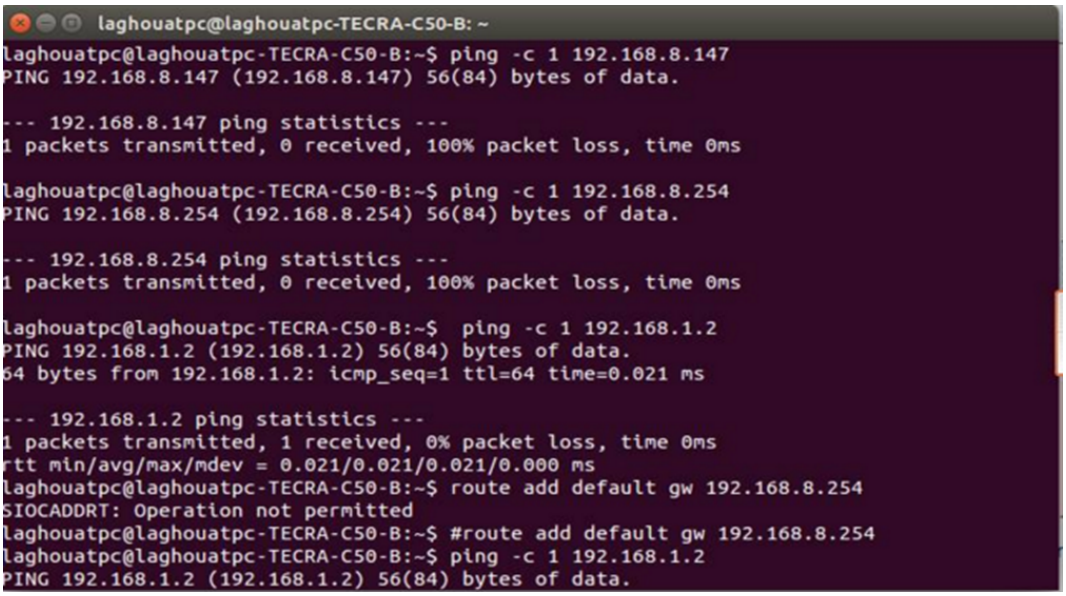

FIGURE 3.30 – Machine virtuelle accède au hôtel.

Pour la configuration de la machine virtuelle, On peut configurer le réseau dans /etc/network/interfaces.

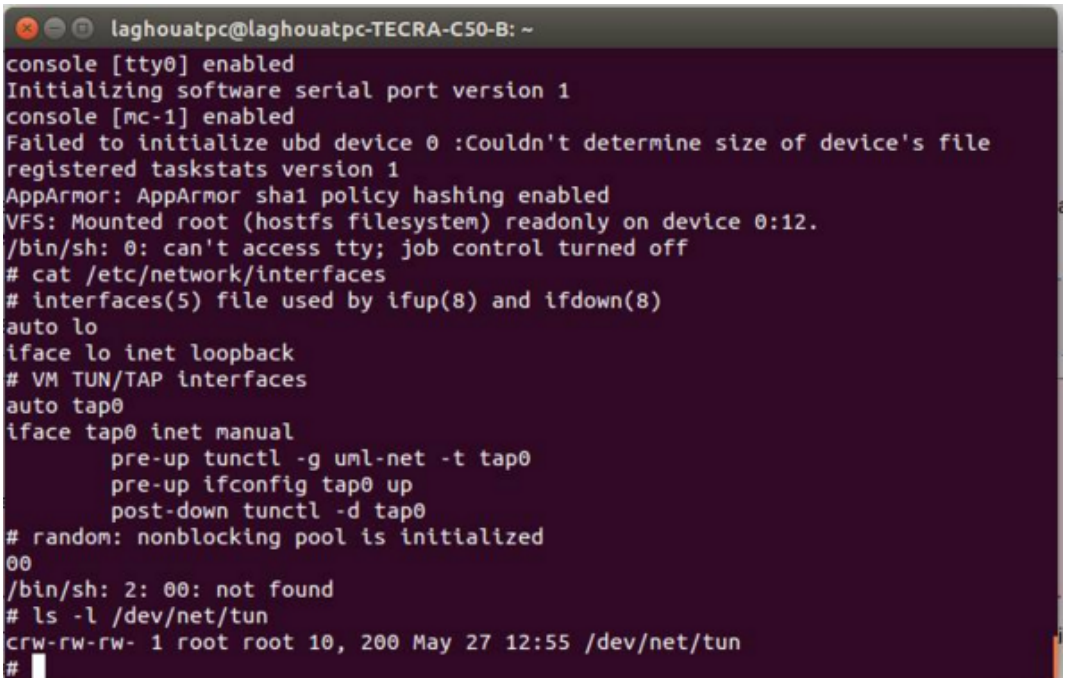

FIGURE 3.31 – Configuration de la machine user mode linux.

3.5 Conclusion :

Au cours de ce chapitre nous avons présenté une création des réseaux virtuelle avec user mode linux la première c'est la création d'un réseau virtuelle dans une seule machine physique et la deuxieme c'est la création de réseaux entre deux machines à distance et accéder àce réseau par la machine uml pour faire des partages de fichier entre machines.

Conclusion générale

La virtualisation est un domaine en pleine croissance, qui évolue très rapidement. Les utilisateurs peuvent s'en servir pour différents usages, aux besoins de leur fin. Les différentes solutions de virtualisation existantes utilisent des technologies variées.

Dans Ce travail ,on a étudier la virtualisation avec UML
Il nous a permis de :

• Comprendre UML et son utilisation

• La notion du noyau d'un système linux

• Le choix et l'utilisation d'un noyau avec UML

• La compilation d'un noyau linux et son de débogage

• La création de laboratoires virtuels avec UML de deux manière différentes : la première un réseau virtuel dans la même machine cette approche est utilisé par nombreux simulation comme netkit La deuxième : connections d'une machine UML avec un réseau réel et accède aux fichiers partagés avec Samba.
Comme perspective :

• On souhait que UML sera utilisé dans les salles de TP de notre département pour profiter de sa puissance.

• La réalisation d'un laboratoire virtuel utilisant UML .

• Création d'un laboratoire virtuel pour réaliser des simulations et des émulations

Bibliographie

[1] [G. Di Battista ,M. Patrignani ,M. Pizzonia ,M. Rimondin 2007] *The poor man's system for experimenting computer networking*

[2] *http ://www.telecom-sudparis.eu/s2ia/user/daumont/uml.html*

[3] *https ://lwn.net/2001/features/OLS/pdf/pdf/uml.pdf*

[4] [Mohamed Bouras UNIVERSITÉ AMMAR TELIDJI LAGHOUAT 2013] *Etude des outils de simulation des réseaux (OMNeT++, Netkit)*

[5] *http ://user-mode-linux.sourceforge.net/UserModeLinux-HOWTO-4.html*

[6] *https ://2009.jres.org/planning files/slideshow/pdf/14.pdf*

[7] *https ://wapiti.telecomlille.fr/commun/ens/peda/options/st/rio/pub/exposes. /exposesrio2006/Miroux-Mende/applications.html*

[8] [Dike Jeff 2006] *User Mode Linux*

[9] [jacques laudru,jean-philippe vaudeborre 2005] *virtual model for ip network architecture lab ,maquette virtuelle de travaux pratiques pour architectuelle de travaux pratiques pour architectures de réseaux ip*

[10] [sébastien bilbeau ,27 novembre 2008] *10 commandes dangeureuse sous linux*

[11] [sylvain leroux ,7 oùt 2010] *premiers pas avec user mode linux*

[12] *http ://www.ijrst.com/images/short pdf/1426506573 Lokesh Madan 2*

[13] *https ://fr.wikipedia.org/wiki/Noyau de syst*

[14] *http ://www.linux-france.org/article/debutant/noyau-compi/noyau-compi-1.html*

[15] *https ://doc.ubuntu-fr.org/tutoriel/comment compiler un kernel sur mesure*

[16] [stelian pop ,5 fevrier 2003] *debogage du noyau linux*

[17] [published by association for computing machiney] *Debugging Kernel Modules with User Mode Linux*

[18] *https ://en.wikipedia.org/wiki/Procfs*

[19] *https ://en.wikipedia.org/wiki/Sysfs*

[20] *https ://en.wikipedia.org/wiki/GNU Zebra*

[21] *http ://www.cisco.com/cisco/web/support/CA/fr/109/1091/1091859 1.html*

[22] *https ://en.wikipedia.org/wiki/Routing Information Protocol*

[23] *http ://www.chicoree.fr/w/Connecter une machine virtuelle UML*

Bibliographie