

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 SYSTÈMES 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	Examineur	Université de Laghouat
Mr Ameer Mohamed el amine	MAA	Examineur	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 chères 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

<b>Introduction générale</b>	<b>14</b>
<b>1 UML</b>	<b>16</b>
1.1 Introduction	16
1.2 Définition UML (User Mode Linux)	16
1.3 Objectif de UML	16
1.4 Architecture de UML	17
1.5 préparation d'um machine virtuelle UML	17
1.5.1 Résultat de Fichiers COW et Rootfs :	18
1.6 Avantages de UML	19
1.7 Inconvénient UML	20
1.8 Application de UML	20
1.9 UML et la sécurité	20
1.10 Conclusion	21
<b>2 L'utilité de UML dans l'environnement d'éducation</b>	<b>22</b>
2.1 Introduction	22
2.2 Utilité d'Uml dans l'enseignements	22
2.3 Utilisation UML pour apprendre le noyau linux	23
2.4 L'exécution des commandes dangereuse en Uml	23
2.5 Remarque et discussions	27
2.6 Compilation du noyau linux	27
2.6.1 Définition d'un noyau (kernel)	27
2.7 Objectif de la compilation du noyau	28
2.8 Les étapes de compilation d'un noyau :	29
2.8.1 Exemple installation de noyau (vanille) avec une distribution debian :	29
2.9 la compilation avec UML :	30
2.10 Un débogage du noyau après la compilation :	30
2.11 Les opérations de compilation	35

2.11.1	La première compilation d'un noyau : (sans utiliser UML)	35
2.11.2	La deuxième compilation du noyau utilisant UML :	39
2.11.3	Résultat :	48
2.12	Conclusion	50
<b>3</b>	<b>Création d'un laboratoire virtuel</b>	<b>51</b>
3.1	Introduction	51
3.2	Creation d'un laboratoire avec user mode linux dans une seule machine	51
3.2.1	Configuration des UML	60
3.3	Creation d'une laboratoire virtuel avec des machines distinctes	70
3.3.1	Configuration du PC1 :	71
3.3.2	Configuration de pc 2	72
3.4	Connexion d'une machine virtuelle user mode linux au réseaux physique	75
3.5	Conclusion :	80
	<b>Conclusion générale</b>	<b>81</b>

# Table des figures

1.1	Architecture de UML [3] . . . . .	17
1.2	Fichiers Rootfs et Cow [7] . . . . .	19
2.1	Commande fdisk sous user mode linux . . . . .	24
2.2	Commande hdparm sous user mode linux . . . . .	25
2.3	Commande mkfs sous user mode linux . . . . .	25
2.4	Commande rm-rf/ sous user mode linux . . . . .	26
2.5	Commande rm-rf* sous user mode linux . . . . .	26
2.6	Noyau linux et ces fonctionnalité . . . . .	28
2.7	Exemple de débogage avec GDB . . . . .	31
2.8	Exemple de débogage avec KDB . . . . .	32
2.9	Exemple de débogage avec KGDB . . . . .	33
2.10	Exemple de débogage avec UML . . . . .	34
2.11	Exemple de débogage avec LKCD . . . . .	35
2.12	installation de différents paquets nécessaire pour la compilation . . . . .	36
2.13	Lancement d'installation des sources . . . . .	37
2.14	Le crash . . . . .	38
2.15	Le crash . . . . .	38
2.16	Session inaccessible n . . . . .	39
2.17	éliminer l'ancienne configuration . . . . .	40
2.18	lancement d'opération de nettoyage . . . . .	40
2.19	Rendre la configuration par défaut . . . . .	41
2.20	Lancement d'interface de configuration . . . . .	42
2.21	l'interface menuconfig . . . . .	42
2.22	l'option « Initial RAM filesystem and RAM disk (initramfs/initrd) support » est activée . . . . .	43
2.23	Vérification d'existence « kernel hacking » . . . . .	44
2.24	lancement de compilation du noyau . . . . .	45
2.25	lancement de compilation du noyau . . . . .	45
2.26	Le temps consacré par la compilation . . . . .	46
2.27	Vérification de fichier vmlinux . . . . .	46

2.28	Lancement de machine UML. . . . .	46
2.29	Débugage de noyau utilisant gdb et Uml . . . . .	47
2.30	Execution UML sous GDB et création de break point . . . . .	47
2.31	Version de noyau noyau . . . . .	48
2.32	Ouverture de fichier log . . . . .	48
2.33	Disparition des erreurs après la compilation . . . . .	49
2.34	Un fichier log sans erreur . . . . .	49
3.1	Normaliser l'environnement user mode linux . . . . .	52
3.2	Monter le home via hostfs . . . . .	52
3.3	Utiliser des privilèges d'accès UML . . . . .	53
3.4	Interface tap R1L . . . . .	54
3.5	Interface tap R2 . . . . .	55
3.6	lancement de user mode switchs . . . . .	56
3.7	Configuration la machine 1. . . . .	57
3.8	Configuration de machine 1 switch . . . . .	58
3.9	Communiquer machine1 UML avec switchh . . . . .	59
3.10	Communiquer machine2 UML avec switch . . . . .	60
3.11	Configuration fichier quagga . . . . .	62
3.12	Configuration de fichier zebra. . . . .	63
3.13	Configuration de fichier ospf . . . . .	64
3.14	Configuration machine R1 . . . . .	65
3.15	Configuration machine R2 . . . . .	66
3.16	Configuration interface ethernet . . . . .	67
3.17	Connexion des machine virtuelles.. . . . .	68
3.18	Configuration de table de routage R1. . . . .	69
3.19	Configuration table de routage de R3. . . . .	70
3.20	Pinge de R2 depuis R1. . . . .	70
3.21	Configuration PC1. . . . .	71
3.22	Configuration fichier leafpad. . . . .	72
3.23	Configuration PC2. . . . .	73
3.24	Configuration fichier leafpad. . . . .	74
3.25	Configuration pc1 avec Gsamba. . . . .	75
3.26	Créer un tunnel. . . . .	76
3.27	Pont Linux. . . . .	77
3.28	Reconfigurer interface physique et le tunnel. . . . .	77
3.29	liaison interface eth0 par un tunnel. . . . .	78
3.30	Machine virtuelle accède au hôtel. . . . .	79
3.31	Configuration de la machine user mode linux. . . . .	80

# Introduction générale

La virtualisation est une technologie qui permet d'exécuter simultanément plusieurs systèmes d'exploitation et plusieurs applications sur un même ordinateur.

Plusieurs logiciels de virtualisation existent ont cité VMware, Virtual box, Xen, User Mode Linux (UML) qui sont différents en matière de performances et de fonctionnalité. Parmi ces logiciels on est va étudier par la virtualisation UML.

Plusieurs simulateurs (émulateurs) travaillent de la même manière 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 ethernet. Un micro-ordinateur de faible puissance (exemple : Pentium III) peut supporter 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 juste un hobby 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 est d'étudier UML et ses applications d'utilisation. Il est composé de trois chapitres.

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 être 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 hôte,et un système de fichiers sauvegardé dans un fichier de système de la machine hôte.

### 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].

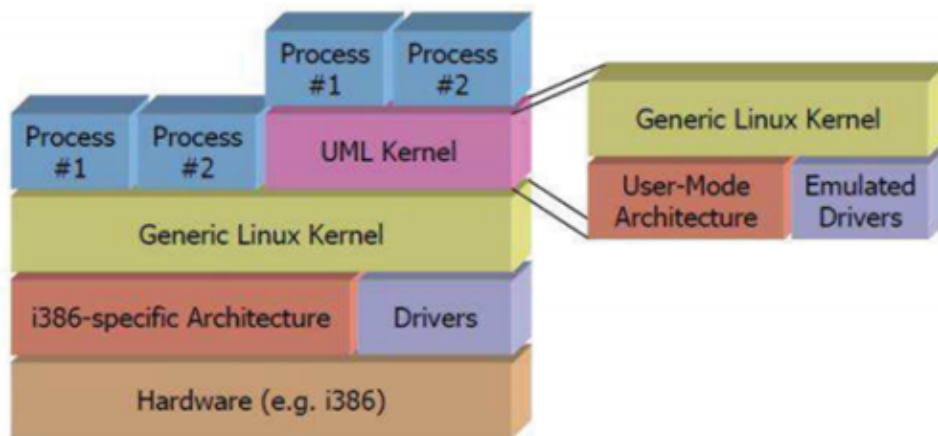


FIGURE 1.1 – Architecture de UML [3] .

## 1.5 préparation d'une machine virtuelle UML

l'installation de UML, peut être faite comme n'importe quelle 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 les changements 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 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].

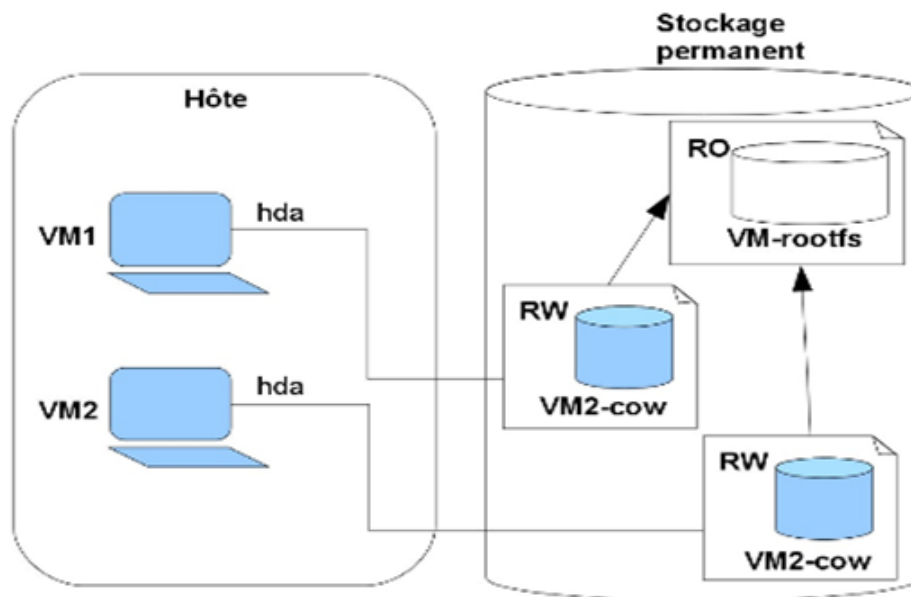


FIGURE 1.2 – Fichiers Rootfs et Cow [7] .

## 1.6 Avantages de UML

De manière générale :

- 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 système, 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 Inconvénient UML

le seul défaut remarqué jusqu'à présent, et qui 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 principale plate-forme de développement, et démarrèrent leurs travaux sur machines physiques plus tard après débogage.

- **Réalisation de réseaux virtuels**

UML est largement utilisé pour créer des réseaux virtuels (des laboratoires) 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 système.

## 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 utilisés, il ne demande pas un matériel très performant, avec plusieurs machines 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. La programmation du noyau fournit un authentique noyau pour modifier, avec les outils de développement et de débogage qui devraient déjà être familiers. 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 quelques 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

```

root@unl:~# fdisk -l

Disk /dev/sda: 320.1 GB, 320072933376 bytes
255 têtes, 63 secteurs/piste, 38913 cylindres, total 625142448 secteurs
Unités = secteurs de 1 * 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Identifiant de disque : 0x000bb3c9

Périphérique Amorce Début Fin Blocs Id Système
/dev/sda1 * 2048 122882047 61440000 7 HPFS/NTFS/exFAT
/dev/sda2 122882048 374009855 125563904 7 HPFS/NTFS/exFAT
/dev/sda3 374009856 517369855 71680000 7 HPFS/NTFS/exFAT
/dev/sda4 517371902 625139711 53883905 f Étendue W95 (LBA)
/dev/sda5 517371904 625139711 53883904 83 Linux
root@unl:~# fdisk -l /dev/sda

Disk /dev/sda: 320.1 GB, 320072933376 bytes
255 têtes, 63 secteurs/piste, 38913 cylindres, total 625142448 secteurs
Unités = secteurs de 1 * 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Identifiant de disque : 0x000bb3c9

Périphérique Amorce Début Fin Blocs Id Système
/dev/sda1 * 2048 122882047 61440000 7 HPFS/NTFS/exFAT
/dev/sda2 122882048 374009855 125563904 7 HPFS/NTFS/exFAT
/dev/sda3 374009856 517369855 71680000 7 HPFS/NTFS/exFAT
/dev/sda4 517371902 625139711 53883905 f Étendue W95 (LBA)
/dev/sda5 517371904 625139711 53883904 83 Linux
root@unl:~# fdisk -s /dev/sda2
125563904

```

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

```
laghouatpc@laghouatpc-TECRA-C50-B: ~/uml-images
laghouatpc@laghouatpc-TECRA-C50-B:~$ mkdir uml-images
laghouatpc@laghouatpc-TECRA-C50-B:~$ cd uml-images
laghouatpc@laghouatpc-TECRA-C50-B:~/uml-images$ dd if=/dev/zero of=lenny
count=0 obs=1MB seek=300
0+0 records in
0+0 records out
0 bytes (0 B) copied, 0,000289779 s, 0,0 kB/s
laghouatpc@laghouatpc-TECRA-C50-B:~/uml-images$ █
```

### 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 ]`

```
mini@uml:~$ sudo -s
sudo: unable to resolve host uml
[sudo] password for mini:
root@uml:~# hdparm -t /dev/sda1

/dev/sda1:
Timing buffered disk reads: 234 MB in 3.00 seconds = 84.59 MB/sec
root@uml:~# █
```

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`

```
# mkfs.ext3 /dev/sda
mke2fs 1.42.9 (4-Feb-2014)
/dev/sda is entire device, not just one partition!
Proceed anyway? (y,n) n
# █
```

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

```
mini@uml:~$ sudo -s
sudo: unable to resolve host uml
[sudo] password for mini:
root@uml:~# hdparm -t /dev/sda1

/dev/sda1:
Timing buffered disk reads: 254 MB in 3.00 seconds = 84.59 MB/sec
root@uml:~#
```

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

### La commande `rm -rf*` :

```
home@uml: ~
rm: cannot remove 'home/home/.uml/4Trf5w/pid': Read-only file system
rm: cannot remove 'home/home/.uml/le6LeI/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/le6LeI/pid': Read-only file system
rm: cannot remove 'home/home/.uml/hLBNp9/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/hLBNp9/pid': Read-only file system
rm: cannot remove 'home/home/.uml/8OckS8/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/8OckS8/pid': Read-only file system
rm: cannot remove 'home/home/.uml/H4ZIDy/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/H4ZIDy/pid': Read-only file system
rm: cannot remove 'home/home/.uml/X4gtsu/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/X4gtsu/pid': Read-only file system
rm: cannot remove 'home/home/.uml/Z1qxfJ/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/Z1qxfJ/pid': Read-only file system
rm: cannot remove 'home/home/.uml/m1igUk/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/m1igUk/pid': Read-only file system
rm: cannot remove 'home/home/.uml/VVy09E/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/VVy09E/pid': Read-only file system
rm: cannot remove 'home/home/.uml/s0knx5/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/s0knx5/pid': Read-only file system
rm: cannot remove 'home/home/.uml/6kbZo7/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/6kbZo7/pid': Read-only file system
rm: cannot remove 'home/home/.uml/IZ4TnJ/nconsole': Read-only file system
rm: cannot remove 'home/home/.uml/IZ4TnJ/pid': Read-only file system
```

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 conclud 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].

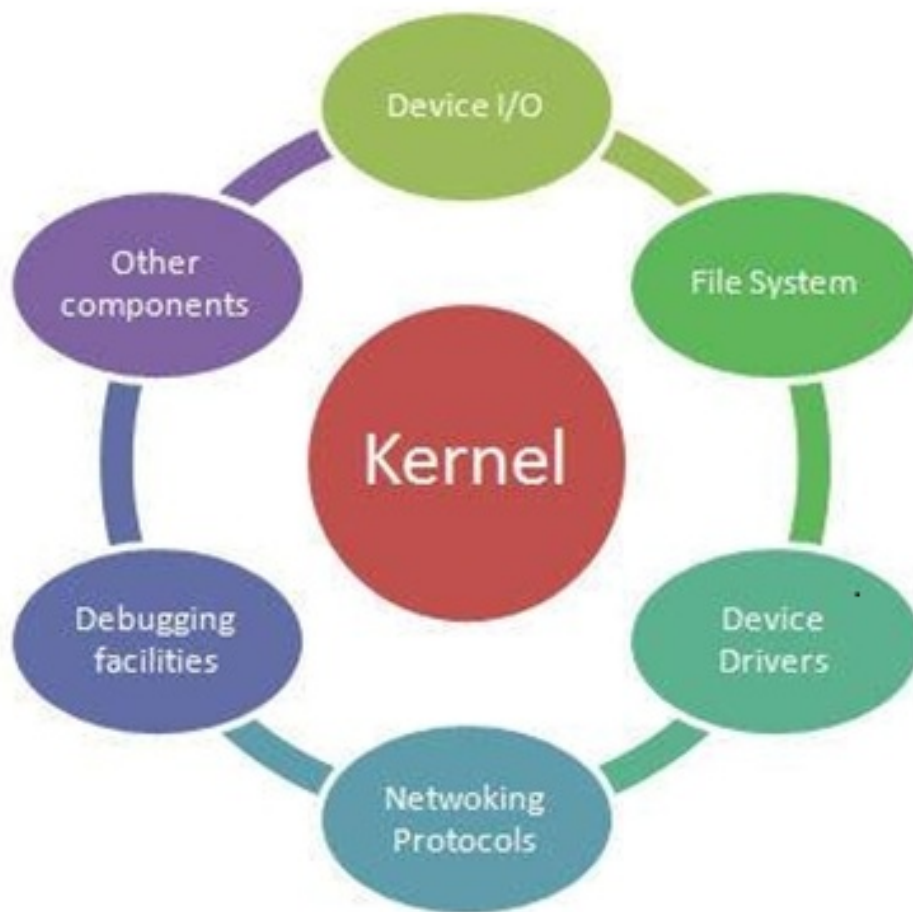


FIGURE 2.6 – Noyau linux et ces fonctionnalité .

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 personnaliser<sup>1</sup> 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 kernel-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 exécute simplement la commande suivante :

- `sudo dpkg -i kernel-image-2.6.12-custom_10.00.Custom_i386.deb`[17]

## 2.9 la compilation avec UML :

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

## 2.10 Un débogage du noyau après la compilation :

Au cours 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 système 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 :**

```

GNU gdb 5.0rh-5 Red Hat Linux 7.1
Copyright 2001 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
0xa00d5631 in __kill ()
Breakpoint 1 at 0xa000e36b: file panic.c, line 52.
Breakpoint 2 at 0xa00c867e: file user_util.c, line 104.
Breakpoint 3 at 0xa00035bf: file init/main.c, line 553.

Breakpoint 3, start_kernel () at init/main.c:553
553          printk(linux_banner);
(gdb) c
Continuing.
□

```

FIGURE 2.7 – Exemple de débogage avec GDB .

### **KDB :**

Débogueur fonctionnant sur la machine à déboguer (dans le handler du clavier). Permet de :

- Interrompre l'exécution du noyau ;
- Mettre des breakpoints (exécution, lecture/écriture) ;
- Exécuter le code en mode pas à pas ;
- Connaître la pile d'appel (tache courante, toutes les taches) ;
- Examiner/modifier la mémoire (adresses, listes chaînées etc) ;
- Examiner/modifier les registres des processeurs.
- Débogueur local à la machine (pas de matériel supplémentaire).
- Débogueur au niveau assembleur

### **Exemple :**

```

Entering kdb (current=0xc02aa000, pid 0) due to Keyboard Entry
kdb> bp sys_open
Instruction(i) BP #0 at 0xc01300cc (sys_open)
      is enabled globally adjust 1
kdb> go
Instruction(i) breakpoint #0 at 0xc01300cc (adjusted)
0xc01300cc sys_open:      int3

Entering kdb (current=0xcacd6000, pid 416) due to Breakpoint @ 0xc01300cc
kdb> bt
0xcacd6000 00000416 00000001 1 000 run 0xcacd6270*sendmail
ESP      EIP      Function (args)
0xcacd7fc0 0xc01300cc sys_open (0x80c0544, 0x0, 0x1b6, 0x80f93a0, 0x80de260)
      kernel .text 0xc0100000 0xc01300cc 0xc013014c
0xcacd7fc4 0xc01088eb system_call+0x33
      kernel .text 0xc0100000 0xc01088b8 0xc01088f0
kdb> ps
Task Addr  Pid      Parent  [*] cpu State Thread      Command
[... ]
0xcacd6000 00000416 00000001 1 000 run 0xcacd6270*sendmail
0xcac2a000 00000426 00000001 1 000 stop 0xcac2a270 sendmail
kdb> ss
SS trap at 0xc01300cd (sys_open+0x1)
0xc01300cd sys_open+0x1:      push  %ebx
kdb> ss
SS trap at 0xc01300ce (sys_open+0x2)
0xc01300ce sys_open+0x2:      pushl 0xc(%esp,1)
kdb> ss
SS trap at 0xc01300d2 (sys_open+0x6)
0xc01300d2 sys_open+0x6:      call 0xc0138da0 getname
kdb> bc 0
Breakpoint 0 at 0xc01300cc cleared
kdb> go

```

FIGURE 2.8 – Exemple de débogage avec KDB .

### KGDB :

Permet de :

- Interrompre l'exécution du noyau ;
- Mettre des breakpoints (execution, lecture/ecriture) ;
- Executer le code en mode pas a pas ;
- Connaitre la pile d'appel (tache courante, toutes les taches) ;  
Examiner/modifier la memoire (adresses, variables, listes chainees 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'état, pas des événements

**Exemple :**

```

=====
LCRASH CORE FILE REPORT
=====
GENERATED ON:
    Wed Jan 22 16:44:03 2003

TIME OF CRASH:
    Wed Jan 22 16:42:01 2003

PANIC STRING:
    sysrq
    ...

=====
CURRENT SYSTEM TASKS
=====

```

ADDR	UID	PID	PPID	STATE	FLAGS	CPU	NAME
0xc022e000	0	0	0	0	0	0	swapper
0xc121e000	0	1	0	1	0x100	0	init
...							
0xca676000	0	545	1	1	0x100	0	mingetty
0xca55c000	0	548	540	1	0x100	0	bash

```

=====
STACK TRACE OF FAILING TASK
=====

STACK TRACE FOR TASK: 0xc022e000 (swapper)

0 default_idle+35 [0xc0106ccf]
1 cpu_idle+64 [0xc0106d38]
2 start_kernel+257 [0xc0230669]
3 is386+74 [0xc010018c]
=====

```

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

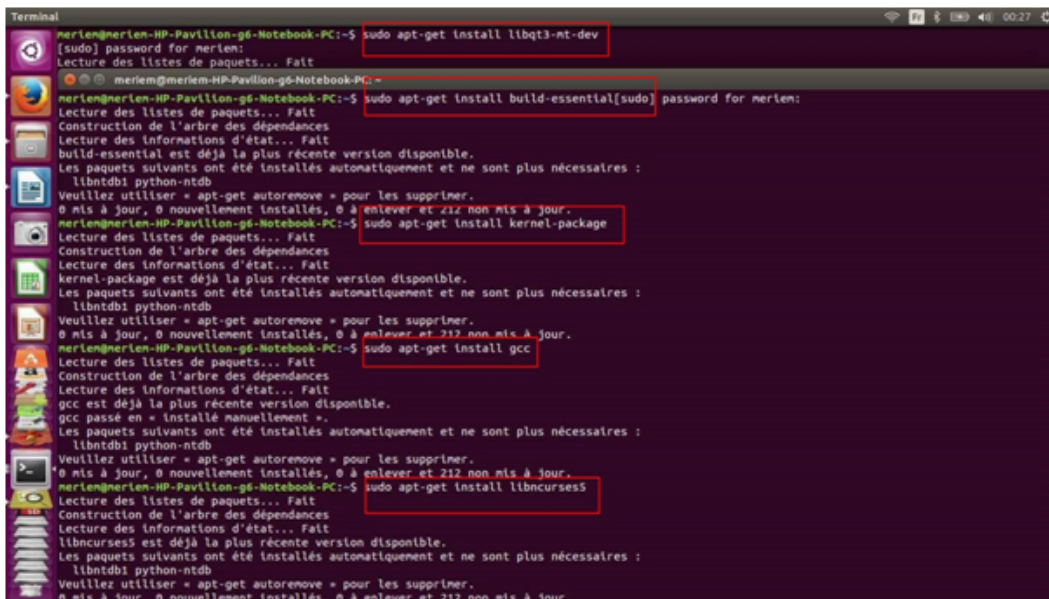
apt-get install build-essential

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 bibliothèques 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'installation de
sudo apt-get install kernel-package
sudo apt-get install gcc
```



```
Terminal
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ sudo apt-get install libqt3-mt-dev
[sudo] password for merlen:
Lecture des listes de paquets... Fait
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ sudo apt-get install build-essential[sudo] password for merlen:
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des Informations d'état... Fait
build-essential est déjà la plus récente version disponible.
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
 libntdbi python-ntdb
Veillez utiliser « apt-get autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 212 non mis à jour.
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ sudo apt-get install kernel-package
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des Informations d'état... Fait
kernel-package est déjà la plus récente version disponible.
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
 libntdbi python-ntdb
Veillez utiliser « apt-get autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 212 non mis à jour.
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ sudo apt-get install gcc
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des Informations d'état... Fait
gcc est déjà la plus récente version disponible.
gcc passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
 libntdbi python-ntdb
Veillez utiliser « apt-get autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 212 non mis à jour.
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ sudo apt-get install libncurses5
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des Informations d'état... Fait
libncurses5 est déjà la plus récente version disponible.
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
 libntdbi python-ntdb
Veillez utiliser « apt-get autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 212 non mis à jour.
```

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

## Deuxième étape :

2. Installation des sources :

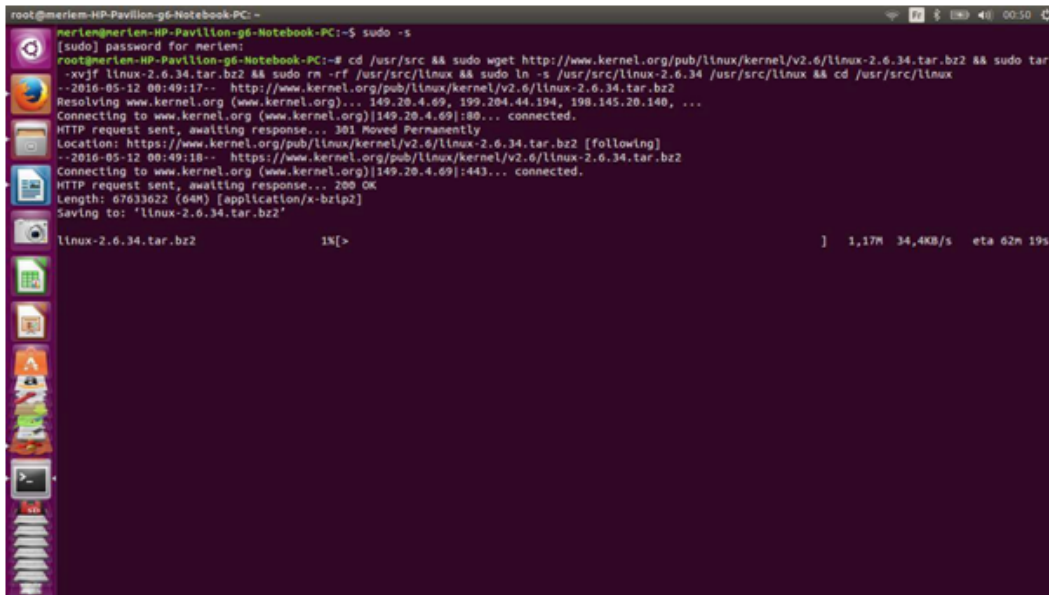


FIGURE 2.13 – Lancement d'installation des sources .

Après lancement de l'installation un crash est apparu 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 .



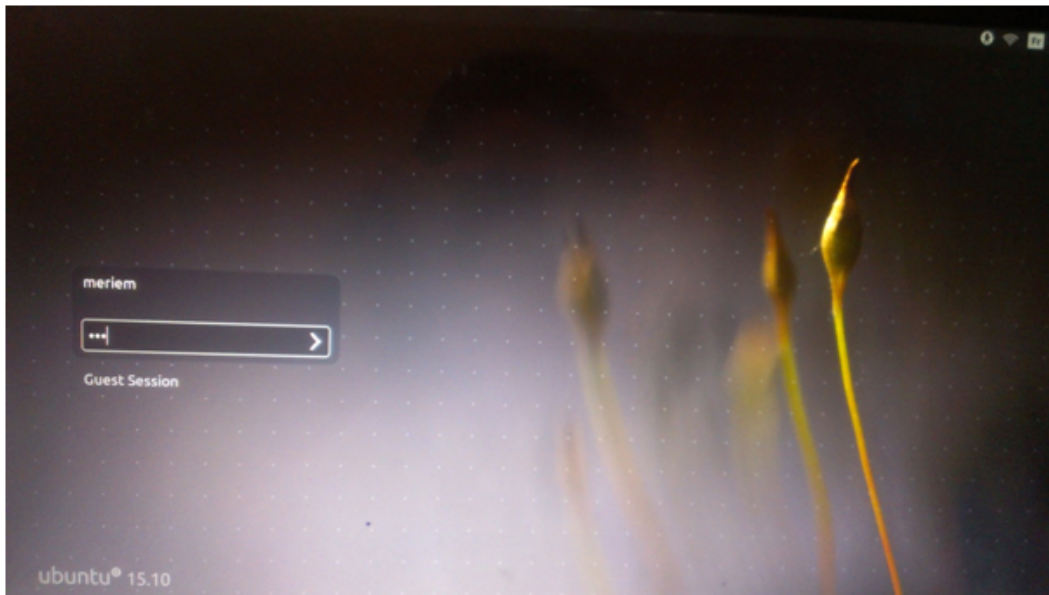


FIGURE 2.16 – Session inaccessible n .

### Discussion :

Après 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 empêché 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 readme fourni avec le noyau on trouve que il est conseillé de faire `make mrproper` avant `make menuconfig` pour avoir un arbre de compilation tout propre

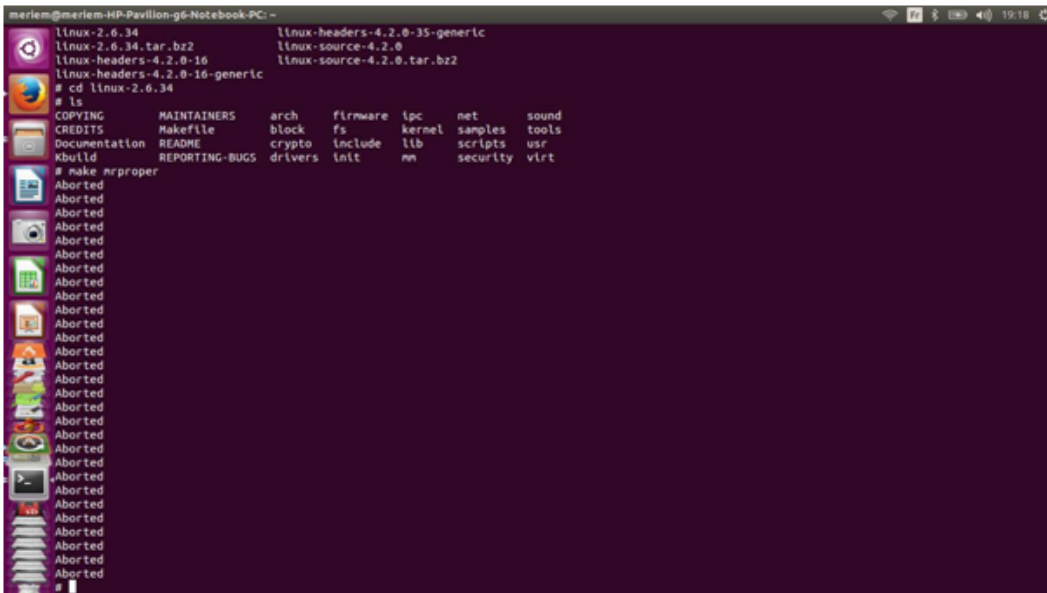


FIGURE 2.17 – éliminer l'ancienne configuration .

2. Voilà le lancement de cette opération

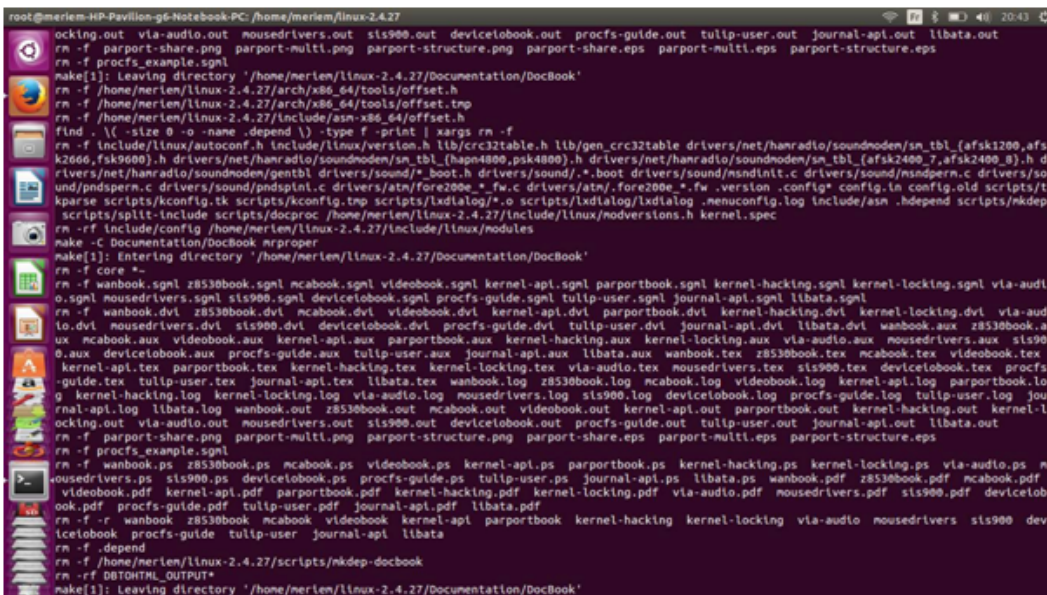


FIGURE 2.18 – lancement d'opération de nettoyage .

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

```

root@merlen-HP-Pavilion-g6-Notebook-PC:/usr/src/linux#
scripts/kconfig/lnages.c:61:20: warning: 'xpm_load' defined but not used [-Wunused-variable]
static const char *xpm_load[] = {
^
scripts/kconfig/lnages.c:36:20: warning: 'xpm_save' defined but not used [-Wunused-variable]
static const char *xpm_save[] = {
^
scripts/kconfig/lnages.c:66:20: warning: 'xpm_back' defined but not used [-Wunused-variable]
static const char *xpm_back[] = {
^
scripts/kconfig/lnages.c:175:20: warning: 'xpm_symbol_no' defined but not used [-Wunused-variable]
static const char *xpm_symbol_no[] = {
^
scripts/kconfig/lnages.c:192:20: warning: 'xpm_symbol_mod' defined but not used [-Wunused-variable]
static const char *xpm_symbol_mod[] = {
^
scripts/kconfig/lnages.c:209:20: warning: 'xpm_symbol_yes' defined but not used [-Wunused-variable]
static const char *xpm_symbol_yes[] = {
^
scripts/kconfig/lnages.c:226:20: warning: 'xpm_choice_no' defined but not used [-Wunused-variable]
static const char *xpm_choice_no[] = {
^
scripts/kconfig/lnages.c:243:20: warning: 'xpm_choice_yes' defined but not used [-Wunused-variable]
static const char *xpm_choice_yes[] = {
^
scripts/kconfig/lnages.c:277:20: warning: 'xpm_menu_inv' defined but not used [-Wunused-variable]
static const char *xpm_menu_inv[] = {
^
scripts/kconfig/lnages.c:294:20: warning: 'xpm_menuback' defined but not used [-Wunused-variable]
static const char *xpm_menuback[] = {
^
HOSTLD scripts/kconfig/gconf
scripts/kconfig/gconf Kconfig
root@merlen-HP-Pavilion-g6-Notebook-PC:/usr/src/linux# make ARCH=x86 defconfig
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
*** Default configuration is based on 'x86_64_defconfig'
kernel/time/Kconfig:157:warning: range is invalid
#
# configuration written to .config
#
root@merlen-HP-Pavilion-g6-Notebook-PC:/usr/src/linux#

```

FIGURE 2.19 – Rendre la configuration par défaut .

4. après on va revenir pour le choix d'option de configuration et pour cela on va exécuter la commande `make menuconfig ARCH=x86` 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 périphérique et toutes les fonctionnalités de Linux avec un même ordinateur, il s'agit donc de répondre aux questions appropriées selon la configuration. La moindre erreur de configuration peut endommager le système.

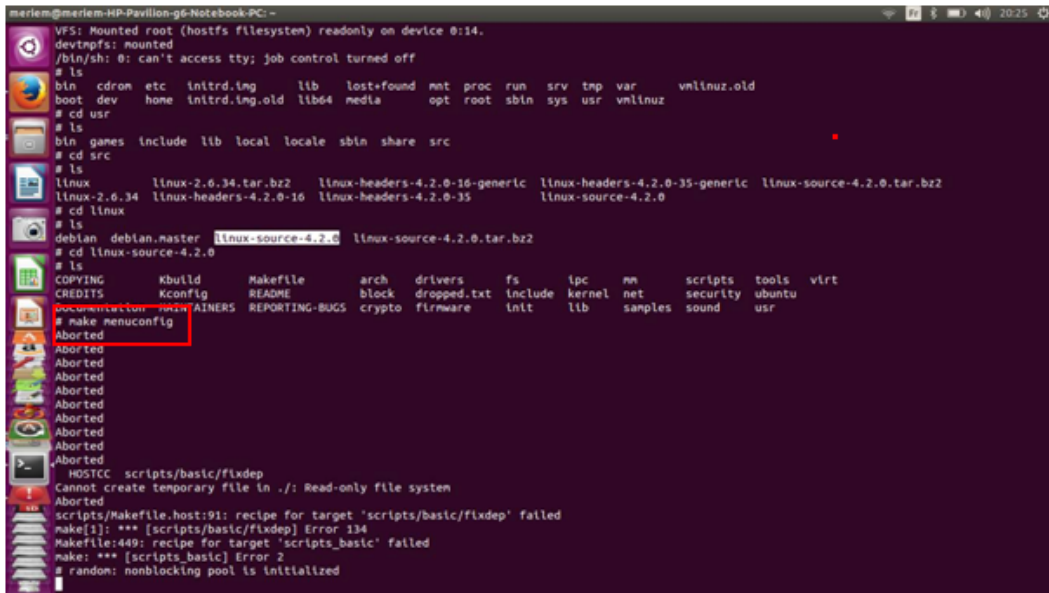


FIGURE 2.20 – Lancement d'interface de configuration .

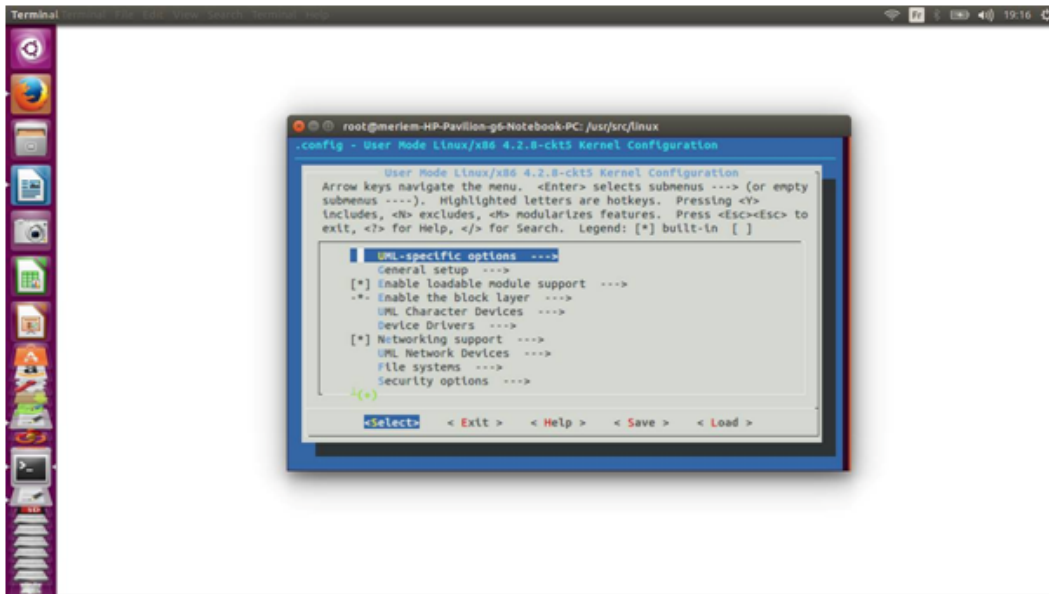


FIGURE 2.21 – l'interface menuconfig .

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

(initramfs/initrd) support » pour l activer

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

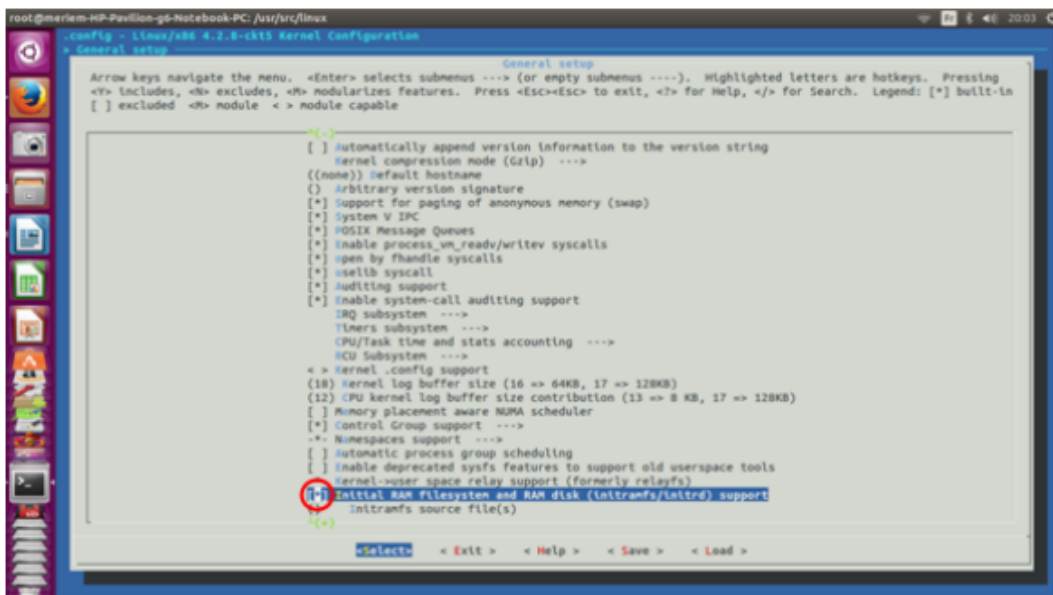


FIGURE 2.22 – l'option « Initial RAM filesystem and RAM disk (initramfs/initrd) support » est activée .

6. ensuite on passe a l'option kernel 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

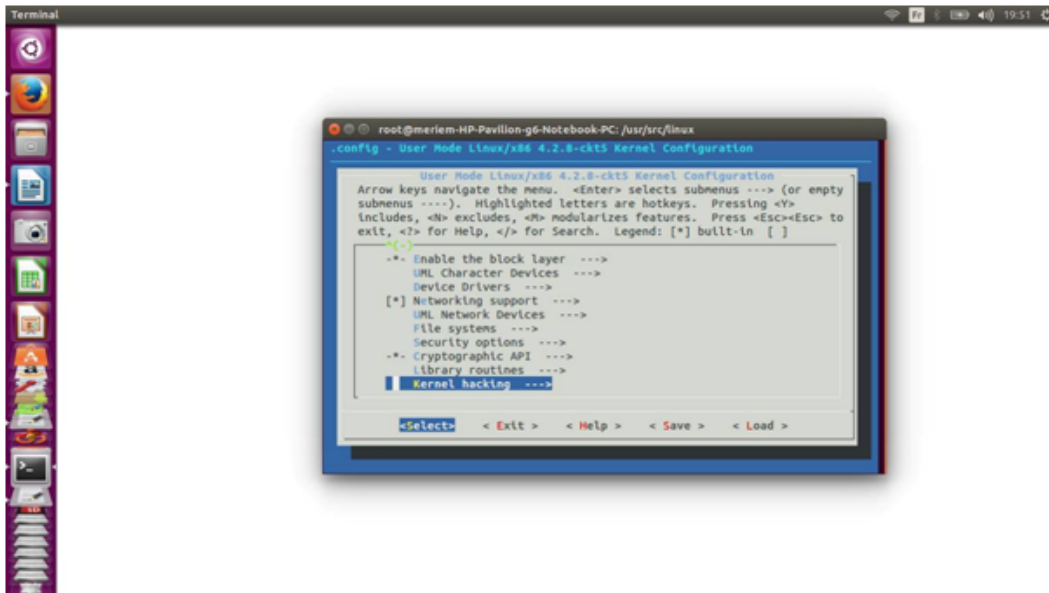


FIGURE 2.23 – Vérification d'existence « kenel hacking » .

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

```

root@meriem-HP-Pavillon-g6-Notebook-PC: /usr/src/linux
CC fs/filesystems.o
CC fs/namespace.o
CC fs/seq_file.o
CC fs/xattr.o
CC fs/lbf.o
CC fs/fs-writeback.o
CC fs/pnode.o
CC fs/splice.o
CC fs/sync.o
CC fs/utimes.o
CC fs/stack.o
CC [M] fs/fs_struct.o
CC [M] fs/staifs.o
CC [M] fs/fs_pln.o
CC [M] fs/nsfs.o
CC fs/buffer.o
CC fs/block_dev.o
CC fs/direct_io.o
CC fs/mpage.o
CC fs/proc_namespace.o
LD fs/autofs4/built-in.o
CC [M] fs/autofs4/initt.o
CC [M] fs/autofs4/lnode.o
CC [M] fs/autofs4/root.o
CC [M] fs/autofs4/symlink.o
CC [M] fs/autofs4/waitq.o
CC [M] fs/autofs4/expire.o
CC [M] fs/autofs4/dev-ioctl.o
LD [M] fs/autofs4/autofs4.o
CC fs/devpts/lnode.o
LD fs/devpts/devpts.o
LD fs/devpts/built-in.o
LD fs/exofs/built-in.o
CC fs/ext4/balloc.o
CC fs/ext4/bitmap.o
CC fs/ext4/dir.o
CC fs/ext4/file.o
CC fs/ext4/fsync.o
CC fs/ext4/lalloc.o
CC fs/ext4/lnode.o

```

FIGURE 2.24 – lancement de compilation du noyau .

```

root@meriem-HP-Pavillon-g6-Notebook-PC: /usr/src/linux
CC block/blk-sysfs.o
CC block/blk-flush.o
CC block/blk-settings.o
CC block/blk-loc.o
CC block/blk-map.o
CC block/blk-exec.o
CC block/blk-merge.o
CC block/blk-softirq.o
CC block/blk-timeout.o
CC block/blk-topoll.o
CC block/blk-lb.o
CC block/blk-nq.o
CC block/blk-nq-tag.o
CC block/blk-nq-sysfs.o
CC block/blk-nq-cpu.o
CC block/blk-nq-cpunap.o
CC block/loctl.o
CC block/genhd.o
CC block/scsi_ioctl.o
CC block/partition-generic.o
CC block/loprto.o
CC block/partitions/check.o
CC block/partitions/msdos.o
CC block/partitions/efi.o
LD block/partitions/built-in.o
CC block/blk-cgroup.o
CC block/noop-losched.o
CC block/deadline-losched.o
LD block/built-in.o
CC [M] block/cfq-losched.o
LD drivers/amba/built-in.o
LD drivers/auxdisplay/built-in.o
CC drivers/base/component.o
CC drivers/base/core.o
CC drivers/base/bus.o
CC drivers/base/dd.o
CC drivers/base/syscore.o
CC drivers/base/driver.o
CC drivers/base/class.o
CC drivers/base/platform.o

```

FIGURE 2.25 – lancement de compilation du noyau .

cette opération prendra avec nous

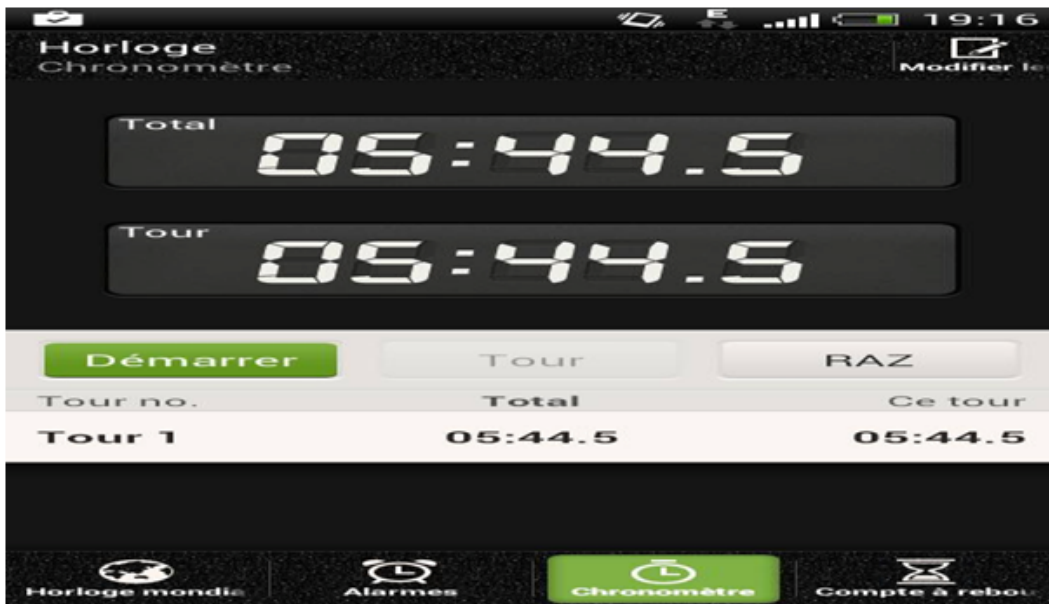


FIGURE 2.26 – Le temps consacré par la compilation .

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/tsofs/tsofs.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-4.2.0# ./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 :

```

merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$ ps -ef|grep linux
root      1002      1  0  na113  tty1      00:00:00 /sbin/agetty --noclear tty1 linux
merlen    1131    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/hud/window-stack-bridge
merlen    1203    1015  0  na113  ?         00:01:54 /usr/lib/x86_64-linux-gnu/hud/hud-service
merlen    1277    1015  0  na113  ?         00:00:15 /usr/lib/x86_64-linux-gnu/banf/banfdaemon
merlen    1361    1212  0  na113  ?         00:00:03 /usr/lib/x86_64-linux-gnu/indicator-application/indicator-application-service
merlen    1406    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/gconf/gconfd-2
merlen    1408    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-messages/indicator-messages-service
merlen    1409    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-bluetooth/indicator-bluetooth-service
merlen    1410    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-power/indicator-power-service
merlen    1411    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-datetime/indicator-datetime-service
merlen    1415    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-keyboard/indicator-keyboard-service --use-gtk
merlen    1416    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-sound/indicator-sound-service
merlen    1417    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-printers/indicator-printers-service
merlen    1419    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/indicator-session/indicator-session-service
merlen    1500    1015  0  na113  ?         00:00:02 /usr/lib/x86_64-linux-gnu/notlfy-osd
merlen    1609    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/zeitgeist-fts
merlen    1748    1212  0  na113  ?         00:00:02 /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
merlen    1892    1015  0  na113  ?         00:00:02 /usr/lib/x86_64-linux-gnu/unity-scope-home/unity-scope-home
merlen    1908    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/unity-lens-files/unity-files-daemon
merlen    2181    1015  0  na113  ?         00:00:00 /usr/lib/x86_64-linux-gnu/unity-lens-music/unity-music-daemon
merlen    16864   1015  0  20:20  ?         00:00:00 /usr/lib/x86_64-linux-gnu/ubuntu-geopm-provider
merlen    17644   1015  0  20:37  ?         00:00:00 /usr/lib/x86_64-linux-gnu/xfce4/xfconf/xfconfd
merlen    20315  20275  0  22:25  pts/6    00:00:00 grep --color=auto linux
merlen@merlen-HP-Pavillon-g6-Notebook-PC:~$

```

FIGURE 2.29 – Débogage de noyau utilisant gdb et Uml .

## 9. Débogage de noyau utilisant gdb et Uml

Exécuté la commande Gdb linux ;

Ensuite la commande `List sys_clone pour creun breakpoint (boucle fork)`;

```

root@merien-HP-Pavillon-g6-Notebook-PC: /usr/src/linux
1836 }
1837
1838 void __init proc_caches_init(void)
1839 {
1840     sighand_cache = kmem_cache_create("sighand_cache",
(gdb) b 1822
Breakpoint 1 at 0x6002b31a: file kernel/fork.c, line 1822.
(gdb) info break
Num      Type           Disp Enb Address              What
1        breakpoint      keep y   0x000000006002b31a in Sys_clone
                                                at kernel/fork.c:1822
(gdb) sh
Ambiguous command "sh": sharedlibrary, shell, show.
(gdb) attach 16864
Attaching to program: /usr/src/linux-source-4.2.0/linux, process 16864
Reading symbols from /lib64/ld-linux-x86-64.so.2...Reading symbols from /usr/lib
/debug//lib/x86_64-linux-gnu/ld-2.21.so...done.
done.
Error in re-setting breakpoint 1: Cannot access memory at address 0x6002b319
Error in re-setting breakpoint 1: Cannot access memory at address 0x6002b319
Error in re-setting breakpoint 1: Cannot access memory at address 0x6002b319
0x00007f4a5024f8dd in ?? ()
(gdb) attach 1002
A program is being debugged already. Kill it? (y or n)

```

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

```
Foot@merlen-HP-Pavillon-g6-Notebook-PC:~# uname -a
Linux merlen-HP-Pavillon-g6-Notebook-PC 4.2.0-35-generic #40-Ubuntu SMP Tue Mar 15 22:15:45 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
Foot@merlen-HP-Pavillon-g6-Notebook-PC:~#
```

FIGURE 2.31 – Version de noyau noyau .

```
Foot@merlen-HP-Pavillon-g6-Notebook-PC:~#
Starting WPA supplicant...
[ESC[32m OK [ESC[0m Started WPA supplicant.
[ESC[32m OK [ESC[0m Started LSB: start and stop UML networking services.
[ESC[0m[ESC[31m* [ESC[0m (1 of 4) A start job is running for LSB: start Samba
daemons for the AD DC (26s / 5min 17s)^MESC[K[ESC[1;31m*ESC[0m[ESC[31m* [ESC[0
m] (1 of 4) A start job is running for LSB: start Samba daemons for the AD DC (2
7s / 5min 17s)^MESC[K[ESC[31m*ESC[1;31m*ESC[0m[ESC[31m* [ESC[0m] (1 of 4) A star
t job is running for LSB: start Samba daemons for the AD DC (27s / 5min 17s)^M
^MESC[K Starting Network Manager Script Dispatcher Service...
[ESC[32m OK [ESC[0m Started Network Manager Script Dispatcher Service.
[ESC[32m OK [ESC[0m Started LSB: start Samba daemons for the AD DC.
[ESC[32m OK [ESC[0m Started LSB: start Samba NetBIOS nameserver (nmbd).
Starting LSB: start Samba SMB/CIFS daemon (smbd)...
[ESC[32m OK [ESC[0m Started LSB: start Samba SMB/CIFS daemon (smbd).
[ESC[32m OK [ESC[0m Started Detect the available GPUs and deal with any system
changes.
Starting Light Display Manager...
/var/log/boot.log (END)
```

FIGURE 2.32 – Ouverture de fichier log .

```

0.000000 Initializing cgroup subsys cpuset
0.000000 Initializing cgroup subsys cpu
0.000000 Initializing cgroup subsys cpacct
0.000000 Linux version 4.2.0-35-generic (buildd@lgw01-43) (gcc version 5.2.1 20151010 (Ubuntu 5.2.1-22ubuntu2)) #40-Ubuntu SMP Tue Mar 1
5 22:15:45 UTC 2016 (Ubuntu 4.2.0-35.40-generic 4.2.8-ckt5)
0.000000 Command line: BOOT_IMAGE=/boot/vmlinuz-4.2.0-35-generic root=UUID=cf282870-30d0-4a95-82b6-74e1ad981233 ro quiet splash vt.handof
f=7
0.000000 KERNEL supported cpus:
0.000000 Intel GenuineIntel
0.000000 AMD AuthenticAMD
0.000000 Centaur CentaurHauls
0.000000 x86/fpu: xstate_offset[2]: 0x240, xstate_sizes[2]: 0x100
0.000000 x86/fpu: Supporting XSAVE feature 0x01: 'x87 Floating point registers'
0.000000 x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
0.000000 x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
0.000000 x86/fpu: Enabled xstate features 0x7, context size is 0x340 bytes, using 'standard' format.
0.000000 x86/fpu: Using 'eager' FPU context switches.
0.000000 e820: BIOS-provided physical RAM map:
0.000000 BIOS-e820: [mem 0x0000000000000000-0x0000000000009d7fff] usable
0.000000 BIOS-e820: [mem 0x0000000000009d800-0x0000000000009ffff] reserved
0.000000 BIOS-e820: [mem 0x000000000000e0000-0x000000000000fffff] reserved
0.000000 BIOS-e820: [mem 0x0000000000100000-0x000000000000ace3ffff] usable
0.000000 BIOS-e820: [mem 0x000000000000ace3f000-0x000000000000acebeffff] reserved
0.000000 BIOS-e820: [mem 0x000000000000acebf000-0x000000000000acfbffff] ACPI NVS
0.000000 BIOS-e820: [mem 0x000000000000acfbf000-0x000000000000acffeffff] ACPI data
0.000000 BIOS-e820: [mem 0x000000000000acffff000-0x000000000000acffffff] usable
0.000000 BIOS-e820: [mem 0x000000000000ad00000-0x000000000000af9ffff] reserved
0.000000 BIOS-e820: [mem 0x00000000000000000-0x00000000000000fffff] reserved
0.000000 BIOS-e820: [mem 0x00000000000000000-0x000000000000003ffff] reserved
0.000000 BIOS-e820: [mem 0x00000000000000000-0x000000000000000ffff] reserved
0.000000 BIOS-e820: [mem 0x00000000000000000-0x0000000000000019ffff] reserved

```

FIGURE 2.33 – Disparition des erreurs après la compilation .

```

0.000000 BIOS-e820: [mem 0x00000000000000000-0x0000000000001401ffff] usable
0.000000 NX (Execute Disable) protection: active
0.000000 SMBIOS 2.7 present.
0.000000 DMI: Hewlett-Packard HP Pavilion g6 Notebook PC /166F, BIOS F.33 08/30/2011
0.000000 e820: update [mem 0x000000000-0x00000ffff] usable ==> reserved
0.000000 e820: remove [mem 0x000a00000-0x0000fffff] usable
0.000000 e820: last_pfn = 0x14fe00 max_arch_pfn = 0x400000000
0.000000 MTRR default type: uncachable
0.000000 MTRR fixed ranges enabled:
0.000000 00000-9FFFF write-back
0.000000 A0000-BFFFF uncachable
0.000000 C0000-E7FFF write-protect
0.000000 E8000-EFFFF write-combining
0.000000 F0000-FFFFFF write-protect
0.000000 MTRR variable ranges enabled:
0.000000 0 base 000000000 mask F80000000 write-back
0.000000 1 base 080000000 mask FC0000000 write-back
0.000000 2 base 0AD000000 mask FFF000000 uncachable
0.000000 3 base 0AE000000 mask FFE000000 uncachable
0.000000 4 base 0B0000000 mask FF0000000 uncachable
0.000000 5 base 0FFC00000 mask FFFC00000 write-protect
0.000000 6 base 100000000 mask FC0000000 write-back
0.000000 7 base 140000000 mask FF0000000 write-back
0.000000 8 base 14FE00000 mask FFFE00000 uncachable
0.000000 9 disabled
0.000000 x86/PAT: Configuration [0-7]: WB WC UC- UC WB WC UC- WT
0.000000 e820: last_pfn = 0xad000 max_arch_pfn = 0x400000000
0.000000 Found SMP MP-table at [mem 0x000fe1b0-0x000fe1bf] mapped at [ffff8800000fe1b0]
0.000000 Scanning 1 areas for low memory corruption
0.000000 Base memory trampoline at [ffff880000097000] 97000 size 24576
0.000000 reserving inaccessible SNB gfx pages
0.000000 init_memory_mapping: [mem 0x000000000-0x0000fffff]
0.000000 [mem 0x000000000-0x0000fffff] page 4k
0.000000 BRK [0x021f1000, 0x021f1fff] PGTABLE
0.000000 BRK [0x021f2000, 0x021f2fff] PGTABLE
0.000000 BRK [0x021f3000, 0x021f3fff] PGTABLE
0.000000 init_memory_mapping: [mem 0x14fc00000-0x14fdfffff]
0.000000 [mem 0x14fc00000-0x14fdfffff] page 2M
0.000000 BRK [0x021f4000, 0x021f4fff] PGTABLE
0.000000 init_memory_mapping: [mem 0x140000000-0x14fbfffff]
0.000000 [mem 0x140000000-0x14fbfffff] page 2M

```

FIGURE 2.34 – Un fichier 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 particulière 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’entrée en session sur un système Unix. Il est indispensable si vous souhaitez vous loguer sur votre machine

```

home@uml: ~
# mount -o remount,rw /
# touch /tmp/test1
# touch /etc/test1
touch: cannot touch '/etc/test1': Permission denied
# exec getty -n -l /bin/bash 38400 /dev/tty0
Kernel panic - not syncing: Attempted to kill init! exitcode=0x00000100

CPU: 0 PID: 1 Comm: getty Not tainted 3.13.0-rc7 #2
Stack:
 084c0c24 084c0c24 0854c4e3 0a870000 0855481c 0a870000 08426a03 00000000
 00000000 08421074 00000007 08633b5c 084cc052 0a877e54 0a870000 0855481c
 0a870000 0aa19034 0809b03d 084cc052 00000100 00000000 00000000 00000001
Call Trace:
[<08426a03>] ? dump_stack+0x23/0x27
[<08421074>] ? panic+0x81/0x170
[<0809b03d>] ? do_exit+0x3a1/0x830
[<0809c6be>] ? do_group_exit+0x69/0xc0
[<0809c728>] ? __wake_up_parent+0x0/0x34
[<08065817>] ? handle_syscall+0x6d/0x7f
[<0809484a>] ? userspace+0x3c9/0x49e
[<0811a0ec>] ? free_bprm+0x5b/0x60
[<0811a0ec>] ? free_bprm+0x5b/0x60
[<0811a6b9>] ? do_execve+0x5c8/0x5e1
[<08063061>] ? new_thread_handler+0x84/0x88

```

FIGURE 3.3 – Utiliser des privilèges d'accès UML .

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

```
home@uml:~  
home@uml:~$ tap tap-R1 br-R1R2  
No command 'tap' found, but there are 21 similar ones  
tap: command not found  
home@uml:~$ linux init=/bin/sh rootfstype=hostfs eth0=tuntap,tap-R1  
Locating the bottom of the address space ... 0x10000  
Locating the top of the address space ... 0xc0000000  
Core dump limits :  
    soft - 0  
    hard - NONE  
Checking that ptrace can change system call numbers...OK  
Checking syscall emulation patch for ptrace...OK  
Checking advanced syscall emulation patch for ptrace...OK  
Checking for tmpfs mount on /dev/shm.../run/shm...OK  
Checking PROT_EXEC mmap in /dev/shm/...OK  
Checking for the skas3 patch in the host:  
- /proc/mm...not found: No such file or directory  
- PTRACE_FAULTINFO...not found  
- PTRACE_LDT...not found  
UML running in SKAS0 mode  
Adding 2580480 bytes to physical memory to account for exec-shield gap  
Initializing cgroup subsys cpuset  
Initializing cgroup subsys cpu  
Initializing cgroup subsys cpuacct  
Linux version 3.13.0-rc7 (root@akateko) (gcc version 4.8.2 (Ubuntu/Linaro 4.8.2-
```

FIGURE 3.4 – Interface tap R1L .

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

```
home@uml: ~
home@uml:~$ tap tap-R2 br-R1R2
No command 'tap' found, but there are 21 similar ones
tap: command not found
home@uml:~$ linux init=/bin/sh rootfstype=hostfs eth0=tuntap,tap-R2
Locating the bottom of the address space ... 0x10000
Locating the top of the address space ... 0xc0000000
Core dump limits :
    soft - 0
    hard - NONE
Checking that ptrace can change system call numbers...OK
Checking syscall emulation patch for ptrace...OK
Checking advanced syscall emulation patch for ptrace...OK
Checking for tmpfs mount on /dev/shm.../run/shm...OK
Checking PROT_EXEC mmap in /dev/shm/...OK
Checking for the skas3 patch in the host:
- /proc/mem...not found: No such file or directory
- PTRACE_FAULTINFO...not found
- PTRACE_LDT...not found
UML running in SKAS0 mode
Adding 30937088 bytes to physical memory to account for exec-shield gap
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.13.0-rc7 (root@akateko) (gcc version 4.8.2 (Ubuntu/Linar
```

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.

```
root@uml: /etc/network
* Starting User-mode networking switch uml_switch [ OK ]
root@uml:/etc/network# sudo linux ubda=machine1.cow,lenny-image \
> con=pts con0=fd:0,fd:1 \
> eth0=daemon,,unix
Locating the bottom of the address space ... 0x0
Locating the top of the address space ... 0xc0000000
Core dump limits :
    soft - 0
    hard - NONE
Checking that ptrace can change system call numbers...OK
Checking syscall emulation patch for ptrace...OK
Checking advanced syscall emulation patch for ptrace...OK
Checking for tmpfs mount on /dev/shm.../run/shm...OK
Checking PROT_EXEC mmap in /dev/shm/...OK
Checking for the skas3 patch in the host:
- /proc/mem...not found: No such file or directory
- PTRACE_FAULTINFO...not found
- PTRACE_LDT...not found
UML running in SKAS0 mode
Adding 23281664 bytes to physical memory to account for exec-shield gap
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.13.0-rc7 (root@akateko) (gcc version 4.8.2 (Ubuntu/Linaro 4.8.2-
```

FIGURE 3.6 – lancement de user mode switches .

```
root@uml: /etc/network
root@uml:/etc/network# sudo ifup uml-tap
Set 'uml-tap' persistent and owned by gid 126
root@uml:/etc/network# sudo ifconfig uml-tap
uml-tap  Link encap:Ethernet  HWaddr fa:ca:cf:84:f7:a9
          inet addr:10.129.37.99  Bcast:10.129.37.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@uml:/etc/network# sudo cat /etc/default/uml-utilities
# Options to pass to uml_switch.

# set to "false" if you want to prevent uml_switch from
# starting with SysV scripts in /etc/init.d
# UML_SWITCH_START="false"

# For preconfigured tap setup, see
# /usr/share/doc/uml-utilities/examples/interfaces.example
#UML_SWITCH_OPTIONS="-tap tap0"

# User as which to run uml_switch
#UML_SWITCH_USER="uml-net"
```

FIGURE 3.7 – Configuration la machine 1.

```
root@uml: /etc/network
# Options to pass to uml_switch.

# set to "false" if you want to prevent uml_switch from
# starting with SysV scripts in /etc/init.d
# UML_SWITCH_START="false"

# For preconfigured tap setup, see
# /usr/share/doc/uml-utilities/examples/interfaces.example
#UML_SWITCH_OPTIONS="-tap tap0"

# User as which to run uml_switch
#UML_SWITCH_USER="uml-net"

# Socket file to use
# Debian's default is:
#UML_SWITCH_CTL="/var/run/uml-utilities/uml_switch.ctl"
#
# if you instead use your rolled up kernel from upstream
# sources you may want to uncomment the following:
#UML_SWITCH_CTL="/tmp/uml.ctl"
root@uml:/etc/network# sudo /etc/init.d/uml-utilities restart
 * Stopping User-mode networking switch uml_switch          [ OK ]
 * Starting User-mode networking switch uml_switch          [ OK ]
root@uml:/etc/network#
```

FIGURE 3.8 – Configuration de machine 1 switch .

On va faire communiquer deux UML à travers ce switch. La première 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
```

```
home@uml: ~
watchdog: Software Watchdog: cannot register miscdev on minor=130 (err=-16).
watchdog: Software Watchdog: a legacy watchdog module is probably present.
softdog: Software Watchdog Timer: 0.08 initialized. soft_noboot=0 soft_margin=60
sec soft_panic=0 (nowayout=1)
TCP: cubic registered
NET: Registered protocol family 17
Initialized stdio console driver
Console initialized on /dev/tty0
console [tty0] enabled
Initializing software serial port version 1
console [mc-1] enabled
Failed to initialize ubd device 0 :Couldn't determine size of device's file
Choosing a random ethernet address for device eth0
Netdevice 0 (12:e0:e1:27:9c:ff) :
TUN/TAP backend -
registered taskstats version 1
AppArmor: AppArmor sha1 policy hashing enabled
VFS: Mounted root (hostfs filesystem) readonly on device 0:12.
/bin/sh: 0: can't access tty; job control turned off
# ip link set up dev eth0
TUNSETIFF failed, errno = 1
RTNETLINK answers: Operation not permitted
# ip addr add 192.168.0.1/24 dev eth0
```

FIGURE 3.9 – Communiquer machine1 UML avec switchh .

**Et la deuxième 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
```

```
home@uml: ~
softdog: Software Watchdog Timer: 0.08 initialized. soft_noboot=0 soft_margi
sec soft_panic=0 (nowayout=1)
TCP: cubic registered
NET: Registered protocol family 17
Initialized stdio console driver
Console initialized on /dev/tty0
console [tty0] enabled
Initializing software serial port version 1
console [mc-1] enabled
Failed to initialize ubd device 0 :Couldn't determine size of device's file
Choosing a random ethernet address for device eth0
Netdevice 0 (be:ae:20:a6:2a:0d) :
TUN/TAP backend -
registered taskstats version 1
AppArmor: AppArmor sha1 policy hashing enabled
VFS: Mounted root (hostfs filesystem) readonly on device 0:12.
/bin/sh: 0: can't access tty; job control turned off
# ip link set up dev eth0
TUNSETIFF failed, errno = 1
RTNETLINK answers: Operation not permitted
# ip addr add 192.168.0.2/24 dev eth0
# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
```

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 :

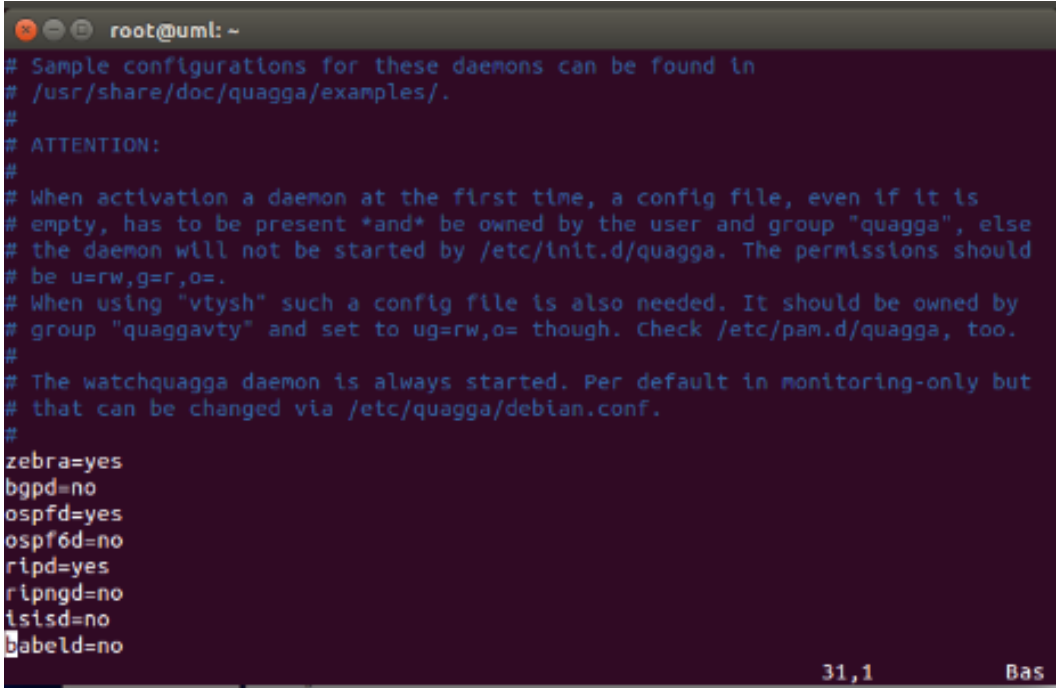
Case \$\$ in

```
# Inside UML. Three states :
TMP=$(mktemp -d)
trap "rm -rf $TMP" EXIT
check_dependencies
setup_screen
# Setup switches
setup_switch sitel
setup_switch sitel01
setup_switch internet
# Start VM
Start_vm R1 eth0=vde, $TMP/switch-sitel.sock
Start_vm R2 eth0=vde, $TMP/switch-sitel01.sock
Start_vm V1 eth0=vde, $TMP/switch-sitel.sock eth1=vde,$TMP/switch-internet.sock
Start_vm V2 eth0=vde, $TMP/switch-sitel.sock eth1=vde, $TMP/switch-internet.sock
start_vm V3 eth0=vde,$TMP/switch-sitel01.sock eth1=vde,$TMP/switch-internet.sock
start_vm V4 eth0=vde,$TMP/switch-sitel01.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.



```
root@uml: ~
# Sample configurations for these daemons can be found in
# /usr/share/doc/quagga/examples/.
#
# ATTENTION:
#
# When activation a daemon at the first time, a config file, even if it is
# empty, has to be present *and* be owned by the user and group "quagga", else
# the daemon will not be started by /etc/init.d/quagga. The permissions should
# be u=rw,g=r,o=.
# When using "vtysh" such a config file is also needed. It should be owned by
# group "quaggavty" and set to ug=rw,o= though. Check /etc/pam.d/quagga, too.
#
# The watchquagga daemon is always started. Per default in monitoring-only but
# that can be changed via /etc/quagga/debian.conf.
#
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=yes
ripngd=no
isisd=no
babeld=no
```

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]

```
home@uml: /etc/quagga
- *- zebra - *-
!
! zebra sample configuration file
!
! $Id: zebra.conf.sample,v 1.1 2002/12/13 20:15:30 paul Exp $
!
hostname Router
password zebra
enable password zebra
!
! Interface's description.
!
!interface lo
! description test of desc.
!
!interface sit0
! multicast
!
! Static default route sample.
!
!ip route 0.0.0.0/0 203.181.89.241
!
"zebra.conf" [lecture-seule] 25L, 385C 1,1 Hat
```

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].



```

echo 192.168.15.1 $HOSTNAME >> /etc/hosts

modprobe dummy

modprobe dummy numdummies=3

ifconfig dummy0 $HOSTNAME

ifconfig dummy0 192.168.15.1

route add $HOSTNAME dummy0

route add 192.168.15.1 dummy0 :0

route -n

```

```

root@uml: ~
home@uml:~$ sudo -s
[sudo] password for home:
root@uml:~# echo 192.168.15.1 $HOSTNAME >> /etc/hosts
root@uml:~# modprobe dummy
root@uml:~# modprobe dummy numdummies=3
root@uml:~# ifconfig dummy0 $HOSTNAME
root@uml:~# ifconfig dummy0:0 192.168.15.1
root@uml:~# route add $HOSTNAME dummy0
root@uml:~# route add 192.168.15.1 dummy0:0
root@uml:~# route -n
Kernel IP routing table
Destination      Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0          192.168.8.1    0.0.0.0        UG    0      0      0 wlan0
10.0.0.0         0.0.0.0        255.0.0.0      U      0      0      0 dummy0
10.0.0.1         0.0.0.0        255.255.255.255 UH     0      0      0 dummy0
10.129.37.0     0.0.0.0        255.255.255.0  U      0      0      0 eth0
169.254.0.0     0.0.0.0        255.255.0.0    U     1000    0      0 eth0
192.168.8.0     0.0.0.0        255.255.255.0  U      9      0      0 wlan0
192.168.15.0    0.0.0.0        255.255.255.0  U      0      0      0 dummy0
192.168.15.1    0.0.0.0        255.255.255.255 UH     0      0      0 dummy0
root@uml:~#

```

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

```
root@uml: ~
home@uml:~$ sudo -s
[sudo] password for home:
root@uml:~# echo 192.168.115.1 $HOSTNAME >> /etc/hosts
root@uml:~# modprobe dummy
root@uml:~# ip addr add 192.168.115.1/24 dev dummy0
root@uml:~# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.8.1    0.0.0.0         UG    0      0      0 wlan0
10.0.0.0          0.0.0.0        255.0.0.0       U     0      0      0 dummy0
18.0.0.1          0.0.0.0        255.255.255.255 UH    0      0      0 dummy0
10.129.37.0       0.0.0.0        255.255.255.0   U     0      0      0 eth0
169.254.0.0       0.0.0.0        255.255.0.0     U    1000   0      0 eth0
192.168.8.0       0.0.0.0        255.255.255.0   U     9      0      0 wlan0
192.168.15.0      0.0.0.0        255.255.255.0   U     0      0      0 dummy0
192.168.15.1      0.0.0.0        255.255.255.255 UH    0      0      0 dummy0
192.168.115.0     0.0.0.0        255.255.255.0   U     0      0      0 dummy0
```

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

```
home@uml: ~  
Router# show ip route 192.168.1.0  
Routing entry for 192.168.1.0/24  
  Known via "connected", distance 0, metric 1, best  
  * directly connected, wlan0  
  
Router# show ip route 10.129.37.0  
Routing entry for 10.129.37.0/24  
  Known via "connected", distance 0, metric 1, best  
  * directly connected, eth0  
  
Router# show ip route 127.0.0.0  
Routing entry for 127.0.0.0/8  
  Known via "connected", distance 0, metric 1, best  
  * directly connected, lo  
  
Router# show ip route 192.254.0.0  
Routing entry for 0.0.0.0/0  
  Known via "kernel", distance 0, metric 0, best  
  * 192.168.1.1, via wlan0  
  
Router# show ip route 0.0.0.0  
Routing entry for 0.0.0.0/0  
  Known via "kernel", distance 0, metric 0, best  
  * 192.168.1.1, via wlan0
```

FIGURE 3.16 – Configuration interface ethernet .

```
root@uml: ~
root@uml:~# ping -c 1 10.129.37.99
PING 10.129.37.99 (10.129.37.99) 56(84) bytes of data.
64 bytes from 10.129.37.99: icmp_seq=1 ttl=64 time=0.064 ms

--- 10.129.37.99 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.064/0.064/0.064/0.000 ms
root@uml:~# ping -c 10 10.129.36.243
PING 10.129.36.243 (10.129.36.243) 56(84) bytes of data.
From 10.129.37.201 icmp_seq=1 Destination Host Unreachable
From 10.129.37.201 icmp_seq=2 Destination Host Unreachable
From 10.129.37.201 icmp_seq=3 Destination Host Unreachable
From 10.129.37.201 icmp_seq=4 Destination Host Unreachable
From 10.129.37.201 icmp_seq=5 Destination Host Unreachable
From 10.129.37.201 icmp_seq=6 Destination Host Unreachable
From 10.129.37.201 icmp_seq=7 Destination Host Unreachable
From 10.129.37.201 icmp_seq=8 Destination Host Unreachable
From 10.129.37.201 icmp_seq=9 Destination Host Unreachable
From 10.129.37.201 icmp_seq=10 Destination Host Unreachable

--- 10.129.36.243 ping statistics ---
10 packets transmitted, 0 received, +10 errors, 100% packet loss, time 9047ms
pipe 3
root@uml:~#
```

FIGURE 3.17 – Connexion des machine virtuelles..

```
home@uml: ~
Connection closed by foreign host.
home@uml:~$ telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
Router> en
Password:
Router# show ip route connected
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 10.129.37.0/24 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.1.0/24 is directly connected, wlan0
Router#
```

FIGURE 3.18 – Configuration de table de routage R1.

```
home@uml: ~
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
Router> en
Password:
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 192.168.1.1, wlan0
C>* 10.129.37.0/24 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
K>* 169.254.0.0/16 is directly connected, eth0
C>* 192.168.1.0/24 is directly connected, wlan0
Router#
```

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>
```

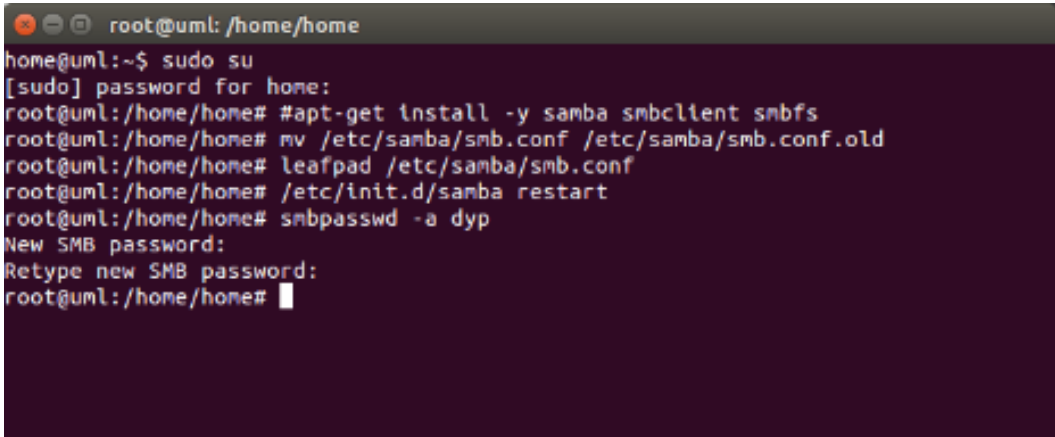
FIGURE 3.20 – Pinge de R2 depuis R1.

### 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 :

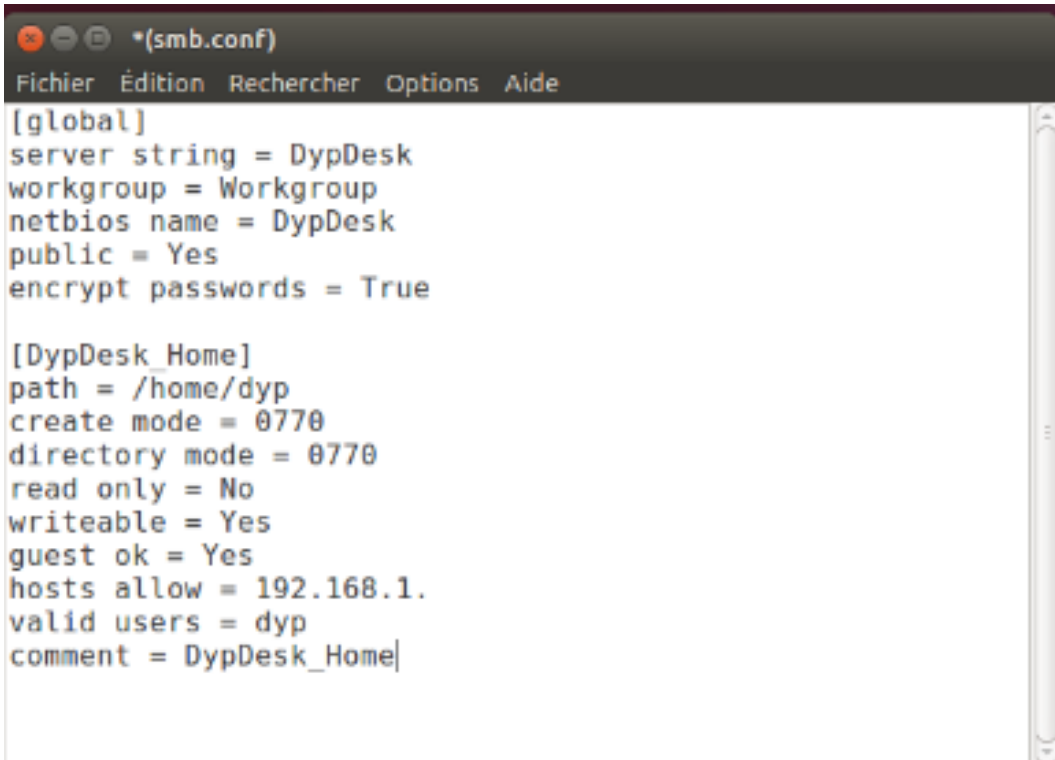
On va installer Samba ainsi que le client smbclient et le monteur de dossier smbfs et on Sauvegarde le fichier smb.conf en smb.conf.old afin de protéger le fichier original on va Créer le fichier de partage avec leafpad Ce fichier se décompose en deux parties : la section générale appelée "global" et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

A terminal window with a dark background and light text. The prompt is root@uml: /home/home. The user runs 'sudo su' and enters a password. Then they run '#apt-get install -y samba smbclient smbfs', 'mv /etc/samba/smb.conf /etc/samba/smb.conf.old', 'leafpad /etc/samba/smb.conf', and '/etc/init.d/samba restart'. Finally, they run 'smbpasswd -a dyp', enter a new password, and retype it.

```
root@uml: /home/home
home@uml:~$ sudo su
[sudo] password for home:
root@uml:/home/home# #apt-get install -y samba smbclient smbfs
root@uml:/home/home# mv /etc/samba/smb.conf /etc/samba/smb.conf.old
root@uml:/home/home# leafpad /etc/samba/smb.conf
root@uml:/home/home# /etc/init.d/samba restart
root@uml:/home/home# smbpasswd -a dyp
New SMB password:
Retype new SMB password:
root@uml:/home/home#
```

FIGURE 3.21 – Configuration PC1.

Fichier de partage avec leafpad



```
[global]
server string = DypDesk
workgroup = Workgroup
netbios name = DypDesk
public = Yes
encrypt passwords = True

[DypDesk_Home]
path = /home/dyp
create mode = 0770
directory mode = 0770
read only = No
writable = Yes
guest ok = Yes
hosts allow = 192.168.1.
valid users = dyp
comment = DypDesk_Home|
```

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

```
root@meriem-HP-Pavillon-g6-Notebook-PC: /home/meriem
meriem@meriem-HP-Pavillon-g6-Notebook-PC:~$ sudo su
[sudo] password for meriem:
root@meriem-HP-Pavillon-g6-Notebook-PC:/home/meriem# mousepad /etc/samba/smb.conf

(mousepad:2114): GtkSourceView-CRITICAL **: gtk_source_style_scheme_get_id: assertion 'GTK_SOURCE_IS_STYLE_SCHEME (scheme)' failed

(mousepad:2114): Glib-CRITICAL **: g_variant_new_string: assertion 'string != NULL' failed

(mousepad:2114): GtkSourceView-CRITICAL **: gtk_source_style_scheme_get_id: assertion 'GTK_SOURCE_IS_STYLE_SCHEME (scheme)' failed

(mousepad:2114): Glib-CRITICAL **: g_variant_new_string: assertion 'string != NULL' failed

(mousepad:2114): GtkSourceView-CRITICAL **: gtk_source_style_scheme_get_id: assertion 'GTK_SOURCE_IS_STYLE_SCHEME (scheme)' failed
root@meriem-HP-Pavillon-g6-Notebook-PC:/home/meriem#
```

FIGURE 3.23 – Configuration PC2.

```
smb.conf - Mousepad
Fichier  Édition  Rechercher  Affichage  Document  Aide
Attention, vous utilisez le compte root ; vous risquez d'endommager votre système.

[global]
server string = DypLap
workgroup = Workgroup
netbios name = DypLap
public = Yes
encrypt passwords = True

[DypDesk_Home]
path = /home/dyp
create mode = 0770
directory mode = 0770
read only = No
writeable = Yes
guest ok = Yes
hosts allow = 192.168.1.
valid users = dyp
comment = DypLap_Home
```

FIGURE 3.24 – Configuration fichier leafpad.

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

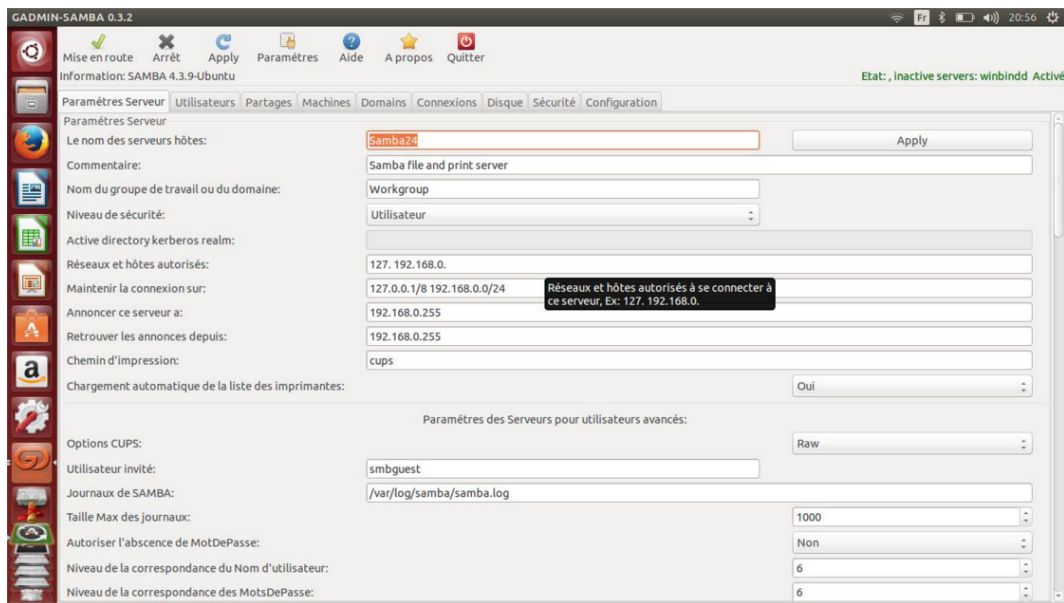


FIGURE 3.25 – Configuration pc1 avec Gsamba.

### 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

```
root@laghouatpc-TECRA-C50-B: ~
laghouatpc@laghouatpc-TECRA-C50-B:~$ sudo -s
[sudo] password for laghouatpc:
root@laghouatpc-TECRA-C50-B:~# #apt-get install uml-utilities
root@laghouatpc-TECRA-C50-B:~# tuncctl -g uml-net
Set 'tap1' persistent and owned by gid 125
root@laghouatpc-TECRA-C50-B:~# useradd sylvain uml-net
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
  -b, --base-dir BASE_DIR      base directory for the home directory of the
                               new account
  -c, --comment COMMENT        GECOS field of the new account
  -d, --home-dir HOME_DIR      home directory of the new account
  -D, --defaults                print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE expiration date of the new account
  -f, --inactive INACTIVE      password inactivity period of the new account
  -g, --gid GROUP              name or ID of the primary group of the
                               new account
  -G, --groups GROUPS          list of supplementary groups of the new
                               account
  -h, --help                    display this help message and exit
```

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.

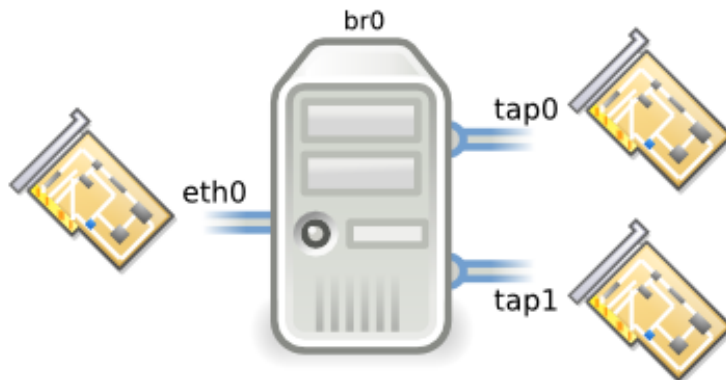


FIGURE 3.27 – Pont Linux.

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 255.255.255.0
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

```
laghouatpc@laghouatpc-TECRA-C50-B: ~  
laghouatpc@laghouatpc-TECRA-C50-B:~$ ifconfig eth0  
eth0      Link encap:Ethernet  HWaddr b8:6b:23:95:bb:84  
          UP BROADCAST MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
laghouatpc@laghouatpc-TECRA-C50-B:~$ ifconfig eth0 192.168.8.78 netmask 255.255.  
255.0 up  
SIOCSIFADDR: Operation not permitted  
SIOCSIFFLAGS: Operation not permitted  
SIOCSIFNETMASK: Operation not permitted  
SIOCSIFFLAGS: Operation not permitted  
laghouatpc@laghouatpc-TECRA-C50-B:~$ ifconfig eth0 192.168.8.78 netmask 255.255.  
255.0 up  
SIOCSIFADDR: Operation not permitted  
SIOCSIFFLAGS: Operation not permitted  
SIOCSIFNETMASK: Operation not permitted  
SIOCSIFFLAGS: Operation not permitted  
laghouatpc@laghouatpc-TECRA-C50-B:~$ #ifconfig eth0 192.168.8.78 netmask 255.255  
.255.0 up  
laghouatpc@laghouatpc-TECRA-C50-B:~$ ping -c 1 192.168.8.147  
PING 192.168.8.147 (192.168.8.147) 56(84) bytes of data.
```

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.

```
laghouatpc@laghouatpc-TECRA-C50-B: ~
laghouatpc@laghouatpc-TECRA-C50-B:~$ ping -c 1 192.168.8.147
PING 192.168.8.147 (192.168.8.147) 56(84) bytes of data.

--- 192.168.8.147 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

laghouatpc@laghouatpc-TECRA-C50-B:~$ ping -c 1 192.168.8.254
PING 192.168.8.254 (192.168.8.254) 56(84) bytes of data.

--- 192.168.8.254 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

laghouatpc@laghouatpc-TECRA-C50-B:~$ ping -c 1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.021 ms

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.021/0.021/0.021/0.000 ms
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:~$ ping -c 1 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
```

FIGURE 3.30 – Machine virtuelle accède au hôtel.

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

```
laghouatpc@laghouatpc-TECRA-C50-B: ~
console [tty0] enabled
Initializing software serial port version 1
console [mc-1] enabled
Failed to initialize ubd device 0 :Couldn't determine size of device's file
registered taskstats version 1
AppArmor: AppArmor sha1 policy hashing enabled
VFS: Mounted root (hostfs filesystem) readonly on device 0:12.
/bin/sh: 0: can't access tty; job control turned off
# cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
# VM TUN/TAP interfaces
auto tap0
iface tap0 inet manual
    pre-up tunctl -g uml-net -t tap0
    pre-up ifconfig tap0 up
    post-down tunctl -d tap0
# random: nonblocking pool is initialized
00
/bin/sh: 2: 00: not found
# ls -l /dev/net/tun
crw-rw-rw- 1 root root 10, 200 May 27 12:55 /dev/net/tun
#
```

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 deuxième c'est la création de réseaux entre deux machines à distance et accéder à ce 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 étudié 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](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 oût 2010] *premiers pas avec user mode linux*
- [12] [http://www.ijrst.com/images/short\\_pdf/1426506573 Lokesh Madan 2](http://www.ijrst.com/images/short_pdf/1426506573_Lokesh_Madan_2)
- [13] [https://fr.wikipedia.org/wiki/Noyau\\_de\\_syst](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](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/Procsfs>
- [19] <https://en.wikipedia.org/wiki/Sysfs>
- [20] [https://en.wikipedia.org/wiki/GNU\\_Zebra](https://en.wikipedia.org/wiki/GNU_Zebra)
- [21] <http://www.cisco.com/cisco/web/support/CA/fr/109/1091/10918591.html>
- [22] [https://en.wikipedia.org/wiki/Routing\\_Information\\_Protocol](https://en.wikipedia.org/wiki/Routing_Information_Protocol)
- [23] [http://www.chicoree.fr/w/Connecter\\_une\\_machine\\_virtuelle\\_UML](http://www.chicoree.fr/w/Connecter_une_machine_virtuelle_UML)

# Bibliographie