

# Rapport Projet 2CS

## Partie 02

Réalisation d'une distribution Linux réalisant la  
fonction principale de terminal intelligent dédiée  
pour l'enseignement

### Réalisé Par l'équipe 12 :

- HENNOUNI Narimane (CE)
- EL HASSANE Nour
- RAIHAH Amine
- AZIZ Rezak
- SEHDI Nassim

### Encadré Par :

1. AMROUCHE Hakim

# Table des matières

<b>Table des figures</b>	<b>4</b>
<b>Liste des tableaux</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
<b>1 Plan d'action et étude de l'outil LiveBuild</b>	<b>5</b>
1.1 Stratégie de la mise en place . . . . .	5
1.2 Live build plus approfondi . . . . .	5
1.2.1 Partie essentielle de Live Build . . . . .	5
1.2.2 Explication des commandes : . . . . .	6
1.2.3 Étapes de la construction de la distribution par Live Build : . . . . .	7
1.2.4 Personnalisations possibles : . . . . .	8
1.3 Mise en place de la distribution . . . . .	8
1.3.1 Installation des outils : . . . . .	8
1.3.2 Préparation de l'environnement : . . . . .	8
1.3.3 Vue d'ensemble de dossier de configuration : . . . . .	9
1.3.4 Étapes de la personnalisation : . . . . .	10
1.3.5 Génération des images ISO : . . . . .	14
1.3.6 Lancement de la distribution : . . . . .	16
1.4 Mise en place du serveur . . . . .	16
1.4.1 Installation des applications : . . . . .	16
1.4.2 Configuration des outils : . . . . .	17
<b>2 Performances et Tests</b>	<b>19</b>
2.1 Performances : . . . . .	19
2.2 Tests de performances : . . . . .	20
2.2.0.1 La commande "top" : . . . . .	20
2.2.0.2 La commande "vmstat" : . . . . .	22
2.2.0.3 La commande "ip" : . . . . .	23
2.3 Tests de l'architecture client-serveur : . . . . .	24

<b>3</b>	<b>Adaptation sur les architectures ARM (Machine Raspberry Pi)</b>	<b>28</b>
<b>4</b>	<b>Procédure de mise à jour de la distribution :</b>	<b>29</b>
4.1	Mise à jour des paquets : . . . . .	29
4.2	Mise à jour de la distribution entière . . . . .	30
<b>5</b>	<b>Bilan</b>	<b>31</b>
5.1	Partie gestion de projet : . . . . .	31
5.1.1	Organisation de l'équipe : . . . . .	31
5.1.2	Méthodologie suivie : . . . . .	31
5.2	Partie technique : . . . . .	32
5.2.1	Intérêt du projet : . . . . .	32
5.2.2	Problèmes rencontrés : . . . . .	32
5.2.3	Compétences acquises : . . . . .	32
	<b>Conclusion</b>	<b>33</b>

# Table des figures

1.1	Création dossier de travail . . . . .	8
1.2	Initialisation dossier de travail . . . . .	9
1.3	Vue de l'ensemble des dossiers de configuration . . . . .	9
1.4	Listes des packages . . . . .	10
1.5	Mirror Geogebra . . . . .	11
1.6	Clé Geogebra . . . . .	11
1.7	Personnalisation des contenus . . . . .	11
1.8	Squelette des utilisateurs . . . . .	12
1.9	Personnalisation de dossier usr . . . . .	12
1.10	Personnalisation de bootloader . . . . .	13
1.11	Personnalisation de l'installation . . . . .	13
1.12	Personnalisation de Kernel . . . . .	14
1.13	Suppression de certains packages . . . . .	14
1.14	Generation de la version 64 bits . . . . .	15
1.15	Génération de la version 32 bits . . . . .	15
1.16	Lykeio installé . . . . .	16
1.17	Script de création des utilisateurs . . . . .	18
2.1	Cmd Top : Aucune application n'est en exécution . . . . .	20
2.2	Cmd Top : "Remmina" en exécution . . . . .	21
2.3	Cmd Top : Toutes les applications en exécution . . . . .	21
2.4	Cmd vmstat : Aucune application n'est en exécution . . . . .	22
2.5	Cmd vmstat : Toutes les applications en exécution . . . . .	23
2.6	Cmd ip . . . . .	24
2.7	Demande Accès SSH . . . . .	25
2.8	Accès SSH avec Succes . . . . .	25
2.9	Demande Accès RDP . . . . .	25
2.10	Accès RDP avec Succès . . . . .	25
2.11	Configuration serveur VLC . . . . .	26
2.12	Configuration numéro de port . . . . .	26

2.13	Configuration de serveur streaming . . . . .	26
2.14	Accéder au serveur de streaming depuis client . . . . .	26
2.15	Accès VLC avec Succès . . . . .	26
2.16	Accès Moodle . . . . .	27
2.17	Accès distant à GeoGebra . . . . .	27
3.1	Commande pour générer ARM . . . . .	28
4.1	Contenu de sources.list . . . . .	30
4.2	Site Web Lykeio . . . . .	30

# Liste des tableaux

1.1 Arguments et explication de lb config . . . . . 6

# Introduction

À l’issue de la première partie, nous sommes parvenus à découvrir les différentes distributions existantes basées sur Debian et dédiées à l’enseignement, choisir un niveau scolaire qui est le “Lycée” et définir ses besoins, et enfin à faire une étude comparative entre les différents outils de réalisation existants sur le marché. Cette étude comparative nous a permis de choisir l’outil “**Live Build**” qui permet de répondre le plus aux exigences techniques ainsi qu’aux différents besoins fonctionnels et utilisateurs.

Dans cette 2ème partie du projet et dans un premier lieu, nous allons réaliser notre distribution proposée qui s’appelle “**Lykeio**” (Lycée en Grec) et qui est basée sur Debian, cette dernière doit être personnalisée le maximum possible et doit fonctionner sur les architectures amd64 (64-bits) ainsi que les architectures i386 (32-bits). Nous allons ensuite effectuer des tests en mettant en place une architecture client-serveur, les clients représentent les utilisateurs de notre distribution et le serveur répond à leurs besoins en offrant différents services. Ces tests vont nous permettre de mesurer les performances de notre distribution afin que notre client (l’établissement lycéen) puisse décider sur l’engagement avec notre entreprise “*DistroBuilders*”. On finit par une dernière partie qui présente la procédure de mise à jour de notre distribution, démontrer son adaptation sur les appareils ayant une architecture ARM comme dans le cas de la machine Raspberry Pi, et une conclusion.

# Chapitre 1

## Plan d'action et étude de l'outil LiveBuild

### 1.1 Stratégie de la mise en place

Après avoir effectué une étude bibliographique suivie d'une étude conceptuelle, nous avons été amenés à établir une stratégie pour la mise en place de notre distribution. L'architecture de déploiement est une architecture à base de terminal. Nous avons donc suivi la stratégie suivante :

1. La configuration de la distribution
2. La préparation du serveur central
3. L'installation de la distribution sur les terminaux

Comme il a été spécifié dans le cahier des charges, la distribution doit fonctionner sur les architectures 32-bits, 64-bits ainsi que les architectures ARM. La génération de l'image ISO pour ces dernières est possible grâce à l'outil Live Build et donc nous allons voir dans ce qui suit les détails de la création des 3 versions de notre distribution *“Lykeio”*.

### 1.2 Live build plus approfondi

#### 1.2.1 Partie essentielle de Live Build

Live build est composé de 3 outils : live-build, live-boot et live-config.

##### 1. Le paquet live-build :

Live build est une collection de scripts pour construire des systèmes live. L'idée derrière live-build est de constituer un cadre qui utilise un répertoire de configuration pour automatiser et personnaliser complètement tous les aspects de la construction d'une image Live.

Il utilise principalement 3 commandes en ligne de commandes : lb config, lb build, lb clean qui seront expliquées par la suite.

##### 2. le paquet live-boot :

live-boot est une collection de scripts fournissant des hooks pour initramfs-tools. Il est utilisé pour générer un initramfs capable de démarrer des systèmes live, comme ceux créés par live-build.



### 3. le paquet live-config :

live-config se compose des scripts qui s'exécutent au démarrage après live-boot pour configurer le système live automatiquement. Il gère les tâches telles que l'établissement du nom d'hôte, des paramètres régionaux et du fuseau horaire, la création de l'utilisateur live, l'inhibition des cron jobs et l'auto login de l'utilisateur live.

## 1.2.2 Explication des commandes :

Il utilise principalement 3 commandes en ligne de commandes : `lb config`, `lb build`, `lb clean` qui seront expliqués dans la suite.

### 1. La commande `lb config` :

Elle est responsable de l'initialisation d'un répertoire de configuration pour le système. Elle contient des options intéressantes pour spécifier des arguments ou des directives à la construction. Nous illustrons quelques arguments

TABLE 1.1 – Arguments et explication de `lb config`

Argument	explication	valeurs possible
<code>-mode</code>	choisir un mode global pour charger le projet	debian   ubuntu
<code>-architectures</code>	Définit l'architecture de l'image devant être construite. Par défaut, ceci est paramétré sur l'architecture hôte. Notez que vous ne pouvez pas construire une autre architecture si votre système hôte n'est pas capable d'exécuter nativement les binaires pour la distribution cible.	"amd64"   "amd32"   "arm" ...
<code>-distribution</code>	Définir la distribution du système live résultant	Exemple : le nom de code "buster" représente debian 10
<code>-apt-indices</code>	Permet de laisser les index apt, permet aussi d'économiser une bonne quantité d'espace, cependant la commande <b>apt-get update</b> doit être exécutée avant d'utiliser apt dans le système live	true   false
<code>-archive-areas</code>	Représente les principales divisions de l'archive. Dans Debian, ce sont <b>main</b> , <b>contrib</b> et <b>non-free</b> .	main   contrib   non-free   universe ...
<code>-security</code>	Permet d'utiliser les référentiels de sécurité spécifiés dans les options du miroir de sécurité.	true   false

<code>-updates</code>	Définit si les archives des paquets de mise à jour debian doivent être incluses dans l'image ou pas.	true   false
<code>-binary-images</code>	Définit le type d'image à construire. Par défaut, pour les images utilisant syslinux, ceci est paramétré pour iso-hybrid pour construire des images CD/DVD qui peuvent également être utilisées comme images hdd, pour les images non-syslinux, le défaut est iso.	iso   iso-hybrid   net-boot   tar   hdd
<code>-apt-recommends</code>	Définit si les paquets recommandés doivent être installés par apt.	true   false
<code>-debian-installer-gui</code>	Définit si l'interface graphique GTK de l'installateur-debian devrait être vraie ou pas. En mode Debian et pour la plupart des versions d'Ubuntu, cette option est vraie.	true   false
<code>-win32-loader</code>	Permet d'inclure ou pas win32 loader dans l'image résultante.	true   false
<code>-cache</code>	Définit globalement si un cache devrait être utilisé. Les différents caches peuvent être contrôlés à travers leurs propres options.	true   false

## 2. la commande `lb build` :

Elle est responsable de la construction de l'image. Cette commande lit les configurations depuis le répertoire initialisé et elle exécute les commandes de niveau inférieur si nécessaire

## 3. la commande `lb clean` :

Le rôle de la commande `lb clean` est d'enlever les différentes parties d'une construction afin que d'autres compilations ultérieures puissent commencer à partir d'un état propre.

### 1.2.3 Étapes de la construction de la distribution par Live Build :

Le processus de construction est divisé en étapes, avec des personnalisations différentes appliquées dans l'ordre dans chaque étape. La première étape à exécuter est l'étape bootstrap. C'est la phase initiale de peuplement du répertoire chroot avec des paquets pour faire un système Debian de base. Elle est suivie par l'étape chroot, qui complète la construction du répertoire chroot, le peuplant de tous les paquets listés dans la configuration, ainsi que tout autre matériel. La plupart de la personnalisation du contenu se produit à ce stade. La dernière étape de la préparation de l'image live est l'étape binary, qui construit une image amorçable, en utilisant le contenu du répertoire chroot pour construire le système de fichiers racine pour le système Live. Il comprend l'installateur et tout autre matériel supplémentaire sur le support cible en dehors du système de fichiers du système live.

### 1.2.4 Personnalisations possibles :

Live build offre une variété de possibilités pour optimiser le système sur tout les niveaux. On peut les classer selon les 4 points suivants :

1. **Personnalisation de l'installation des paquets** : La personnalisation la plus fondamentale d'un système est sans doute la sélection des paquets à inclure dans l'image. Grâce à cette personnalisation, on peut arriver même à injecter un noyau personnalisé qu'on a construit.
2. **Personnalisation des contenus** : cette personnalisation utilise les inclusions, les hooks et les preseeding. les inclusions permettent d'ajouter ou de remplacer des fichiers arbitraires dans l'image du système, les hooks permettent d'exécuter des commandes arbitraires dans différentes étapes de la construction et au démarrage et la préconfiguration (preseeding) permet de configurer les paquets quand ils sont installés en fournissant des réponses aux questions debconf.
3. **Personnalisation de l'image binaire** : live-build utilise syslinux et certains de ses dérivés comme chargeurs d'amorçage par défaut. Il est possible de les personnaliser selon les besoins.
4. **Personnalisation de l'installateur** : pour le modifier on utilise la technique des pré configurations et le remplacement des fichiers existants tel que les images et les logos.

## 1.3 Mise en place de la distribution

### 1.3.1 Installation des outils :

Live-build, Live-boot et Live config sont les paquets dont nous aurons besoin lors de la création de la distribution. Ces derniers sont disponibles sur l'entrepôt Debian et sont installés avec la commande :

```
# apt-get install live-build live-config live-boot.
```

### 1.3.2 Préparation de l'environnement :

Dans un premier temps, nous allons créer deux répertoires de travail : un pour la distribution 32 bits et l'autre pour la distribution 64 bits :

```
root@debian:~/debian# mkdir debian_destro
root@debian:~/debian# mkdir debian_destro32
```

FIGURE 1.1 – Création dossier de travail

Dans ce qui suit nous allons illustrer les étapes de configuration et les personnalisations dans le dossier `debian_destro` et les mêmes manipulations seront faites dans le dossier `debian_destro32`.

Nous allons donc nous mettre dans le dossier et nous exécutons les commandes comme suit :

```

root@debian:~/debian# cd debian_destro
root@debian:~/debian/debian_destro# lb config
[2021-06-15 11:55:32] lb config
P: Creating config tree for a debian/stretch/amd64 system
P: Symlinking hooks...
root@debian:~/debian/debian_destro# ls
auto  config  local
root@debian:~/debian/debian_destro# cp /usr/share/doc/live-build/examples/auto
/* ./auto/
root@debian:~/debian/debian_destro# █

```

FIGURE 1.2 – Initialisation dossier de travail

- `lb config` : est exécutée sans paramètres et cela pour initialiser le dossier de travail. Nous remarquons donc la création des dossiers déjà cités.
- ensuite nous avons copié les scripts par défaut de live-build depuis le dossier `auto` pour les modifier dans le dossier `local` et nous n'aurons donc pas écrit une longue commande pour `lb config`.

### 1.3.3 Vue d'ensemble de dossier de configuration :

On s'intéresse dans ce qui suit au dossier `config` qui est le dossier qui va contenir toutes les personnalisations et toutes les configurations. La commande `lb build` va s'appuyer sur ce dossier pour générer la distribution.

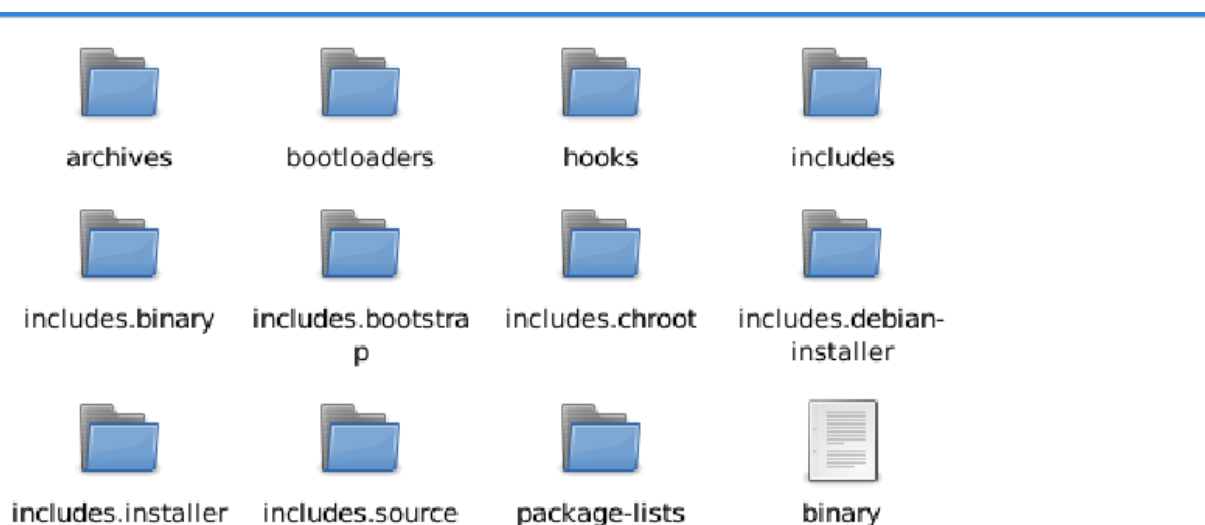


FIGURE 1.3 – Vue de l'ensemble des dossiers de configuration

- **archives** : sert à ajouter des mirrors avec les clés publique pour pouvoir profiter des paquets distribués sur des dépôts externes.
- **bootloaders** : il sert à configurer les différentes personnalisations de bootloader telles que les menu grub ou lilo, les images d'arrière plans, ...etc.
- **hooks** : les hooks sont une suite de scripts à exécuter par le système une fois installé. Grâce a son dossier on pourra faire toute les modifications qu'on souhaite mais d'autre alternatives plus simple peuvent être utilisé grâce aux autres dossiers.
- **includes.binary** : va contenir tous les fichiers qu'on veut inclure dans l'image iso telles que des vidéos d'utilisation, des manuels, des paquets,...

- **includes.bootstrap** : va contenir tous les fichiers qu'on veut inclure dans le bootstrap.
- **includes.chroot** : ce dossier contiendra l'arborescence de notre système de fichiers donc on pourra ajouter tout les fichiers qu'on voudrait qu'ils soient présents dans le système de fichiers. On peut ainsi ajouter des fichiers spécifiques ou des fichiers de personnalisation de l'utilisateur courant.
- **includes.debian-installer** : ce dossier sert à personnaliser l'installation de notre système.
- **includes.installer** : ce dossier sert à inclure des fichiers dans l'installateur tels que des fichiers d'automatisation d'installation ou autres.
- **package-lists** : ce dossier sert à lister les paquets qu'on veut installer dans notre système. C'est là que la personnalisation des paquets se fera.

Plusieurs dossiers pouvant être ajoutés selon les besoins.

### 1.3.4 Étapes de la personnalisation :

#### 1. Choix et intégration des paquets :

Pour cela on utilise le fichier `live.list.chroot` dans le dossier `packages-lists` comme suit :

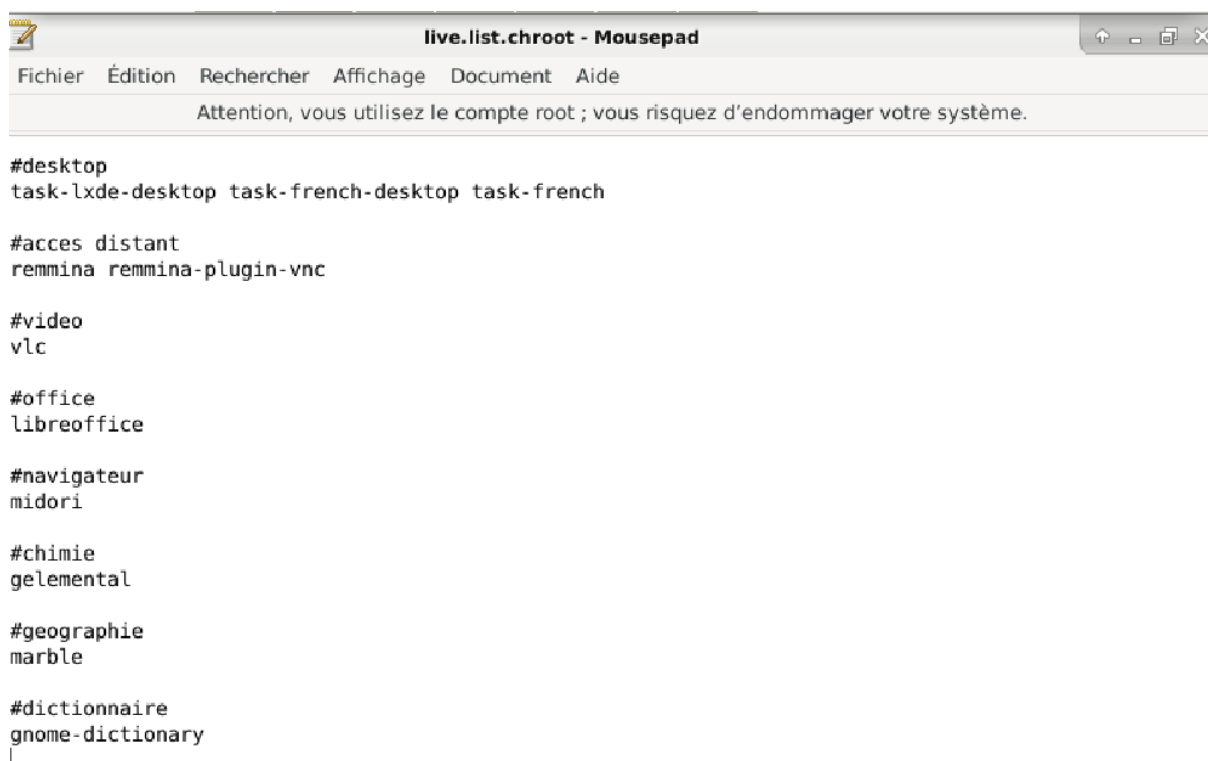


FIGURE 1.4 – Listes des packages

Une autre étape est nécessaire pour installer Geogebra est d'utiliser un dépôt externe. Cela peut se faire dans le dossier `archives`. Ce dernier va contenir deux fichiers un pour le dépôt et un autre pour la clé publique de ce dépôt :

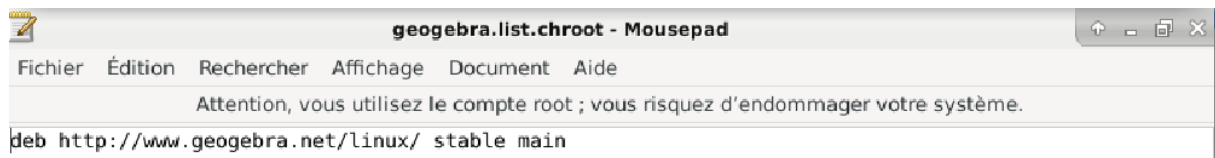


FIGURE 1.5 – Mirror Geogebra



FIGURE 1.6 – Clé Geogebra

## 2. Personnalisation des contenus :

Cette étapes se fait grâce au dossier includes.chroot : Dans notre cas nous avons utilisé 3 dossiers :



FIGURE 1.7 – Personnalisation des contenus

- **/etc** : dans ce dossier nous avons créé un dossier skel qui va servir de squelette pour la création d'autres utilisateurs. Il contient toutes les personnalisations qu'on a fait sur notre environnement bureau grâce aux fichiers de configuration respectif.

```
root@debian:~/debian_destro/config/includes.chroot# ls -a etc/skel/
.  ..  .config  Desktop
```

FIGURE 1.8 – Squelette des utilisateurs

Le dossier Desktop contient les fichiers qui s'afficheront sur le bureau tel que les raccourcis des applications ou autres. Cependant, le dossier .config est un dossier caché contenant les différents fichiers de configuration des applications et il peut être copié depuis une distribution qu'on a fini de personnaliser manuellement.

- **/root** : dans ce dossier nous avons appliqué le squelette créé précédemment.
- **/usr** : dans ce dossier nous avons utilisé les dossiers bin et share. Bin pour inclure un script que nous avons modifié pour la déconnexion d'un utilisateur (modifier la bannière et le texte affiché). Dans le dossier share, nous avons inclu des images pour le Desktop et les différents logos de notre distribution.

```
usr
├── bin
│   └── lxde-logout
├── share
│   ├── images
│   │   ├── desktop-base
│   │   │   ├── login-background.svg
│   │   │   └── splash.png
│   └── lxde
│       └── images
│           ├── logout-banner.png
│           └── lxde-icon.png
```

FIGURE 1.9 – Personnalisation de dossier usr

### 3. Personnalisation des bootloaders :

Pour cette étape nous avons utilisé le dossier bootloader. Nous avons donc modifié les fichiers de Grub et de démarrage pour mettre notre propre arrière plan pour le menu Grub et le menu de démarrage. Nous allons copier ce dossier depuis les fichiers de live-build et le modifier en remplaçant les images et les textes par défaut par nos personnalisations. Voici une illustration après avoir remplacé l'image d'arrière plan par notre image.

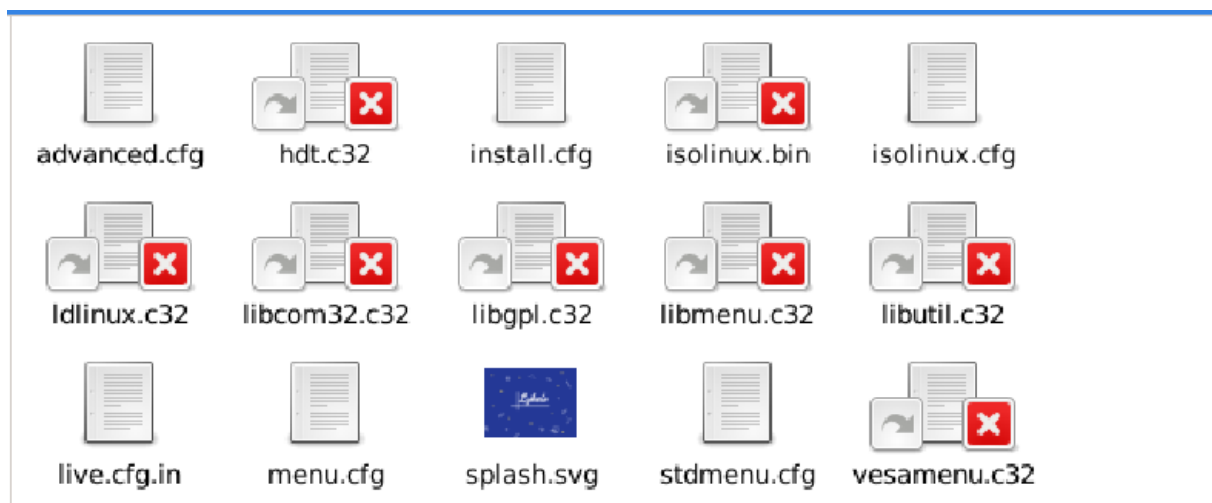


FIGURE 1.10 – Personnalisation de bootloader

#### 4. Personnalisation de l'installation :

L'installateur est modifié grâce au dossier `includes.debian-installer.` et `includes.installer.` Dans le premier dossier on modifie le logo affiché pendant l'installation et cela en ajoutant l'image comme suit :

- `config/includes.debian-installer/usr/share/graphics/logo_debian.png` (une bannière en PNG de 800x75 pixels).
- dans le dossier `includes.installer` nous allons mettre le fichier `preseed` pour automatiser l'installation. Pour effectuer cela, nous allons récupérer un fichier `preseed` générique depuis le site de Debian 10 puis nous l'avons adapté à notre cas

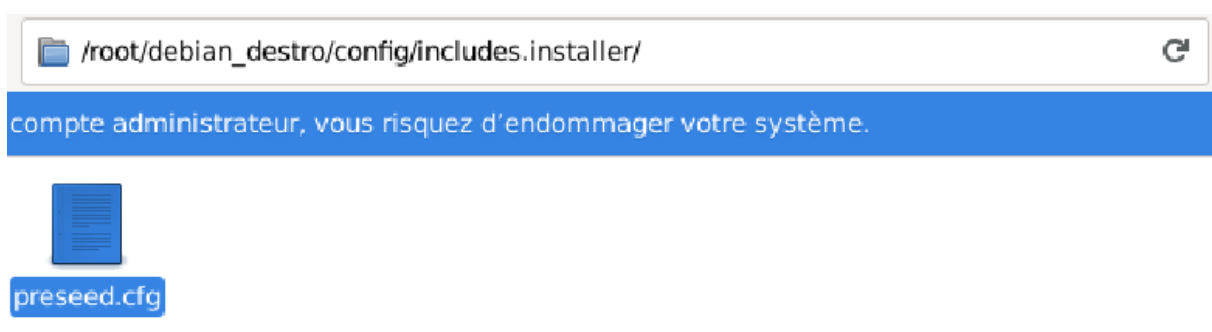


FIGURE 1.11 – Personnalisation de l'installation

#### 5. Personnalisation système :

Comme dernière modification qu'on a essayé de faire est d'optimiser le système en injectant un noyau linux personnalisé. Nous avons donc essayé de modifier un noyau en enlevant certaines fonctionnalités et drivers comme la virtualisation.

Nous avons téléchargé les sources de Kernel en exécutant la commande :

```
# apt-get install linux-source-4.19
$ tar xaf /usr/src/linux-source-4.19.tar.xz
$ cd linux-source-4.19
$ make menuconfig
```

et nous avons eu la fenêtre suivante :



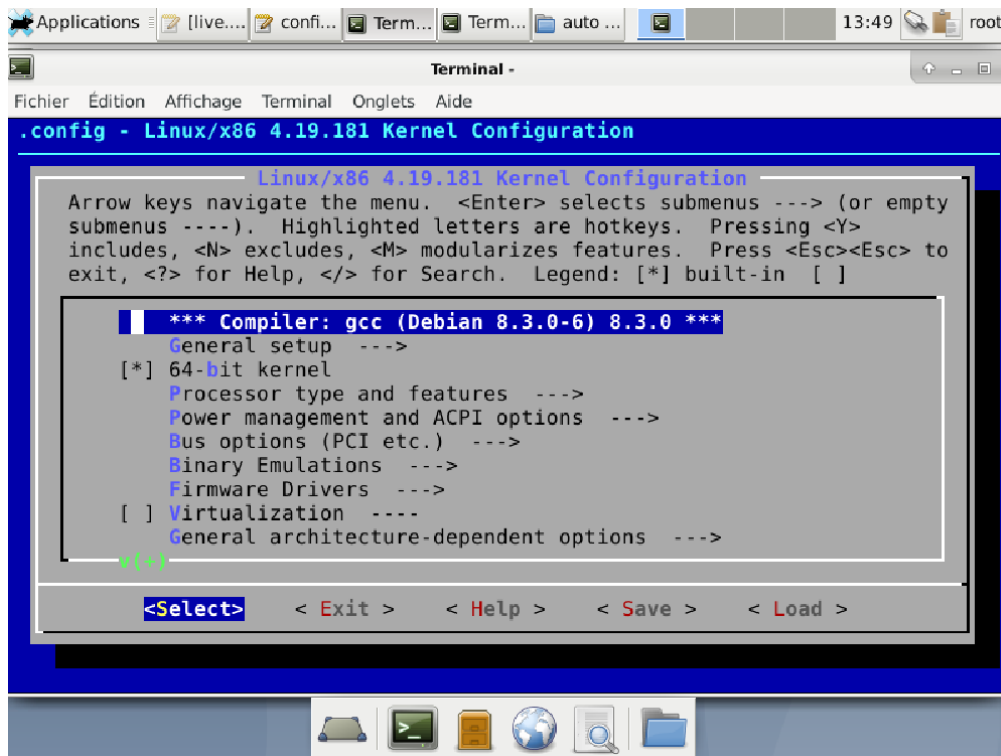


FIGURE 1.12 – Personnalisation de Kernel

Dans cette fenêtre on a indiqué les différents modules à inclure, on a fait *save* puis commencer la construction avec la commande : *make*. Cependant cette construction n'a pas abouti à cause du manque de ressources matérielles et de temps que la construction prends( après 4 heures un écroulement système s'est passé donc on a pas pu continuer la construction). Nous avons donc opté pour une autre méthode qui est la suppression de certains packages de drivers en utilisant des scripts.

```
apt remove bluez-firmware
apt remove vim-common
apt remove vim-tiny
```

FIGURE 1.13 – Suppression de certains packages

A partir de ce moment, on peut copier les mêmes configurations vers le dossier 32 bits et lancer la génération des images iso.

### 1.3.5 Génération des images ISO :

#### 1. Génération de la version 64-bits :

Afin de générer l'image ISO destinée aux processeurs AMD64, on modifie le fichier *auto/config* de la façon suivante pour ne pas avoir à écrire la même commande plusieurs fois.

```

lb config noauto \
--mode "debian" \
--architectures "amd64" \
--distribution "buster" \
--apt-indices "false" \
--archive-areas "main contrib non-free" \
--bootappend-live "locales=fr_FR.UTF-8 keyboard-layouts=fr" \
--bootappend-install "locales=fr_FR.UTF-8" \
--security "true" \
--updates "true" \
--binary-images "iso-hybrid" \
--apt-recommends "false" \
--debian-installer "live" \
--debian-installer-gui "true" \
--win32-loader "false" \
--cache "true" \
--clean \
--debug \
--verbose \
--source "false" \
    "${@}"

```

FIGURE 1.14 – Generation de la version 64 bits

Nous exécutons ensuite la commande : lb config et lb build.

## 2. Génération de la version 32-bits :

Afin de générer l'image ISO destinée aux processeurs 32 bits, on modifie le fichier auto/config en indiquant l'architecture i386 pour ne pas avoir à écrire la même commande plusieurs fois comme le montre la figure, ensuite on exécute lb config et lb build.

```

lb config noauto \
--mode "debian" \
--architectures "i386" \
--distribution "buster" \
--apt-indices "false" \
--archive-areas "main contrib non-free" \
--bootappend-live "locales=fr_FR.UTF-8 keyboard-layouts=fr" \
--bootappend-install "locales=fr_FR.UTF-8 keyboard-layouts=fr" \
--security "true" \
--updates "true" \
--binary-images "iso-hybrid" \
--apt-recommends "false" \
--debian-installer "live" \
--debian-installer-gui "true" \
--win32-loader "false" \
--cache "true" \
--clean \
--debug \
--verbose \
--source "false" \
    "${@}"

```

FIGURE 1.15 – Génération de la version 32 bits

Une fois le build est terminé on retrouve le fichier iso de l'image dans la racine de dossier de travail.

### 1.3.6 Lancement de la distribution :

L'installation de notre distribution se fait en utilisant l'image iso et suivant le manuel d'installation.

Une fois la distribution est installée on aura le bureau suivant.



FIGURE 1.16 – Lykeio installé

## 1.4 Mise en place du serveur

Comme les terminaux sont des terminaux légers qui fonctionnent à partir de ressources stockées sur un serveur central au lieu d'un disque dur local. Ainsi, pour pouvoir exploiter les différents services du serveur, ce dernier doit être proprement configuré et doté des différentes applications dont les utilisateurs auront besoin.

N'importe quel système serveur peut nous assurer la tâche. Dans notre cas nous avons mis en place un serveur sous Ubuntu sur lequel nous avons prévu les applications suivantes qui répondent aux exigences spécifiées dans la première partie : OpenSSH, VNCServer, VLC, xrdp, GeoGebra et moodle.

### 1.4.1 Installation des applications :

Pour mettre en place ces outils nous avons donc procéder comme suit :

#### 1. Installation de serveur SSH :

Dans la plupart des cas cet outil vient par défaut, sinon son installation se fait en utilisant la commande :

```
$ sudo apt-get install openssh
```

#### 2. Installation de serveur d'accès à distance :

Pour installer un serveur de bureau à distance tel que VNC on exécute la commande suivante :

```
$ sudo apt install tigervnc-standalone-server -y
$ sudo apt install xrdp
```

### 3. Installation de serveur video :

Pour le serveur video nous avons installé vlc puisqu'il assure le rôle de streaming. Nous exécutons donc la commande :

```
$ sudo apt install vlc
```

### 4. Installation de Moodle

Moodle est un système de gestion de cours gratuit et open-source basé sur PHP. Il permet aux tuteurs et aux institutions de créer des cours éducatifs pour leurs étudiants ou apprenants. Moodle est particulièrement utile pour les institutions à distance dans le monde entier pour fournir du matériel de formation à leurs apprenants. Les étapes de l'installation :

- **Étape 1** : Installer le serveur web Apache HTTP
- **Étape 2** : Installer MySQL et PHP
- **Étape 3** : Installation de logiciels supplémentaires
- **Étape 4** : Créer une base de données pour Moodle
- **Étape 5** : Télécharger la dernière version de Moodle
- **Étape 6** : Configurer le serveur HTTP Apache2
- **Étape 7** : configuration de Moodle via un navigateur web

### 5. Installation de Geogebra :

Geogebra est une application destinée à l'apprentissage et à l'enseignement des mathématiques et des sciences. Vu sa grande consommation d'espace d'une part, et nos contraintes et exigences techniques (principalement une distribution fonctionnelle aussi légère que possible) d'autre part, nous avons décidé de la mettre sur le serveur qui sera accessible via RDP.

## 1.4.2 Configuration des outils :

Après avoir installer les outils necessaire nous allons effectuer certaines configuration telque la configuration de serveur de streaming et la création des espaces utilisateurs.

#### 1. Configuration de VLC :

- Sur le serveur, il faut d'abord choisir l'option streaming puis ajouter les fichiers et les médias qu'on veut lancer pour le streaming.
- VLC offre la possibilité de faire un affichage local en simultané avec les machines clientes.
- Sur la machine client, on utilise le port requis, affiché par VLC après avoir sélectionné le protocole à utiliser, pour pouvoir accéder au streaming.

#### 2. Création des espaces utilisateurs :

Afin de séparer les espaces utilisateurs au sein de serveur nous avons crée un script qui permet de créer des utilisateurs à partir d'un fichier listusers.txt

```
GNU nano 3.2          ajout-utilisateur.sh          Modifié
#!/bin/bash
if [ $(id -u) -eq 0 ]; then
    input="./listusers.txt"
    while read -r line
    do
        IFS=','
        read -a ADDR <<<"$line"
        username="${ADDR[0]}"
        password="${ADDR[1]}"
        group="${ADDR[2]}"
        egrep "^$username" /etc/passwd > /dev/null
        if [ $? -eq 0 ]; then
            echo "$username exists"
            exit 1
        else
            pass=$(perl -e 'print crypt($ARGV[0],"password")' $password)
            useradd -m -p "$pass" "$username" --home /home/"$username" --create-home --$
            [ $? -eq 0 ] && echo "Utilisateur ajouté" || echo "echec pendant la creation"
        fi
    done < "$input"
else
    echo "Vous n'etes pas root"
    exit 2
fi
```

FIGURE 1.17 – Script de création des utilisateurs

### 3. Configuration de serveur d'accès à distance :

Il suffit de lancer un serveur VNC dans l'utilisateur concerné par l'accès à distance pour que cela fonctionne. On peut aussi utiliser xrdp au lieu de VNC

# Chapitre 2

## Performances et Tests

Les systèmes Linux sont connus pour être performants et peu gourmands en termes de capacité matérielle requise pour son bon fonctionnement. Notre cas (une distribution pour des lycéens), en est un cas exigeant. En effet, les caractéristiques matérielles dont dispose nos utilisateurs sont très faibles comparés à celles qui existent actuellement sur le marché. Des terminaux avec un processeur inférieur à i3, une RAM inférieure à 2GO et un disque de quelques centaines de GO.

### 2.1 Performances :

Afin de répondre à ces exigences, plusieurs métriques de performances sont étudiées afin de s'assurer du bon fonctionnement de notre distribution. Ces métriques vérifient plusieurs composants matériels et logiciels et permettent de juger la performance du système :

- **Charge CPU** : Il est essentiel de surveiller le CPU, la quantité de travail exécutée et depuis combien de temps il fonctionne car il peut atteindre un taux d'utilisation et une température élevés. Il peut avoir plusieurs cœurs, mais une application peut être dirigée vers un seul de ces cœurs, ce qui indique un comportement matériel dangereux.
- **Capacité du disque et IO** : La capacité du disque est particulièrement importante lorsqu'il s'agit de serveurs d'images et de fichiers, car elle peut affecter directement l'arrêt du système, corrompre le système d'exploitation ou provoquer une lenteur extrême de l'IO. Avec la surveillance des disques, il est possible de planifier un éventuel changement ou ajout de disque, et de vérifier le comportement d'un disque qui montre des signes de défaillance matérielle.
- **Réseau** : Lorsqu'il s'agit de RDP, DNS, DHCP, serveur de fichiers et proxy, il est extrêmement important de surveiller les performances du réseau en tant qu'entrée et sortie de paquets de données. Grâce aux journaux de performance du réseau, il est possible de mesurer l'utilisation de la carte, et créer un plan pour adapter l'application en fonction de l'utilisation du réseau.
- **Mémoire** : La surveillance de la mémoire dans d'autres composants permet de déterminer l'arrêt immédiat d'un système en raison d'un débordement de mémoire ou d'une mauvaise orientation pour une seule application.
- **Swap** : Il s'agit de la mémoire virtuelle créée par le système et allouée au disque pour être utilisée en cas de besoin. Son utilisation élevée peut indiquer que la quantité de mémoire du serveur est insuffisante.

## 2.2 Tests de performances :

Afin de réaliser les tests de performances, 2 machines virtuelles (1CPU, 1GB de RAM, 20GB de disque dur) ont été mises en place : une pour héberger la version 32-bits, et l'autre pour héberger 64-bits. Dans ce qui suit nous allons illustrer les résultats obtenus sur la machine 64bits. Les mêmes conclusions sont tirées à partir de la distribution 32bits

### 2.2.0.1 La commande "top" :

La commande "**top**" de Linux est un programme de surveillance des performances qui est utilisé fréquemment par de nombreux administrateurs système pour surveiller les performances de Linux et qui est disponible sous de nombreux systèmes d'exploitation de type Linux/Unix. La commande Top permet de classer tous les processus en cours d'exécution et actifs en temps réel dans une liste ordonnée et la met à jour régulièrement. Il affiche **l'utilisation du processeur, l'utilisation de la mémoire, la mémoire tampon, la taille du cache, la taille du tampon, le PID du processus, l'utilisateur, les commandes** et bien plus encore. Il montre également l'utilisation élevée de la mémoire et du processeur d'un processus en cours. On s'intéresse à deux cas en utilisant cette commande. Le cas où aucune application n'est lancée et le cas où toutes les applications sont lancées. Le but est de les comparer pour savoir la consommation de ces applications.

- **Aucune application en exécution :**

```
lykeio@lykeio: ~
File Edit Tabs Help

top - 15:09:20 up 1:42, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 88 total, 1 running, 87 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 987.4 total, 421.7 free, 211.1 used, 354.5 buff/cache
MiB Swap: 975.0 total, 975.0 free, 0.0 used, 627.4 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 361 root        20   0 258200 62068 35424 S   0.7   6.1   0:11.25 Xorg
2397 lykeio      20   0  8052   3200  2768 R   0.3   0.3   0:00.07 top
   1 root        20   0 23000 10204  7784 S   0.0   1.0   0:01.58 systemd
   2 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kthreadd
   3 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
   4 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
   6 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
   8 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
   9 root        20   0      0      0      0 S   0.0   0.0   0:00.10 ksoftirqd/0
  10 root        20   0      0      0      0 I   0.0   0.0   0:00.36 rcu_sched
  11 root        20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_bh
  12 root        rt    0      0      0      0 S   0.0   0.0   0:00.14 migration/0
  13 root        20   0      0      0      0 I   0.0   0.0   0:03.87 kworker/0:1-events
  14 root        20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
  15 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
  16 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 netns
  17 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kauditd
  18 root        20   0      0      0      0 S   0.0   0.0   0:00.00 khungtaskd
  19 root        20   0      0      0      0 S   0.0   0.0   0:00.00 oom_reaper
  20 root        0 -20      0      0      0 I   0.0   0.0   0:00.00 writeback
  21 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kcompactd0
  22 root        25   5      0      0      0 S   0.0   0.0   0:00.00 ksm
  23 root        39  19      0      0      0 S   0.0   0.0   0:00.15 khugepaged
```

FIGURE 2.1 – Cmd Top : Aucune application n'est en exécution

Lorsque aucune des applications n'est lancée, le taux d'utilisation du processeur et de la mémoire est **très faible** (%Cpu(s) :0.0 dans l'espace user (us) et 21.4% (211.8/987.4) de la mémoire utilisée), ce qui montre que le système d'exploitation seul consomme très peu de ressources en **mode repos**.

- **"Remmina" en exécution :**

```

lykeio@lykeio: ~
File Edit Tabs Help

top - 15:15:38 up 1:49, 2 users, load average: 0.01, 0.02, 0.00
Tasks: 92 total, 1 running, 91 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 987.4 total, 392.1 free, 237.1 used, 358.1 buff/cache
MiB Swap: 975.0 total, 975.0 free, 0.0 used, 600.5 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 361 root        20   0 262808 62592 35948 S   0.7   6.2   0:16.13 Xorg
  13 root         0   0      0      0      0 I   0.3   0.0   0:04.13 kworker/0:1-mm_percpu_wq
 239 root        20   0  22832  5436  3828 S   0.3   0.5   0:00.47 systemd-udevd
2376 lykeio      20   0 292696 33788 24092 S   0.3   3.3   0:01.69 lxterminal
2453 lykeio      20   0   8052   3116  2684 R   0.3   0.3   0:00.20 top
   1 root        20   0 104928 10356  7908 S   0.0   1.0   0:01.60 systemd
   2 root         0   0      0      0      0 S   0.0   0.0   0:00.00 kthreadd
   3 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
   4 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
   6 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
   8 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
   9 root        20   0      0      0      0 S   0.0   0.0   0:00.11 ksoftirqd/0
  10 root        20   0      0      0      0 I   0.0   0.0   0:00.44 rcu_sched
  11 root        20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_bh
  12 root        rt    0      0      0      0 S   0.0   0.0   0:00.15 migration/0
  14 root        20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
  15 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
  16 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 netns
  17 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kauditd
  18 root        20   0      0      0      0 S   0.0   0.0   0:00.00 khungtaskd
  19 root        20   0      0      0      0 S   0.0   0.0   0:00.00 oom_reaper
  20 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 writeback
  21 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kcompactd0

```

FIGURE 2.2 – Cmd Top : "Remmina" en exécution

On s'intéresse maintenant à **Remmina** qui permet l'accès à distance et on remarque que son lancement ne nécessite pas trop de charge CPU, par contre il utilise 26 Mib de mémoire (237-211) ce qui est satisfaisant.

- Toutes les applications en exécution :

```

lykeio@lykeio: ~
File Edit Tabs Help

top - 15:26:42 up 2:00, 2 users, load average: 0.08, 0.35, 0.20
Tasks: 100 total, 1 running, 99 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 97.3 id, 1.7 wa, 0.0 hi, 1.0 si, 0.0 st
MiB Mem : 987.4 total, 81.8 free, 576.6 used, 329.0 buff/cache
MiB Swap: 975.0 total, 945.8 free, 29.2 used, 255.9 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 239 root        20   0  22832  4256  2656 S   0.3   0.4   0:00.56 systemd-udevd
 361 root        20   0 291336 68336 42328 S   0.3   6.8   0:35.41 Xorg
 705 root        20   0 490292 30520 20240 S   0.3   3.0   0:02.10 lxpanel
2376 lykeio      20   0 293240 29820 19464 S   0.3   2.9   0:02.36 lxterminal
2875 lykeio      20   0 98.0g 258356 97860 S   0.3  25.6   0:08.26 WebKitWebProcess
3107 lykeio      20   0   8164   3324  2744 R   0.3   0.3   0:00.12 top
   1 root        20   0 104928  7328  5792 S   0.0   0.7   0:01.61 systemd
   2 root         0   0      0      0      0 S   0.0   0.0   0:00.00 kthreadd
   3 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
   4 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
   6 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
   8 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
   9 root        20   0      0      0      0 S   0.0   0.0   0:00.22 ksoftirqd/0
  10 root        20   0      0      0      0 I   0.0   0.0   0:00.65 rcu_sched
  11 root        20   0      0      0      0 I   0.0   0.0   0:00.00 rcu_bh
  12 root        rt    0      0      0      0 S   0.0   0.0   0:00.16 migration/0
  13 root        20   0      0      0      0 I   0.0   0.0   0:04.50 kworker/0:1-events
  14 root        20   0      0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
  15 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
  16 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 netns
  17 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kauditd
  18 root        20   0      0      0      0 S   0.0   0.0   0:00.00 khungtaskd
  19 root        20   0      0      0      0 S   0.0   0.0   0:00.00 oom_reaper

```

FIGURE 2.3 – Cmd Top : Toutes les applications en exécution

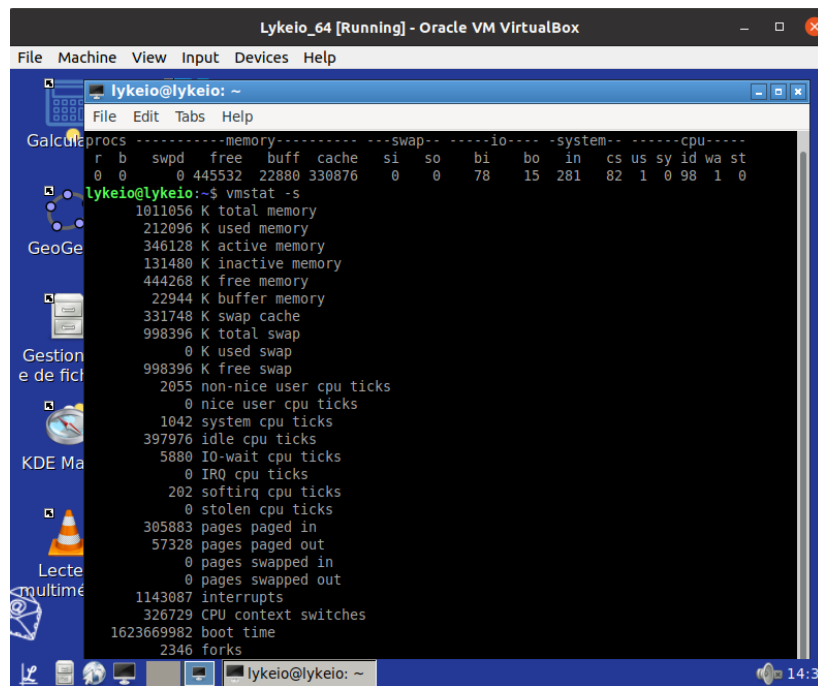
Maintenant, on lance toutes les applications utilisateurs et on voit à quel point le système d'exploitation peut les supporter. On voit bien que le taux d'utilisation du CPU est réparti entre les applications en exécution (0.3 pour chacune) et que trop de mémoire est allouée, par contre il reste toujours de l'espace donc on aura pas un risque d'arrêt du système.



On conclut donc que les 3 applications déjà mentionnées sont des **applications légères** qui ne nécessitent pas trop de ressources pour leurs bons fonctionnements (mise à part de Remmina).

### 2.2.0.2 La commande “vmstat” :

La commande Linux “vmstat” est utilisée pour afficher les **statistiques de la mémoire virtuelle, des threads kernel, des disques, des processus système, des blocs E/S, des interruptions, de l’activité du CPU** et bien plus encore. Pour tester les caractéristiques précédentes, nous avons utilisé les commandes suivantes : **vmstat** et **vmstat -s** : l’option **-s** affiche un résumé des divers compteurs d’événements et des statistiques de la mémoire.



```
Lykeio_64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

lykeio@lykeio: ~
File Edit Tabs Help

-----memory----- --swap-- -----io----- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 0 445532 22880 330876 0 0 78 15 281 82 1 0 98 1 0

lykeio@lykeio:~$ vmstat -s
1011056 K total memory
212096 K used memory
346128 K active memory
131480 K inactive memory
444268 K free memory
22944 K buffer memory
331748 K swap cache
998396 K total swap
0 K used swap
998396 K free swap
2055 non-nice user cpu ticks
0 nice user cpu ticks
1042 system cpu ticks
397976 idle cpu ticks
5880 IO-wait cpu ticks
0 IRQ cpu ticks
202 softirq cpu ticks
0 stolen cpu ticks
305883 pages paged in
57328 pages paged out
0 pages swapped in
0 pages swapped out
1143087 interrupts
326729 CPU context switches
1623669982 boot time
2346 forks
```

FIGURE 2.4 – Cmd vmstat : Aucune application n’est en exécution

En état de repos, le système ne reçoit pas trop d’opérations lectures/écritures depuis le disque (la zone swap n’est pas utilisée) et la mémoire libre est beaucoup plus grande par rapport à celle utilisée.

```

lykeio@lykeio: ~
File Edit Tabs Help
lykeio@lykeio:~$ vmstat
procs -----memory----- --swap-- -----io---- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 29908 83216 13592 323688 0 4 91 17 289 193 1 0 97 1 0
lykeio@lykeio:~$ vmstat -s
1011056 K total memory
590576 K used memory
461976 K active memory
372064 K inactive memory
83216 K free memory
13592 K buffer memory
323672 K swap cache
998396 K total swap
29908 K used swap
968488 K free swap
7730 non-nice user cpu ticks
0 nice user cpu ticks
2916 system cpu ticks
694760 idle cpu ticks
9659 IO-wait cpu ticks
0 IRQ cpu ticks
353 softirq cpu ticks
0 stolen cpu ticks
649663 pages paged in
123484 pages paged out
220 pages swapped in
7467 pages swapped out
2064685 interrupts
1380392 CPU context switches
1623669985 boot time
3161 forks

```

FIGURE 2.5 – Cmd vmstat : Toutes les applications en exécution

Lorsque toutes les applications sont ouvertes, l'utilisation de ces dernières devient massive tout en garantissant leurs bons fonctionnements et un espace libre de 83216 Kb.

### 2.2.0.3 La commande “ip” :

La commande “ip” est un outil réseau Linux destiné aux administrateurs système et réseau. Cet outil est utilisé pour configurer les interfaces réseau, voir les **statistiques de toutes les interfaces réseau (détails tels que les paquets transférés ou abandonnés, ou même les erreurs...etc)**. Ces statistiques vont nous aider à vérifier le bon fonctionnement du réseau dans lequel fait partie notre machine cliente.

```
LXTerminal
File Edit Tabs Help
root@lykeio:~# ip -s -s link ls
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped  overrun  mcast
         0         0         0         0         0         0
    RX errors: length  crc     frame  fifo    missed
               0         0         0         0         0
    TX: bytes  packets  errors  dropped  carrier  collsns
         0         0         0         0         0         0
    TX errors: aborted  fifo    window  heartbeat  transns
                  0         0         0         0         0
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
mode DEFAULT group default qlen 1000
    link/ether 08:00:27:fd:fd:22 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped  overrun  mcast
    996571    980         0         0         0         0
    RX errors: length  crc     frame  fifo    missed
                  0         0         0         0         0
    TX: bytes  packets  errors  dropped  carrier  collsns
    92804     698         0         0         0         0
    TX errors: aborted  fifo    window  heartbeat  transns
                  0         0         0         0         4
root@lykeio:~#
```

FIGURE 2.6 – Cmd ip

Les résultats de la commande montrent qu'il existe 2 interfaces réseau : une interface "Loop-back" qui est utilisée généralement pour les tests, on peut visualiser les différents paquets reçus, envoyés ou perdus par cette interface. Il existe aussi une 2ème interface qui est "enp0s3" et qui est en mode Broadcast, multicaste et UP (en marche) et a un MTU de 1500. Le nombre de paquets reçus et transmis est non nul, cela veut simplement dire que la carte réseau de la machine fonctionne bien mais il faut toujours la surveiller.

## 2.3 Tests de l'architecture client-serveur :

- Assurer l'accès à distance au serveur, à travers les protocoles RDP et SSH.

- **SSH :**

Pour se connecter au serveur en utilisant SSH, l'outil à utiliser est Remmina, il faut choisir le protocole SSH et faire rentrer l'adresse IP du serveur. Ensuite une demande d'authentification (nom et mot de passe de l'utilisateur) sera demandé pour que la connexion soit établie.

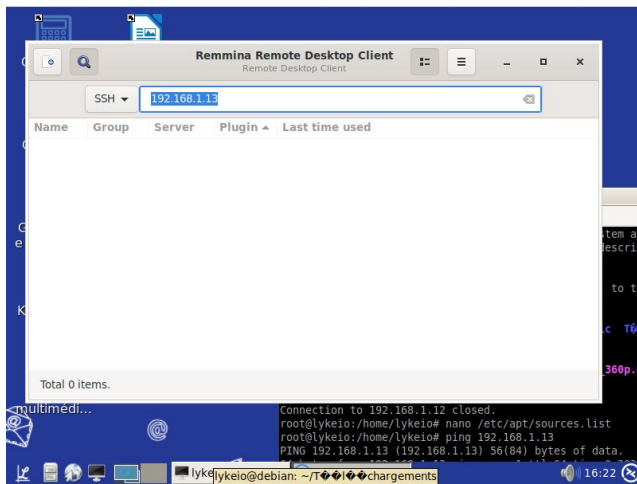


FIGURE 2.7 – Demande Accès SSH

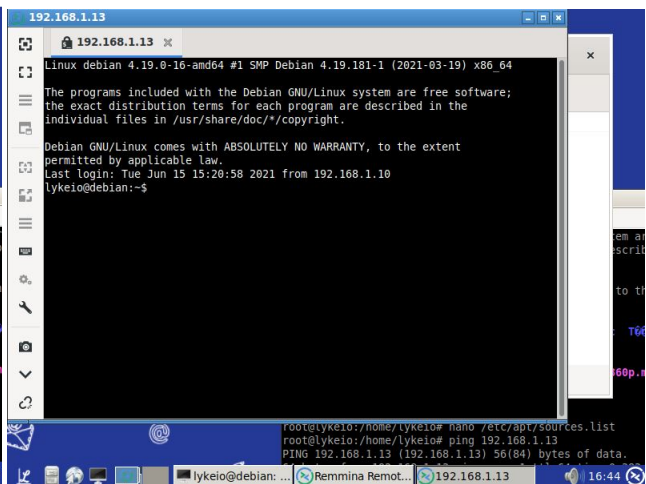


FIGURE 2.8 – Accès SSH avec Succès

### — RDP ou VNC :

Pour se connecter au serveur en utilisant RDP ou VNC, l'outil à utiliser est Remmina, il faut choisir le protocole RDP ou VNC et faire rentrer l'adresse IP du serveur. Ensuite une demande d'authentification (nom et mot de passe de l'utilisateur) sera demandé pour que la connexion soit établie :

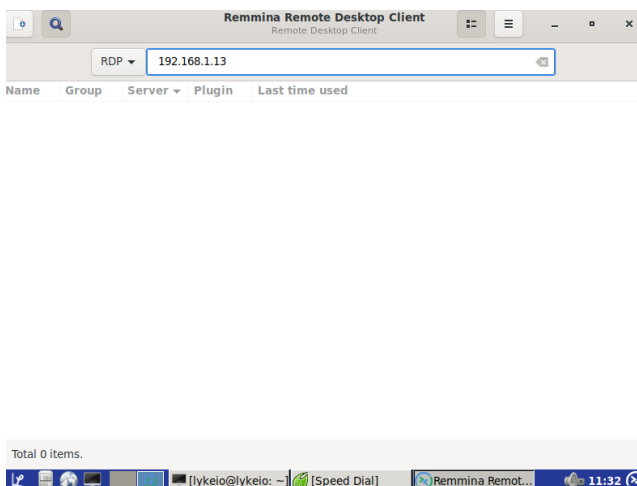


FIGURE 2.9 – Demande Accès RDP

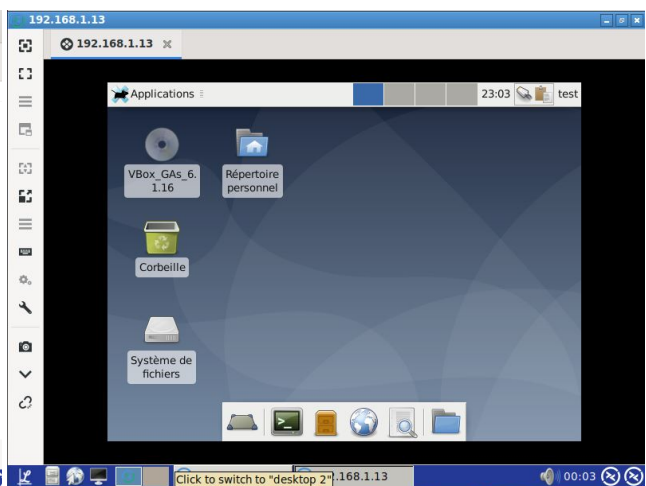


FIGURE 2.10 – Accès RDP avec Succès

### – Assurer l'accès au serveur vidéo :

Afin d'assurer l'accès au serveur vidéo, on doit d'abord lancer la diffusion depuis le serveur, pour cela on choisit l'option de **Stream** pour on rajoute les fichiers des vidéos qu'on veut lancer. On spécifie le protocole HTTP ainsi que le numéro de port 8080. Puis nous lançons la diffusion et nous aurons les résultats suivants :

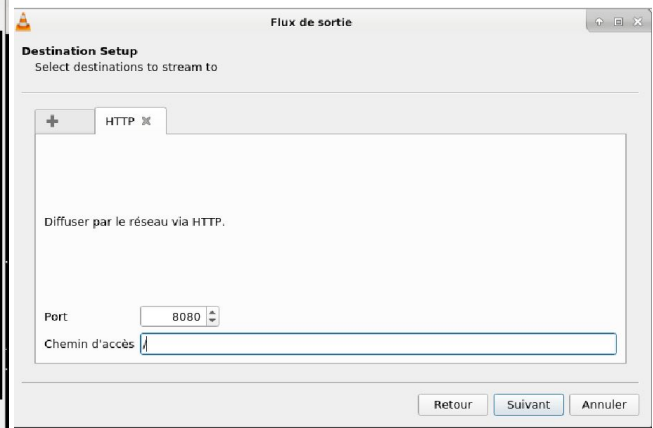
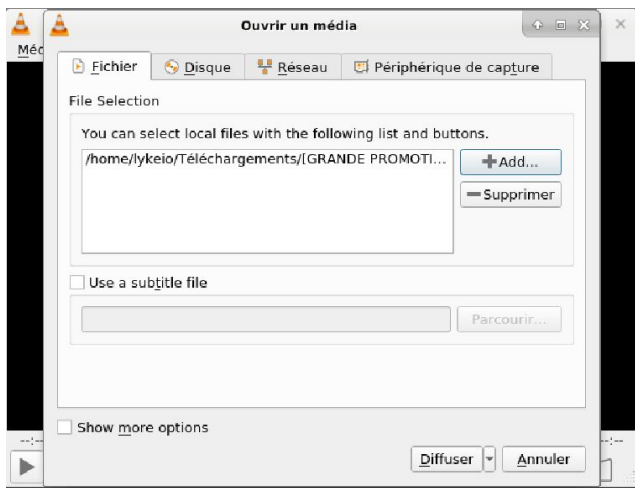


FIGURE 2.11 – Configuration serveur VLC

FIGURE 2.12 – Configuration numéro de port

FIGURE 2.13 – Configuration de serveur streaming

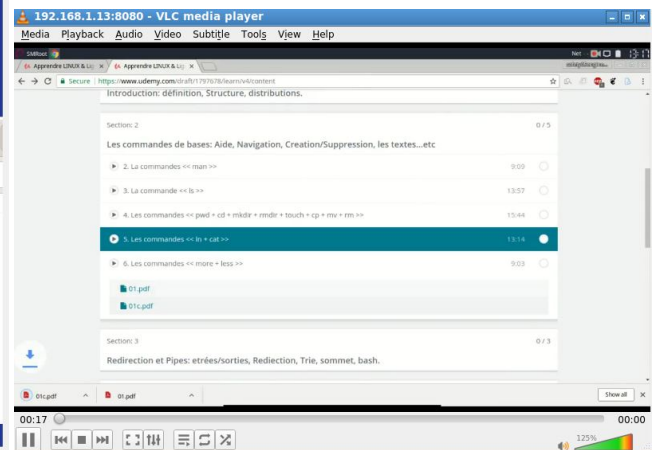
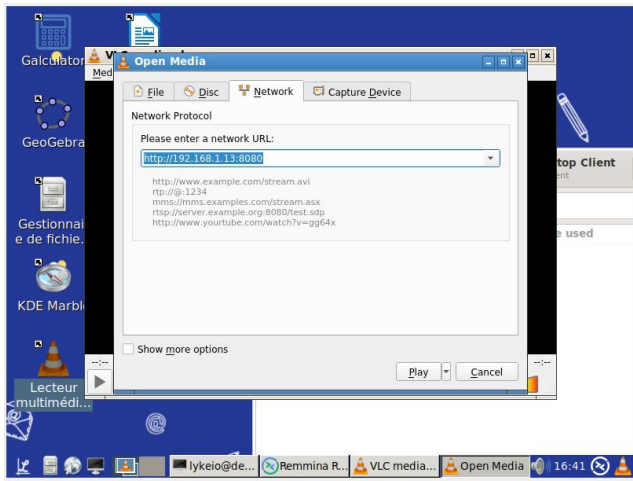


FIGURE 2.14 – Accéder au serveur de streaming depuis client

FIGURE 2.15 – Accès VLC avec Succès

- L'accès à Moodle :

Pour que les utilisateurs puissent suivre leur progrès dans les cours, remettre les devoirs et d'autres manipulations ils peuvent y accéder à leur propre compte Moodle, le serveur Moodle est hébergé sur notre serveur, et ceci se fait en introduisant dans l'URL du navigateur : **l'adresse IP du serveur/Moodle**

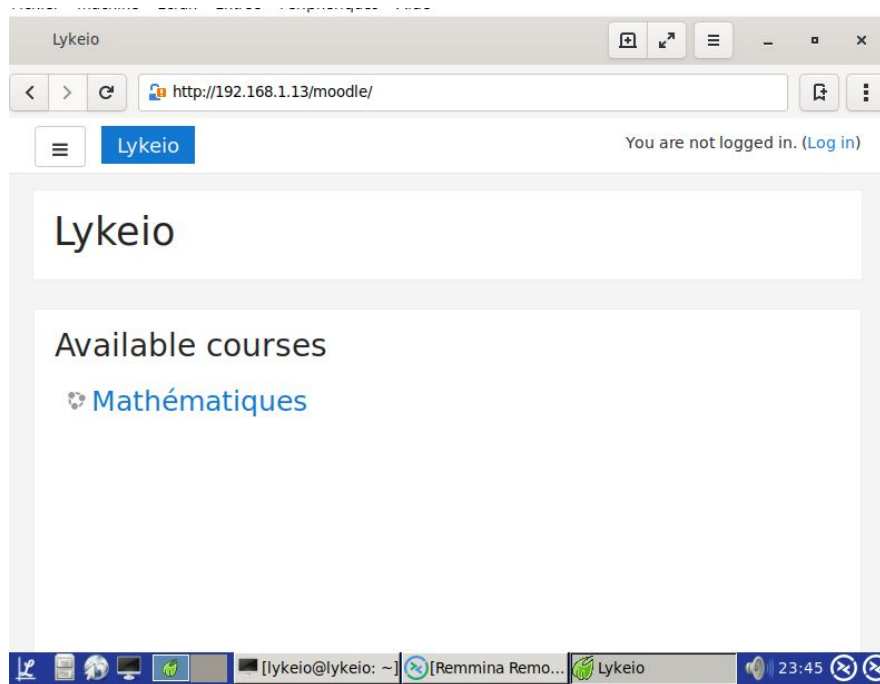


FIGURE 2.16 – Accès Moodle

– **L'accès à GeoGebra :**

GeoGebra est un outil mathématique très intéressant est utile pour les étudiant, mais vu sa grande consommation des ressources et les terminaux utilisateur sont légers, il est possible d'accéder à cette application installée sur le serveur et l'utiliser en utilisant un protocole d'accès distant (RDP ou VNC)

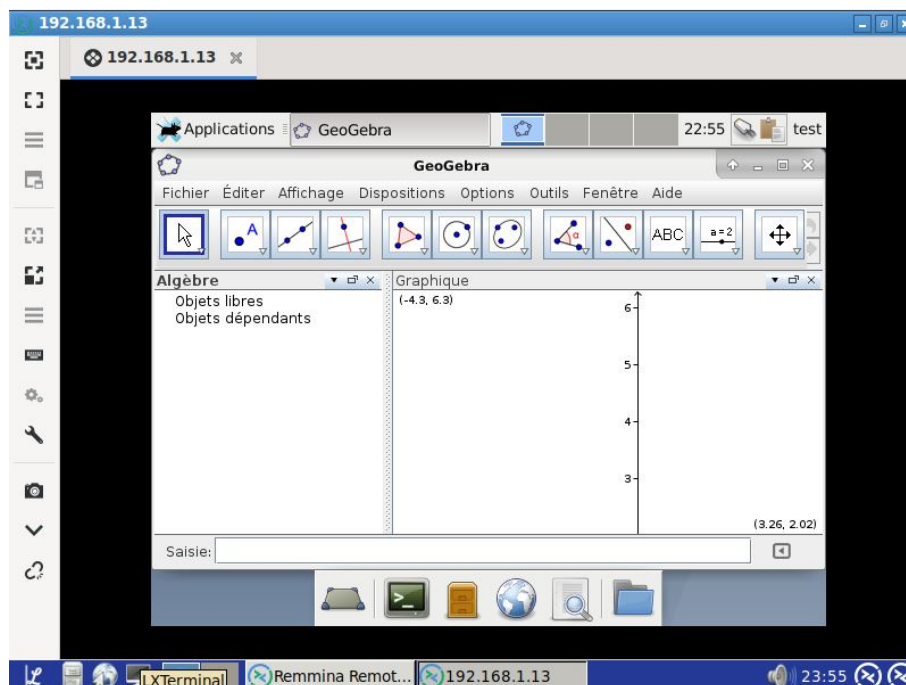


FIGURE 2.17 – Accès distant à GeoGebra

Toute autre application ou logiciel que alourdi le terminal et risque de faire baisser les performances peut être installé sur le serveur et utilisé à travers un accès distant via RDP ou VNC.

## Chapitre 3

# Adaptation sur les architectures ARM (Machine Raspberry Pi)

Le Raspberry Pi est un ordinateur de la taille d'une carte de crédit alimenté par le System On a Chip (SoC) **Broadcom BCM2835**. Ce SoC comprend un processeur **ARM1176JZFS** 32 bits, cadencé à 700MHz, et un GPU Videocore IV. Afin d'installer notre distribution sur cette machine, il était nécessaire de créer une nouvelle distribution similaire aux 2 précédentes, mais destinée aux architectures ARM. Ainsi, le travail sera fait sur une machine virtuelle à architecture "armhf" (Il existe des versions de Debian fonctionnant sur l'architecture ARM). Pour ce faire, il suffit de remplacer "amd64" par "armhf" dans la commande "lb config" comme le montre la figure suivante :

```
lb config noauto \
--mode "debian" \
--architectures "armhf" \
--distribution "buster" \
--apt-indices "false" \
--archive-areas "main contrib non-free" \
--bootappend-live "locales=fr_FR.UTF-8 keyboard-layouts=fr" \
--bootappend-install "locales=fr_FR.UTF-8" \
--security "true" \
--updates "true" \
--binary-images "iso-hybrid" \
--apt-recommends "false" \
--debian-installer "live" \
--debian-installer-gui "true" \
--win32-loader "false" \
--cache "true" \
--clean \
--debug \
--verbose \
--source "false" \
    "${@}"
```

FIGURE 3.1 – Commande pour générer ARM

Le reste de la configuration se fait de la même manière que la précédente. Il est à noter que cette distribution ne peut fonctionner que sur un environnement d'émulation tel que **QEMU**, ou bien entendu directement sur une machine **Raspberry** si disponible car les logiciels VirtualBox et VMware ne permettent d'exécuter que des machines virtuelles x86 sur un processeur x86.

# Chapitre 4

## Procédure de mise à jour de la distribution :

Dans le but de prévoir une façon de mettre à jour notre distribution, nous avons pensé à la façon de mettre à jour les paquets utilisés et la distribution toute entière.

### 4.1 Mise à jour des paquets :

Pour que les clients puissent faire les mises à jour nécessaires, nous avons choisi de créer un dépôt local hébergé sur le serveur depuis lequel les machines clientes peuvent y accéder.

#### 1. Définitions :

**Gestionnaire de paquets :** Un gestionnaire de paquet avancé comme apt ou yum gère des sources de logiciels et leur authenticité. Ces dernières sont placées au niveau du dépôt de paquet.

**APT :** APT est essentiellement une bibliothèque C++ de fonctions utilisées par plusieurs programmes de gestion de paquets.

**Sources APT :** Les sources à partir desquelles apt va chercher les paquets sont définies dans le fichier `/etc/apt/sources.list`

#### 2 Démarche pour créer un dépôt sur le serveur :

- Installation d'un serveur Web puis dans le dossier `/var/www/html` nous allons créer un répertoire "debian" qui contiendra toute l'arborescence du futur repository apt.
- Installation du paquet apt-mirror qui nous permettra de cloner (puis par la suite de synchroniser pour les mises à jour) le référentiel distant vers notre référentiel local.
- Configuration de apt-mirror et mise en miroir des référentiels distants dans le dossier local du système.

#### Mise à jour automatique :

Afin de maintenir le dépôt local à jour, on peut paramétrer sa mise à jour par l'intermédiaire d'une simple tâche **cron**. A chaque fois que notre dépôt local se synchronise avec le distant, s'il récupère de nouveaux paquets, ils sont automatiquement téléchargés, ainsi que leurs fichiers d'index.

#### Utilisation du dépôt :



Pour utiliser le dépôt nous devons configurer le fichier `sources.list` en remplaçant son contenu par les lignes suivantes où `ip` est l'adresse ip du serveur et sauvegarder le fichier.

```
deb [arch=amd64 trusted=yes] http://repository-local-  
ip/debian/mirror/ftp.fr.debian.org/debian/ buster main contrib non-free  
  
deb [arch=amd64 trusted=yes] http://repository-local-  
ip/debian/mirror/ftp.fr.debian.org/debian/ buster-updates main contrib non-free  
  
deb [arch=amd64 trusted=yes] http://repository-local-  
ip/debian/mirror/ftp.fr.debian.org/debian/ buster-proposed-updates main contrib non-free  
  
deb [arch=amd64 trusted=yes] http://repository-local-  
ip/debian/mirror/security.debian.org/debian-security buster/updates main contrib non-  
free  
  
deb [arch=amd64 trusted=yes] http://repository-local-  
ip/debian/mirror/ftp.fr.debian.org/debian/ buster-backports main contrib non-free
```

FIGURE 4.1 – Contenu de `sources.list`

Puis pour faire les mises à jour il faut simplement utiliser la commande **apt update**.

## 4.2 Mise à jour de la distribution entière

Pour mettre à jour de la distribution toute entière nous avons penser à créer un site Web sur lequel les clients peuvent télécharger les différentes release de notre distribution.

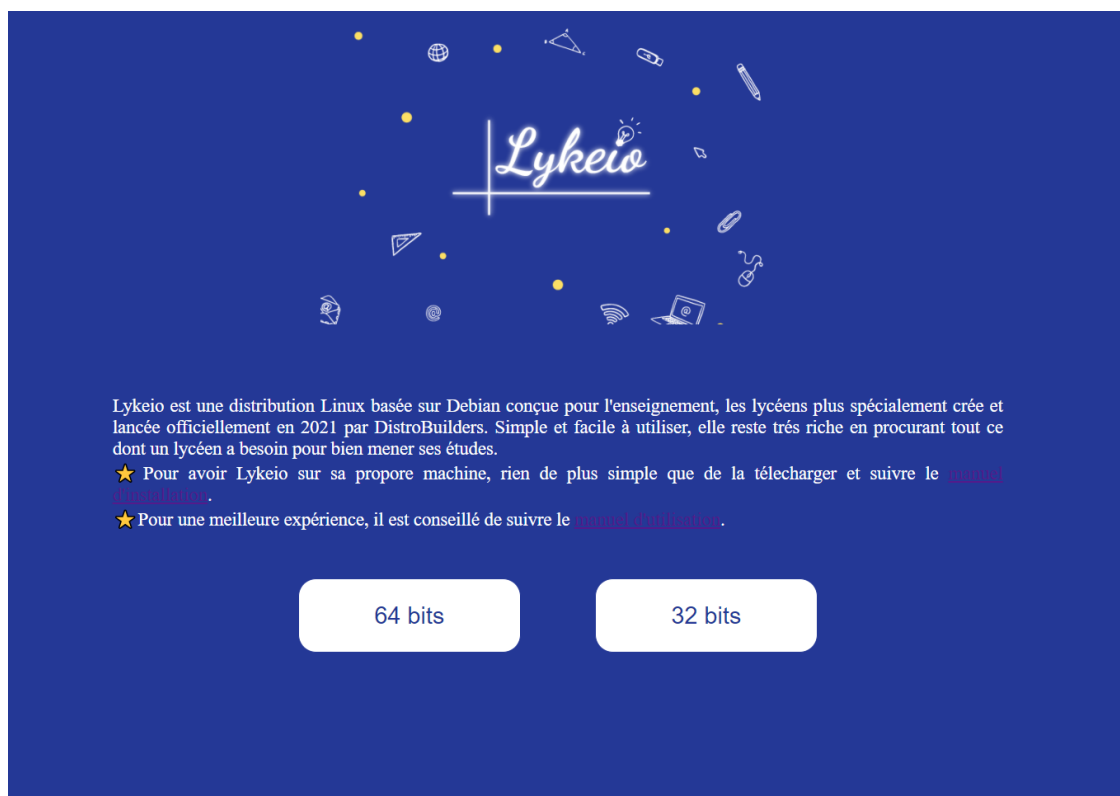


FIGURE 4.2 – Site Web Lykeio

# Chapitre 5

## Bilan

### 5.1 Partie gestion de projet :

#### 5.1.1 Organisation de l'équipe :

Lors de la réalisation de la partie technique du projet, notre équipe de travail "*DistroBuilders*" s'est divisée comme suit :

- **Conception de la distribution** : toute l'équipe.
- **Réalisation de la distribution** : Rezak.
- **Génération et installation des images ISO 64-bits & 32-bits** : Rezak, Amine et Nassim.
- **Tests de performance** : Nour
- **Tests de l'architecture client-serveur** : Narimane et Rezak
- **Préparation du manuel d'installation** : Amine
- **Préparation du manuel d'utilisation** : Nassim
- **Préparation des vidéos de formation** : Narimane et Nour
- **Évaluation du travail** : toute l'équipe.

#### 5.1.2 Méthodologie suivie :

Tout type de projet nécessite une méthodologie de travail à suivre afin d'atteindre ses objectifs. C'est pour cela qu'on a établi la méthodologie suivante :

- Étude comparative des outils de réalisation.
- Choix de l'outil et documentation approfondie.
- Définition et affectation des tâches.
- Conception de la distribution.
- Génération et installation de la distribution.
- Effectuer des tests de performances.
- Effectuer des différents tests de l'architecture client-serveur.
- Définition de la procédure de mise à jour ainsi que l'adaptation de la distro aux processeurs ARM.

- Préparation des manuels d’installation et d’utilisation.
- Préparation des vidéos pour la formation des utilisateurs.
- Évaluation du travail.

## **5.2 Partie technique :**

### **5.2.1 Intérêt du projet :**

Ce projet nous a permis d’utiliser toutes les connaissances acquises dans les fondamentaux des réseaux informatiques et les bases des systèmes d’exploitation tout au long de notre cursus, et chercher encore à atteindre un niveau plus avancé dans ces domaines afin de pouvoir créer sans difficultés une distribution Linux qui respecte des contraintes et des thématiques qui nous ont été exigées.

### **5.2.2 Problèmes rencontrés :**

- Confusions et ambiguïtés dans le cahier des charges.
- Manque de documentation de quelques outils de création de la distribution.
- Apparition des erreurs lors de l’installation de la distribution avec l’outil Live Build.
- Les ressources matérielles nécessaires pour la génération d’un Kernel créé à partir de 0. (c-à-d celui créé par notre équipe).
- Mise en place des protocoles RDP et SSH entre le client et le serveur.
- Installation de Moodle sur le serveur.

### **5.2.3 Compétences acquises :**

- Acquisition des bases du système Linux.
- Bonne connaissance sur le fonctionnement du Linux.
- Maîtrise de l’outil Live Build pour la création d’une distribution Linux basée sur Debian pour les architectures 64 et 32 bits.
- Maîtrise des protocoles RDP et SSH entre machine cliente et serveur.
- Respect des exigences et restrictions techniques pour la mise en œuvre de la distribution.
- Adaptation rapide aux différentes problématiques rencontrées.
- Maîtrise de l’outil Overleaf et Latex pour la rédaction des différents livrables : rapports et manuels d’utilisation et d’installations.
- Gestion de projet et travail en équipe tout au long du projet et en respectant les règles de conduite.

# Conclusion

À l'issue de ce projet qui consiste à réaliser une distribution Linux basée sur Debian réalisant la fonction principale d'un terminal intelligent dédiée pour l'enseignement, nous nous sommes investis pour réaliser cette solution qui est une solution meilleure, moins coûteuse et économique. En effet, elle est légère en termes de ressources, c'est-à-dire ne demande pas beaucoup de mémoire et de ressources processeur, ce qui va aider les écoles à migrer leur parc informatique ancien en une architecture basée sur des terminaux Linux. Ce n'est pas seulement que ça, mais elle permet aussi de répondre parfaitement aux besoins techniques, fonctionnels et utilisateurs grâce au système d'exploitation "Debian" qui est Open Source et aux différentes personnalisations qu'on peut appliquer à l'aide de l'outil de réalisation qui est "Live Build" pour notre cas.

Cette distribution est installée par la suite sur des terminaux ou bien des machines spéciales (systèmes embarqués tel que Raspberry Pi) qui se connectent à un serveur central pour accéder aux données et exécuter les applications. Cependant, il fallait faire plusieurs tests avant le déploiement de cette dernière pour démontrer ses performances sur les différents niveaux : capacité de calcul, stockage, réseau...etc.

Ce projet nous a permis de mettre en pratique les connaissances acquises durant notre cycle supérieure à travers un cas d'étude proche de la réalité (environnement d'entreprise), travailler en équipe tout en respectant les règles de conduite de projet et surtout délivrer un produit de qualité avec un meilleur coût et dans les plus brefs délais.

# Bibliographie

- [1] <https://www.cloudflare.com/fr-fr/learning/access-management/what-is-the-remote-desktop-protocol/>.
- [2] <https://doc.ubuntu-fr.org/ssh>.
- [3] <https://cloudcone.com/docs/article/how-to-install-moodle-on-ubuntu-18-04/>.
- [4] <https://smallbusiness.chron.com/set-up-vlc-server-26983.html>.
- [5] <https://www.tecmint.com/command-line-tools-to-monitor-linux-performance/4>.
- [6] <https://www.tecmint.com/linux-performance-monitoring-with-vmstat-and-iostat-command>.
- [7] <https://phoenixnap.com/kb/linux-ip-command-examples#:~:text=The%20ip%20command%20is%20a,ifconfig%20command%2C%20which%20operates%20similarly..>
- [8] <https://www.it-connect.fr/debian-comment-creer-son-propre-repository-local/>.
- [9] <https://live-team.pages.debian.net/live-manual/html/live-manual/index.en.html>.
- [10] <https://debian-handbook.info/browse/stable/sect.kernel-compilation.html>.
- [11] <https://wiki.ubuntu.com/Live-Build>.
- [12] <https://phoenixnap.com/kb/build-linux-kernel>.
- [13] Maurice J. Bach. *The design of the UNIX operationg system*. Pearson Education, 1986.
- [14] Karim Yaghmour Jon Masters Gilad Ben-Yossef and Philippe Gerum. *Building embedded Linux systems*. O'reilly, 2008.
- [15] Patrick Cegielski. *Conception de systeme d'exploitation, le cas Linux*. Eyrolles, 2004.
- [16] Pierre Ficheux. *Linux Embarqué 2eme édition*. Eyrolles, 2002.
- [17] P. Raghavan Amol Lad Sriram Neelakandan. *Embedded Linux system design and development*. Auerbach Publications, 2006.