

IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex

Département informatique

Rapport TD Outils-Libres en Latex

Etudiant : Enzo Collot
Année universitaire 2022–2023

Table des matières

1 Efficacité de l'environnement de travail

1.1 TD-1 : La souris

— Désactiver votre souris au niveau système avec la commande `xinput`

Pour désactiver la souris au niveau système, nous allons utiliser la commande `xinput`.

Je tiens à préciser que je vais effectuer ce test sur mon ordinateur personnel qui fonctionne sous Linux Mint.

Voici ci-dessous, la commande qui permet de désactiver la souris :

```
xinput set-prop "device name" "Device Enabled" 0
```

Dans mon cas, j'ai dû rentrer la commande suivante :

```
xinput set-prop "Logitech Wireless Receiver Mouse" "Device Enabled" 1
```

— Initialiser un fichier dans lequel nous allons lister tous les problèmes d'efficacité rencontrés pendant cette séance.

Priorité	Problème	Correctif
1	Logout difficile au clavier	Raccourci clavier Ctrl+Alt+Suppr/Delete
2	Impossible d'éditer des documents PDF avec Google Drive	Utilisation de LaTeX
3	La souris est bloquée et ne répond plus	Utilisez le raccourci clavier Ctrl+Alt+F1 pour ouvrir une console en mode texte, puis connectez-vous à votre session. Ensuite, utilisez la commande sudo service gdm3 restart pour redémarrer le serveur d'affichage et réinitialiser la souris
4	Impossible d'avoir plusieurs terminaux en parallèle	Sous Terminator, Ctrl+Shift+O pour split le terminal verticalement, Ctrl+Shift+L pour split le terminal horizontalement
5	Accéder au navigateur de fichier	shortcut configurable dans les settings ex : Alt+F
6	Naviguer dans discord sans la souris	Tab pour se déplacer de haut en bas, Shift+Tab pour aller de bas en haut et Ctrl+Tab pour naviguer de gauche à droite
7	La souris ne fonctionne pas sur un ordinateur portable	Utilisez le raccourci clavier Fn+F9 pour activer ou désactiver le pavé tactile, qui peut parfois interférer avec la souris

1.2 TD-2 : Le clavier

— Identifier un site permettant de s'améliorer à taper au clavier.

J'ai trouvé un site qui permet de tester la rapidité de frappe au clavier. Voici le lien ci-dessous :

Site de TapTouch

Pourquoi celui-ci plutôt qu'un autre ?

J'ai choisie ce site car il explique quelques informations sur notre score à la fin et il nous donne des astuces pour nous améliorer.

Inclure quelques screenshots montrant l'interface

Screen de l'interface du site :

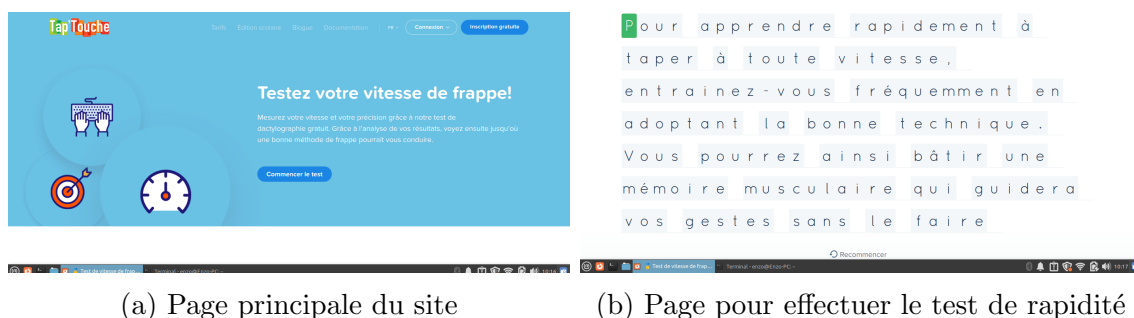


FIGURE 1 – Deux captures d'écran du site TapTouch

Essayer de masquer ses mains pour taper sans regarder

Inclure quelques résultats de rapidité.

Premier test effectué le 22/12/2022 :



Deuxième test effectué le 23/12/2022 :



1.3 TD-3 : Vim et Emacs

- Effectuer les tutoriels pour Vim et Emacs :

Vim : `vimtutor`

Emacs : `emacs`, puis `Ctrl+H+T`

- Paramétrer GNU Readline (le line editor par défaut de bash) pour être en mode Emacs ou Vim

Une fois que j'ai effectué les deux tutoriels, je vais paramétrer GNU Readline en mode Vim. Pour ce faire, je vais utiliser la commande suivante :

```
export EDITOR=vim
```

Après, pour l'avoir de façon permanente, on peut ajouter cette ligne dans le fichier `/.bashrc`.

- Tester les deux 2 modes, en choisir un

J'ai testé ces deux modes et j'ai choisie d'utiliser Vim.

- Paramétrer Emacs ou Vim comme étant éditeur par défaut en plus du mode Readline.

Pour qu'il soit de façon permanente, j'ai ajouté cette ligne suivante dans le fichier `bashrc` comme ci-dessous :

```
export EDITOR=vim
```

1.4 TD-4 : Commande history

— Regarder votre history

Voici ci-dessous, une capture d'écran de mon history :

```
125 cd Documents/cours_serveurs_web/
126 ls
127 cd apache2/
128 ls
129 vim Vagrantfile
130 vagrant up
131 vagrant ssh
132 vagrant halt
133 cd nginx/
134 vim Vagrantfile
135 vagrant up
136 vagrant ssh
137 vagrant halt
138 xinput
139 xinput set-prop "Virtual core XTEST pointer" "Device Enabled" 0
140 xinput
141 xinput set-prop "Logitech Wireless Receiver Mouse" "Device Enabled" 0
142 xinput set-prop "Logitech Wireless Receiver Mouse" "Device Enabled" 1
143 vim
144 export EDITOR=vim
145 ls
146 sudo vim ~/.bashrc
147 history
```

— Y a-t-il des informations sensible ? Comment y remédier ?

Oui, il peut y avoir des informations sensibles dans le history comme les mot de passe etc. Pour remédier à cela, nous pouvons utiliser un autre interpréteur de commande comme ZSH.

— Un employé de longue date semble avoir un **history** très court, quelle sont les causes possibles ?

```
cat ~/.bash_history
```

```
sudo apt install curl zsh git
```

```
sh -c "(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

On peut constater que cette employé à utiliser un autre interpréteur de commande qui est ZSH en le combinant avec OHMYZSH. Donc on ne peut plus voir son history dans `bash_history`.

— Nos history sont souvent pollués par beaucoup de `ls`, `cd`, `pwd` ... Ces commandes sont tellement courtes que les taper entièrement est plus rapide. On veut éviter qu'elles n'apparaissent dans les résultats de recherche de l'history, comment faire ?

Pour éviter d'avoir certaine commande dans notre history, on peut ajouter l'option `HISTIGNORE` dans le fichier `bashrc` :

```
export HISTIGNORE="&:ls:cd:pwd"
```

Voici le resultat ci-dessous :

```
zeyrox@Zeyrox-PC:~$ cd
zeyrox@Zeyrox-PC:~$ ld
ld: no input files
zeyrox@Zeyrox-PC:~$ pwd
/home/zeyrox
zeyrox@Zeyrox-PC:~$
```

(a) Quelques commandes pour tester HISTIGNORE

```
73  cd
74  history
zeyrox@Zeyrox-PC:~$
```

(b) Résultat dans history

FIGURE 2 – Deux captures d’écran montrant un test de HISTIGNORE

1.5 TD-5 : Alias

— Il est possible de créer des alias avec des arguments à l’aide des bash functions :

Ecrire une bash function mkcd mon dossier qui crée le dossier mon dossier puis navigue dedans.

Voici les étapes ci-dessous pour créer une bash function mkcd :

```
enzo@Enzo-PC:~$ function mkcd() { mkdir -p "$@" && cd "$_"; }
enzo@Enzo-PC:~$ alias mkcd="mkcd"
enzo@Enzo-PC:~$ mkcd test
enzo@Enzo-PC:~/test$
```

Ecrire une bash function gitemergency qui add, commit, et push tout son travail sur l’origine, permettant de ne rien perdre en cas d’alerte d’incendie.

Voici les étapes ci-dessous pour créer une bash function gitemergency :

```
enzo@Enzo-PC:~$ function gitemergency() { git add .; git commit -m "Commit rapide"; git push -u origin; }
```


1.6 TD-8 : Raccourci ZSH

- Dans zsh, créer un raccourci **Ctrl + Shift + A** :
 - Lance les services apache et mariadb
 - Log dans le terminal quand tout est lancé
 - Ce raccourci est un interrupteur : apache et mariadb s'arrêtent si j'appuie à nouveau dessus.

Pour répondre à la question, nous allons tout d'abord créer un script bash comme ci-dessous :

```
#!/bin/bash

if [[ $1 == "start" ]]; then
    echo "Starting Apache and MariaDB..."
    sudo service apache2 start
    sudo service mysql start
    echo "Apache and MariaDB started."
else
    echo "Stopping Apache and MariaDB..."
    sudo service apache2 stop
    sudo service mysql stop
    echo "Apache and MariaDB stopped."
fi
```

Maintenant que nous avons créé le script `.sh`, nous allons créer deux fonctions dans le fichier `.zshrc` pour faire fonctionner le script comme ci-dessous :

```
function start_services() {
    echo "Starting Apache and MariaDB..."
    ~/services.sh start
    echo "Apache and MariaDB started."
}

function stop_services() {
    echo "Stopping Apache and MariaDB..."
    ~/services.sh stop
    echo "Apache and MariaDB stopped."
}
```

Maintenant que nous avons créé les deux fonctions, nous allons les combiner dans le fichier `.zshrc` comme ci-dessous :

```
function toggle_services() {
    if [[ $SERVICES_RUNNING == "true" ]]; then
        stop_services
        unset SERVICES_RUNNING
    else
        start_services
        export SERVICES_RUNNING=true
    fi
}

bindkey '^A' toggle_services
```

Il nous reste plus qu'à utiliser la commande `source ~/.zshrc` pour que les modifications soient prises en compte.

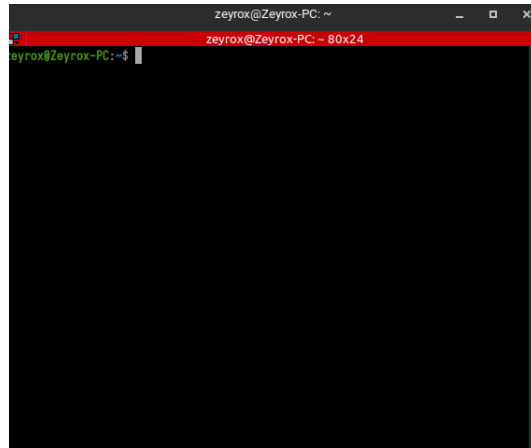
1.7 TD-9 : Terminaux

— Installez et essayez 3+ émulateur de terminaux parmi la liste précédente :

Pour ma part, je vais installer Terminator, Kitty et Tilix. Les installations des différents terminaux sont en dessous :

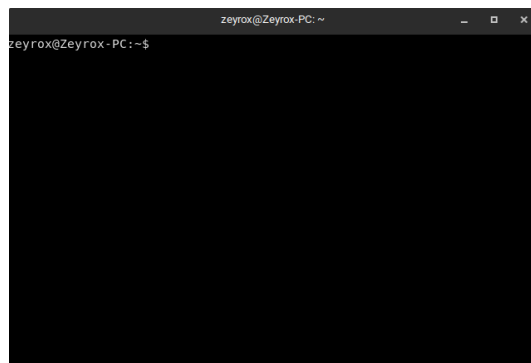
Terminator : `sudo apt-get install terminator`

Image de Terminator :



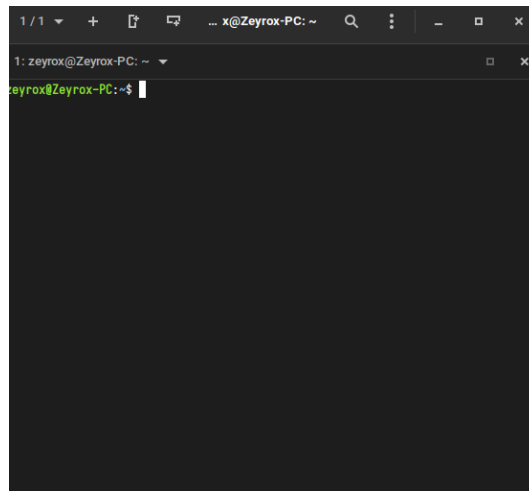
Kitty : `sudo apt-get install kitty`

Image de Kitty :



Tilix : `sudo apt-get install tilix`

Image de Tilix :



- Choisir un terminal et justifier pourquoi
 - Fonctionnalités
 - Rapidité
 - Documentation/Ecosystème

J'ai décidé de choisir terminator car il offre beaucoup de possibilités. Au niveau des fonctionnalités, Terminator permet d'avoir plusieurs terminaux dans un seul terminal (Multiplexage). On peut créer des onglets, définir des raccourcis et bien plus encore.

Ensuite, en ce qui concerne la rapidité, terminator est un terminal très rapide et réactif.

Et pour finir, Terminator a une communauté très active et il est très bien documenté. Donc on peut conclure que dans l'ensemble, Terminator est un bon choix de terminal en raison de ses fonctionnalités, de sa rapidité et de sa très bonne documentation. C'est pour cela que j'ai décidé d'utiliser Terminator.

Et de mon point de vue, j'ai toujours utilisé terminator car je trouve qu'il est facile à utiliser et une fois que l'on connaît les différentes combinaisons de touches, il peut devenir très puissant.

- Désinstaller les autres terminaux

Pour désinstaller les autres terminaux, nous allons utiliser la commande suivante :

`sudo apt-get remove (le nom de terminal)`. Par exemple : `sudo apt-get remove tilix`.

2 Client SSH

2.1 TD-1-SSH : Utilisation SSH

— Récupérer et démarrer l’environnement Vagrant sur arche ?

Pour ce nouveau TD, nous allons utiliser Vagrant pour utiliser SSH. Pour ce faire, nous allons télécharger depuis Arche le fichier Vagrant. Une fois que le fichier Vagrant est téléchargé, nous allons exécuter la machine Vagrant avec la commande ci-dessous :

```
enzocollot@MacBook-Air-de-Enzo vagrant % vagrant up  
Bringing machine 'cli' up with 'virtualbox' provider
```

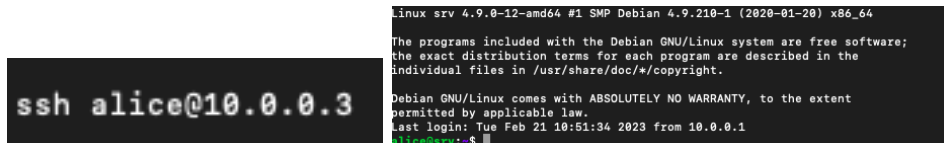
Une fois que la commande est terminée, on doit avoir ceci comme message qui indique que les machines sont bien démarrées :

```
==> cli: Machine 'cli' has a post 'vagrant up' message. This is a message  
==> cli: from the creator of the Vagrantfile, and not from Vagrant itself:  
==> cli:  
==> cli: Vanilla Debian box. See https://app.vagrantup.com/debian for help and b  
up reports  
  
==> srv: Machine 'srv' has a post 'vagrant up' message. This is a message  
==> srv: from the creator of the Vagrantfile, and not from Vagrant itself:  
==> srv:  
==> srv: Vanilla Debian box. See https://app.vagrantup.com/debian for help and b  
up reports
```

— Se connecter à la machine srv avec les utilisateurs alice, bob et carole sans utiliser vagrant ssh ?

Pour se connecter à la machine srv avec les utilisateurs alice, bob et carole sans utiliser vagrant ssh, nous allons utiliser la commande suivante :

Connexion à l'utilisateur alice en ssh :



```
ssh alice@10.0.0.3  
  
Linux srv 4.9.0-12-amd64 #1 SMP Debian 4.9.210-1 (2020-01-20) x86_64  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Tue Feb 21 10:51:34 2023 from 10.0.0.1  
alice@srv:~$
```

(a) Connexion SSH avec l'utilisatrice Alice

(b) Connexion Réussie avec l'utilisatrice Alice

FIGURE 3 – Deux captures d’écran montrant une connexion en ssh au serveur srv avec l’utilisatrice Alice

Maintenant que nous avons réussi à nous connecter au serveur ”srv” avec l’utilisateur ”Alice”, nous allons faire la même chose pour les deux autres utilisateurs.

Connexion à l'utilisateur bob en ssh :

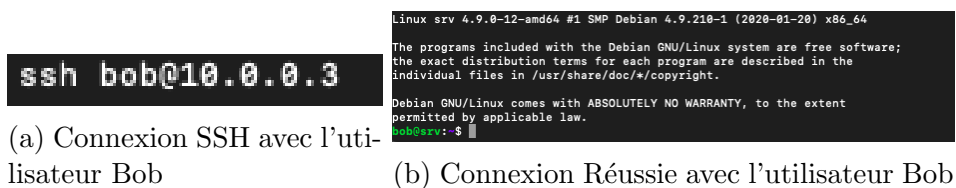


FIGURE 4 – Deux captures d'écran montrants une connexion en ssh au serveur srv avec l'utilisateur Bob

Connexion à l'utilisateur carol en ssh :

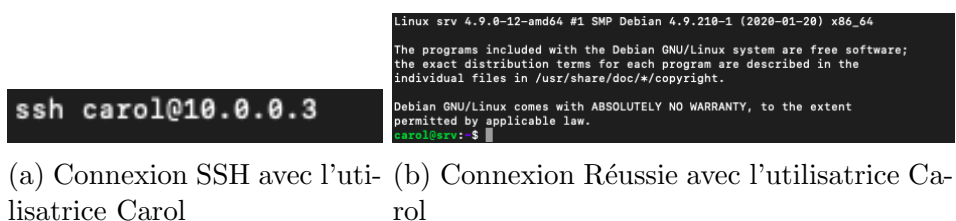


FIGURE 5 – Deux captures d'écran montrants une connexion en ssh au serveur srv avec l'utilisatrice Carol

— Vérifier qu'on est bien sur la machine Vagrant et pas en local ?

Pour vérifier que nous sommes bien sur la machine vagrant et non en local, nous allons utiliser la commande suivante :

```
carol@srv:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:8d:c0:4d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe8d:c04d/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:9c:cc:58 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.3/24 brd 10.0.0.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe9c:cc58/64 scope link
        valid_lft forever preferred_lft forever
```

On peut constater qu'au niveau des adresses IP, nous avons celle du serveur "srv", donc nous sommes bien sur la machine "vagrant".

— Consulter l' history local, que remarque-t-on ?

Voici, ci-dessous l'history local :

```
390 vagrant up
391 vagrant ssh
392 vagrant ssh srv
393 ssh alice@10.0.2.15
394 ssh alice@10.0.0.3
395 ssh alice@10.0.0.3
396 ssh bob@10.0.0.3
397 ssh carol@10.0.0.3
398 vagrant ssh srv
399 ssh carol@10.0.0.3
enzocollot@MacBook-Air-de-Enzo vagrant %
```

On peut constater que nous ne voyons pas les mots de passe que nous avons entrés pour nous connecter au serveur srv avec un utilisateur.

2.2 TD-2-SSH : Authentification Publique

— Créer une paire de clés privée et publique à l'aide de ssh-keygen ?

Pour cette nouvelle étape, nous allons créer une paire de clés privée et publique à l'aide de ssh-keygen. Donc voici les étapes ci-dessous pour le faire :

Pour commencer, nous allons ouvrir un terminal et taper la commande suivante :

```
enzocollot@MacBook-Air-de-Enzo ~ % ssh-keygen
```

Ensuite, il nous demande à quel emplacement nous voulons enregistrer notre clé. Nous laissons par défaut :

```
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/enzocollot/.ssh/id_rsa):
```

Ensuite, il nous demande une passphrase. Vous pouvez en indiquer une, mais j'ai laissé par défaut. Une fois que c'est fini, vous devez avoir cela qui devrait apparaître :

```
The key's randomart image is:
+--[RSA 3072]-----+
|
|  o..
|..   . o.+
|o.=   E +.+
|+* .   +.=
|o.=   . . S.
|+o o . . o
|o=   ... . .
|..*o+o+.
|==+o+.o.
+---[SHA256]-----+
enzocollot@MacBook-Air-de-Enzo ~ %
```

Voilà, nous avons générer une clé rsa.

- Utiliser la commande `ssh-copy-id` pour déposer la clé publique sur le compte `alice@cli`. Vérifier qu'on peut maintenant se connecter sans mot de passe ?

Pour cette nouvelle étape, nous allons utiliser la commande `ssh-copy-id` pour déposer la clé publique sur le compte `alice@cli`. Pour ce faire, nous allons utiliser la commande suivantes :

```
enzocollot@MacBook-Air-de-Enzo ~ % ssh-copy-id -i .ssh/id_rsa.pub alice@10.0.0.2
```

Une fois que nous avons entrée la commande, il va nous demander le mot de passe de Alice. Une fois le mots de passe entré, nous avons ceci qui apparaît :

```
alice@10.0.0.2's password:
Number of key(s) added:      1

Now try logging into the machine, with:  "ssh 'alice@10.0.0.2'"
and check to make sure that only the key(s) you wanted were added.
```

Maintenant, on peut se connecter en compte d'alice sur le serveur cli sans rentrer son mots de passe.

- Déposer manuellement la clé publique sur le compte `bob@cli`
Les nouvelles clés créés sont dans `/.ssh`
Les clés autorisées à se connecter au serveur sont dans `/.ssh/authorized_keys`

Pour déposer manuellement la clé publique sur le compte `bob@cli`, vous pouvez suivre les étapes ci-dessous :

Pour commencer, nous allons afficher le contenu de notre clé publique et le copier avec la commande suivante :

```
enzocollot@MacBook-Air-de-Enzo ~ % cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAI1Hw2okx8VAbmAZbY80Tdufc4DX9Q86xJN8Q
Z3HmYU0x08A+nCUl85YceZNG0ayVYQ18Nnua8dyTnfzRXV7TYg93QMfds/ZVFL+3wt4C/qMFPdJiVwbqA3BzK3Kx3FpY8yUSK4h
qN0Qf4ecPp7pMwMyZms4WCRSG0G9V/f22fcsApPwRVRNL96EkwH8jgJ3MeiITW4QDS3Pte1uKEIYF8xkLpDSzphosYD3j
uKUbTn1ZCMQ2D41s38MD7v8Bsdw70ZV0d1yx38X/wAwkSESvnt2jw081jYDk0c87Nsensws8iED27fthVKjDzydSpd3Ew3DP1pX
OvEX8j99sIrThYEVZC+Kvu8FZsh0nplQ0cyHV//H0wDPz:inBq8MlynpUzOHniInAcB2qHUE10Km1fpOvAH21qF7zn8EK8Z+HUB
Fz8B8li3azF8WdXjC0e-enzocollot@MacBook-Air-de-Enzo.local
enzocollot@MacBook-Air-de-Enzo ~ %
```

Une fois que nous avons copier la clé publique, nous allons nous connecter au compte de bob@cli avec la commande suivante :

```
ssh bob@cli
```

Une fois connecter sur le compte, nous allons verifier si le dossier .ssh existe. Si il existe pas, nous le créons avec la commande suivante :

```
mkdir /.ssh
```

Ensuite, on tape la commande suivante pour créer ou ouvrir le fichier "authorized_keys" :

```
nano /.ssh/authorized_keys
```

Maintenant, on copie le contenu de la clé publique dans le fichier comme ci-dessous :

A terminal window with a black background and white text. The text shows a public key being pasted into the authorized_keys file. The key is a long string of characters, including letters, numbers, and symbols like '+', '=', and ' '. The text is wrapped across several lines. The prompt 'ssh -i' is visible at the start of the first line.

Une fois que nous avons enregistré le fichier, c'est bon, on peut se connecter au compte de bob@cli sans taper le mot de passe avec la commande suivante :

```
ssh -i /.ssh/id_rsa.pub bob@cli
```