

Compte rendu OUTILS LIBRES

REVEL Rémi

9 février 2022

1 efficacité de l'environnement de travail

1.1 Désactivation de la souris

voici les commande pour desactiver la souris :

1. xinput set-prop 4 "Device Enabled" 0
2. xinput set-prop 6 "Device Enabled" 0
3. xinput set-prop 7 "Device Enabled" 0

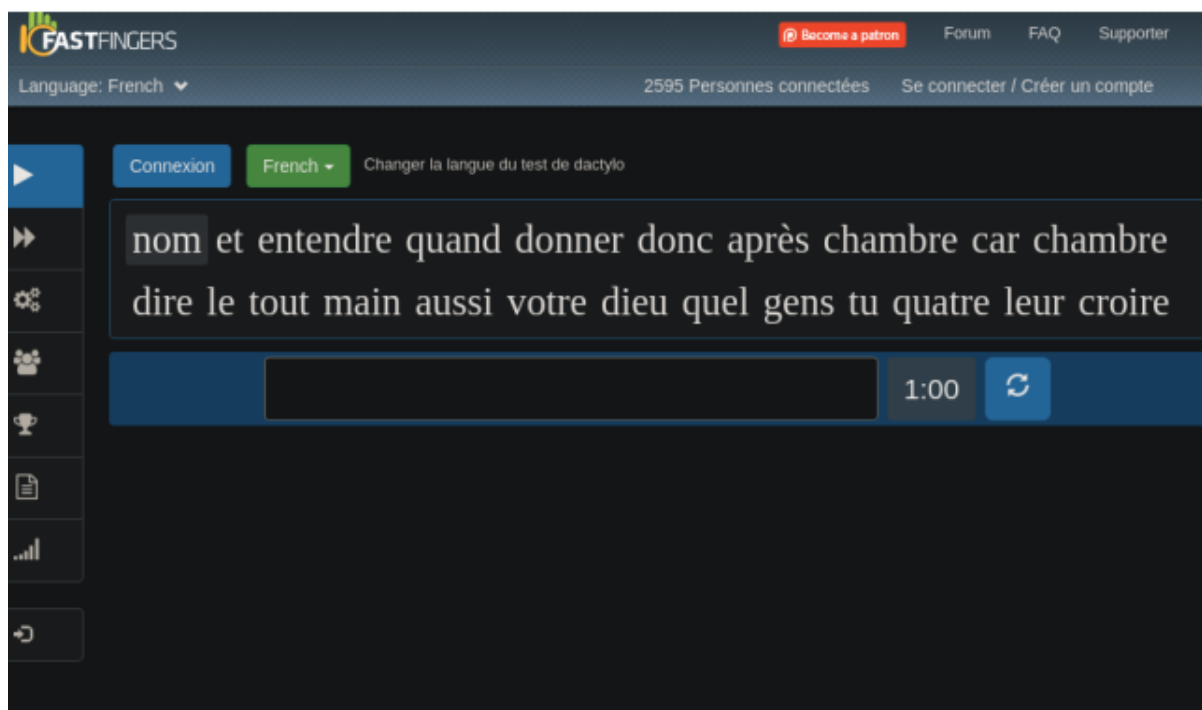
tableau de raccourcis clavier utile :

changer d'application	alt+tab ou windows+tab
gestionnaire d'application	Touche windows
Naviguer sur les elements cliquable d'une page web	tab
Fermer le navigateur	CTRL+W
Fermer une appliacation	ALT+F4
Changer d'onglet sous Brave	CTRL+1,2,3,...
Ouvrir un nouvelle onglet	CMD+T
faire une recherche	F6

1.2 S'ameliorer a la dactylographie

le site que j'ai retenu pour s'ameliorer en dactylographie est 10fastfingers.

On peut s'entrainer sur des mots aleatoire, ou sur nos propres texte, le site est disponible dans plusieurs langue.



1.3 Tutoriel pour VIM

insertion	i
enregister	:w
quitter	:q
aller au debut du fichier	:1
aller a la fin du fichier	:\$
annuler une action	u
recherche d'une occurrence	/occurrence
activer coloration syntaxique	:syntax on
templacer du texte	:s/origin/replacement/g

pour definir VIM comme editeur par default on a juste a rentrer cette commande :
update-alternatives --set editor /bin/vim

1.4 Bash history

Mon mot de passe n'apparaît pas dans le bash history, donc il n'y a pas d'informations sensible.
Les historiques sont propres a chaque shell utilisé.

Pour eviter de poluer notre historique avec des commande basique on peut les exclures avec cette commande :
export HISTIGNORE="ls : cd : pwd"

1.5 Alias de fonction

Les commandes presentes ci dessous sont a placer dans le .bashrc

```
function mkcd {  
    mkdir -p $1 && cd $1  
}  
  
function gitemergency {  
    git add . && git commit -m "test" && git push  
}
```

1.6 Script

Ce script est fait pour la sauvegarde des données des utilisateurs de la machine.

```
#!/usr/bin/env bash  
  
BACKUP_DIR=/tmp/$(hostname)  
  
getent passwd | egrep ':[0-9]{4}:' | while IFS=: read -r name password uid gid gecos home shell; do  
    USER_BACKUP_DIR=$BACKUP_DIR/$name  
    mkdir -p $USER_BACKUP_DIR  
    sudo cp -R $home/. $USER_BACKUP_DIR/home/  
done  
  
sudo dpkg --get-selections > $BACKUP_DIR/package.list  
sudo cp /etc/{passwd,group,shadow} $BACKUP_DIR  
sudo cp -R /etc/apt/sources.list* $BACKUP_DIR  
sudo apt-key exportall > $BACKUP_DIR/repo.keys  
BACKUP_ARCHIVE_FILE=/tmp/backup_$(hostname)_$(date --iso-8601=seconds).tar.gz  
sudo tar -czvf $BACKUP_ARCHIVE_FILE $BACKUP_DIR  
DISTANT_BACKUP_DIR=/backup/data  
mkdir -p $DISTANT_BACKUP_DIR  
cp $BACKUP_ARCHIVE_FILE $DISTANT_BACKUP_DIR  
cd $DISTANT_B
```

1.7 customisation avec OH MY ZSH

Dans le fichier du theme oh y zsh que l'on a selectioné on y rajoute ceci pour pouvoir voir le status de nos Vagrant et notre statut git.

```
# RVM settings
if [[ -s ~/.rvm/scripts/rvm ]] ; then
  RPS1="%{$fg[yellow]}rvm:%{$reset_color%}%{$fg[red]}\\$(~/.rvm/bin/rvm-prompt)%{$reset_color%} $EPS1"
else
  if which rbenv &> /dev/null; then
    RPS1="%{$fg[yellow]}rbenv:%{$reset_color%}%{$fg[red]}\\$(rbenv version | sed -e 's/ (set.*$//')'%{$reset_color%} $EPS1"
  fi
fi

ZSH_THEME_GIT_PROMPT_PREFIX="%{$reset_color%}%{$fg[green]}["
ZSH_THEME_GIT_PROMPT_SUFFIX="%{$reset_color%}"
ZSH_THEME_GIT_PROMPT_DIRTY="%{$fg[red]}*%{$reset_color%}"
ZSH_THEME_GIT_PROMPT_CLEAN=""
ZSH_THEME_VAGRANT_PROMPT_PREFIX="%{$fg_bold[blue]}["
ZSH_THEME_VAGRANT_PROMPT_SUFFIX="%{$fg_bold[blue]}]%{$reset_color%} "
ZSH_THEME_VAGRANT_PROMPT_RUNNING="%{$fg_no_bold[green]}●"
ZSH_THEME_VAGRANT_PROMPT_POWEROFF="%{$fg_no_bold[red]}●"
ZSH_THEME_VAGRANT_PROMPT_SUSPENDED="%{$fg_no_bold[yellow]}●"
ZSH_THEME_VAGRANT_PROMPT_NOT_CREATED="%{$fg_no_bold[white]}○"

# Customized git status, oh-my-zsh currently does not allow render dirty status before branch
git_custom_status() {
  local cb=$(git_current_branch)
  if [ -n "$cb" ]; then
    echo "$(parse_git_dirty)$ZSH_THEME_GIT_PROMPT_PREFIX$(git_current_branch)$ZSH_THEME_GIT_PROMPT_SUFFIX"
  fi
}

PROMPT='$(git_custom_status)%{$fg[cyan]}[%~% ]%{$reset_color%}%B$b $(vagrant_prompt_info)$(svn_prompt_info)$(git_prompt_info)%(!.#.$) '
```

1.8 Raccourci clavier

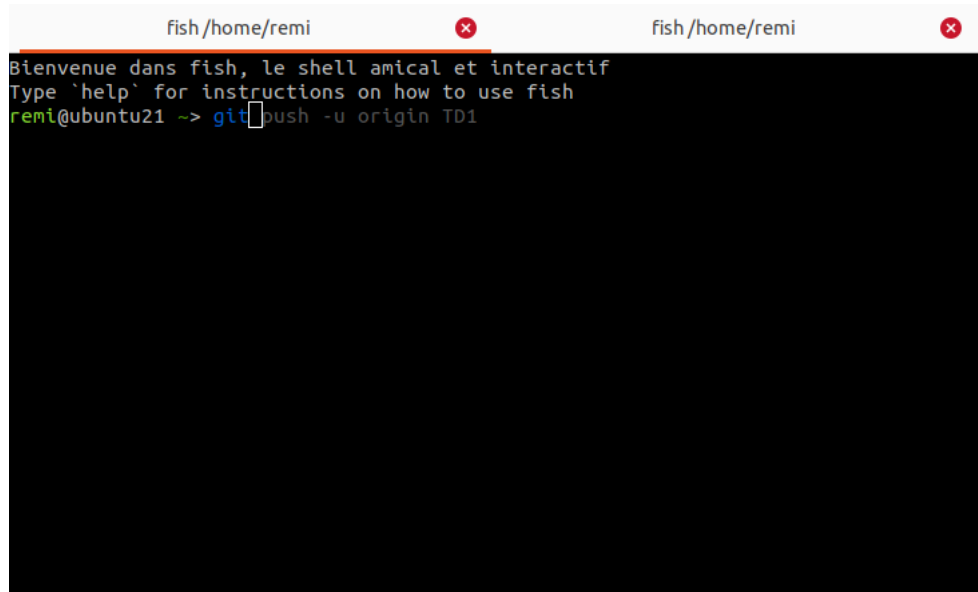
Pour faire un raccourcis clavier qui start ou stop apache en faisant CTRL+A, on place ce petit bout de code dans notre .bashrc

```
function ctrlSA()
{
  stat=`sudo service --status-all | grep apache2 | cut -d" " -f3`
  if [[ $stat == '+' ]]
  then
    systemctl stop apache2
    systemctl status apache2
  else
    systemctl start apache2
    systemctl status apache2
  fi
}
```

1.9 Emulateur de terminaux

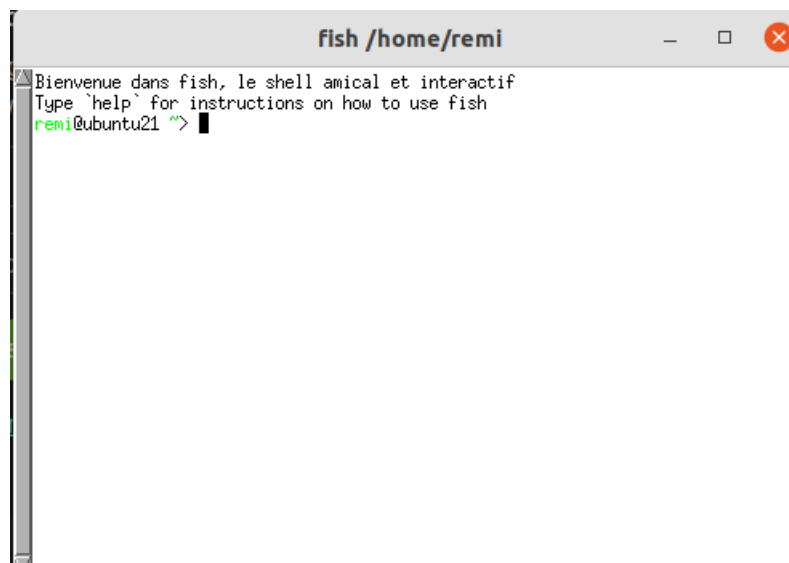
Sakura :

C'est un terminal assez simple, il est léger, il fait rien de plus que le terminal qui est présent de base sur debian, son avantage un noir profond ce qui fait bien ressortir les couleurs, on a la possibilité d'avoir plusieurs onglets. Couplé au shell fish qui possède une coloration syntaxique et de l'auto complétion basé sur l'historique, ça rend le terminal intéressant.





urxvt :

C'est un terminal très léger, il prend très peu de ressource, il n'y a pas de possibilité de faire un nouvel onglet. Il est pas très joli à voir.









Tilix :



C'est un terminal beaucoup plus avancé que les autres, on peut le customiser comme on veut, on peut aussi modifier les raccourcis clavier. Les fenetres de ce terminal peuvent etre logé sous forme de mosaïque. Il y a la possibilité que les commandes que l'on tape soit répliqué sur d'autres terminaux. Pour un admin sys, c'est un tres bon terminal.

1 / 1 +  

Tilix: fish /home/remi


  -  



1: top /home/remi  


2: fish /home/remi  


top - 21:51:41 up 9:50, 2 users, load a
Tâches: 422 total, 1 en cours, 420 en ve
%Cpu(s): 0,7 ut, 0,4 sy, 0,0 ni, 98,9 i
MiB Mem : 15940,5 total, 897,2 libr,
MiB Éch: 2048,0 total, 2043,7 libr,

PID	UTIL.	PR	NI	VIRT	RES
1522	remi	20	0	6427216	385276
69342	remi	20	0	667540	75252
36296	remi	20	0	4286616	241436
68428	remi	20	0	24,5g	187196
1411	remi	9	-11	2801144	26940
3261	remi	20	0	16,6g	464848
66674	root	20	0	0	0
698	root	-2	0	0	0
3318	remi	20	0	323460	11732
3416	remi	20	0	16,7g	209884
3420	remi	20	0	16,4g	101892
5034	remi	20	0	36,8g	166040
5217	remi	20	0	41,1g	367236
69580	remi	20	0	22064	4560
1	root	20	0	164944	11264
2	root	20	0	0	0
3	root	0	-20	0	0
4	root	0	-20	0	0

tmpfs 7,8G 269M 7,6G 4
% /dev/shm
tmpfs 5,0M 4,0K 5,0M 1
% /run/lock
tmpfs 7,8G 0 7,8G 0
% /run/qemu
/dev/nvme0n1p2 96M 31M 66M 32
% /boot/efi
tmpfs 1,6G 18M 1,6G 2
% /run/user/1000
remi@ubuntu21 ~> 

3: fish /home/remi  



Bien
venue dans fish, le shell amical et inter
actif
Type `help` for instructions on how to us
e fish
remi@ubuntu21 ~> 

2 SSH

2.1 Première connection en SSH

Pour se connecter en ssh a srv on entre cette commande :

```
sudo ssh carol@10.0.0.3
```

on est bien en ssh sur cette machine car ce n'est pas mon ip qui est affiché

```
carol@srv:~$ ip addr
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:29:80:01 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:fe29:8001/64 scope link
        valid_lft forever preferred_lft forever
```

2.2 Public key

ssh-keygen -b 4096 va nous créer une clé, on lui indique que l'on veut la stocker dans un fichier spécifique, dans mon cas le fichier id_rsa.

Cette clé une fois créée on la transfère à la machine de destination scp

```
.ssh/id_rsa.pub bob@10.0.0.3 ./clepub.txt
```

 il ne reste plus qu'à copier cette clé dans le fichier .ssh/authorized_keys

```
cat clepub.txt > .ssh/authorized_keys
```

Une passphrase c'est une phrase qui est demandée lors de notre connexion en ssh, elle permet de protéger la clé publique

2.3 Know host

Pour ajouter la clé publique au fichier know host au fichier, il faut taper cette commande :

```
ssh-keyscan -H 10.0.0.3 > .ssh/known_hosts
```

L'alias de commande pour se connecter à bob avec bs, on configure ça dans le fichier config de ssh

```
Host bs
    Hostname 10.0.0.3
    Port 22
    User bob
```

```
user1@debian ~ % ssh bs
Enter passphrase for key '/home/user1/.ssh/id_rsa':
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: Mon Dec  6 10:23:57 2021 from 10.0.0.1
```

2.4 SFTP / SSHFS

SFTP

Pour se connecter en sftp a alice c'est la meme syntaxe que ssh mais juste sftp a la place. Il ne reste plus q'ua taper les commande pour transferer des fichier.

```
user1@debian ~ % sftp alice@10.0.0.3
Enter passphrase for key '/home/user1/.ssh/id_rsa':
Connected to alice@10.0.0.3.
sftp> put /home/user1/vgrt/vagrant/README.md
Uploading /home/user1/vgrt/vagrant/README.md to /home/alice/README.md
/home/user1/vgrt/vagrant/README.md          100% 288 251.6KB/s 00:00
sftp> get /home/bob/test.pub
Fetching /home/bob/test.pub to test.pub
/home/bob/test.pub                          100% 389 173.7KB/s 00:00
sftp> exit
```

on voit bien que les fichier on été transferé

```
alice@srv:~$ ls
README.md

user1@debian ~ % ls | grep test.pub
test.pub
```

SSHFS

```
user1@debian ~ % mkdir /tmp/alicesrv
user1@debian ~ % sshfs alice@10.0.0.3:/home/alice /tmp/alicesrv
Enter passphrase for key '/home/user1/.ssh/id_rsa':
user1@debian ~ % ls /tmp/alicesrv
README.md
```

nano /tmp/alicesrv/README.md

j'ai effacé le contenu et mis test a la place

```
alice@srv:~$ cat README.md
test
```

2.5 Tunnel SSH

Pour se connecter au serveur en passant par cli il faut entrer cette commande, et ne pas quitter le terminal pour ne pas fermer la connection.

La commande est celle la : ssh -L 8000 :10.0.0.3 :80 alice@10.0.0.2

ensuite on essaie de voir si on arrive a recuperer un fichier sur le serveur

```
remi@ubuntu21 ~-> curl http://localhost:8000/cgi-bin/test1.cgi
Connexion depuis: 10.0.0.2
Adress de cli.local: 10.0.0.2
OK: la connexion est bien établie depuis l'adresse de la VM cli
```

Si on coupe le tunnel le curl va echouer

```
remi@ubuntu21 ~-> curl http://localhost:8000/cgi-bin/test1.cgi
curl: (7) Failed to connect to localhost port 8000: Connexion refusée
```

2.6 Tunnel SSH TSOCKS

On fait la meme commande qui pour le tunnel ssh mais en changeant de port ssh -L 9000 :10.0.0.3 :80 alice@10.0.0.2

Sur la machine intermediaire on ouvre le port 9000 : ssh -D 9000 alice@cli.local

Pour permettre un SOCKS Proxy sur srv on fait ca : tsocks firefox

2.7 X11 Forwarding

Sur la machine cible on installe x11-apps, ensuite on s'y connecte en ssh avec l'option -X pour le forwarding x11.

```
user1@debian ~/v/vagrant> ssh -X carol@10.0.0.2
carol@cli:~$ xeyes
```

l'application se lance bien sur la cible et s'affiche sur notre poste



2.8 Proxyjump

Il faut ajouter dans le fichier config de ssh

```
Host bastion
  Hostname 10.0.0.2
  User bob

Host srv
  Hostname 10.0.0.3
  ProxyJump bastion
  User bob
```

avec cette methode on ne laisse pas de trace sur les machines et c'est plus securisé

2.9 Proxycommand

Il faut ajouter dans le fichier config de ssh

```
Host bastion
  Hostname 10.0.0.2
  User bob

Host srvcmd
  Hostname 10.0.0.3
  User bob
  ProxyCommand ssh bastion -W %h:%p
```

Cette methode est moins securisé et laisse des traces sur la machine intermediaire

3 GIT

3.1 Premier pas git

on fait un git init

```
user@debian:~$ cd testgit/  
user@debian:~/testgit$ git init
```

apres avoir copié le vagrant, lors du git status on voit les fichier non enregistré

```
user@debian:~/testgit$ git status  
Sur la branche master  
Aucun commit  
Fichiers non suivis:  
(utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)  
Vagrantfile  
srv/
```

apres un up un fichier .vagrant apparait

```
user@debian:~/testgit$ git status  
Sur la branche master  
Aucun commit  
Fichiers non suivis:  
(utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)  
.vagrant/  
Vagrantfile  
srv/
```

pour eviter ca on ajoute ce fichier dans un gitignore

ensuite on fait un git add .

puis un git commit -m "test commit"

```
user@debian:~/testgit$ git commit -m "test commit"  
[master (commit racine) dd31ff2] test commit  
3 files changed, 70 insertions(+)  
create mode 100644 Vagrantfile  
create mode 100755 srv/test1.cgi  
create mode 100755 srv/test2.cgi
```

on peut voir dans les log que ce'st bien pris en compte

```
user@debian:~/testgit$ git log  
commit dd31ff2929f50b46eddbf5ea8a03a615e14ee8 (HEAD -> master)  
Author: remi <remirevel2000@gmail.com>  
Date: Thu Jan 13 09:54:03 2022 +0100  
test commit
```

3.2 git branche

Pour creer une nouvelle branche on entre ca : git checkout -b nouvelle-branche-test

```
user@debian:~/testgit$ git add -p Vagrantfile  
diff --git a/Vagrantfile b/Vagrantfile  
index ea456ce..27a0779 100644  
--- a/Vagrantfile  
+++ b/Vagrantfile  
@@ -9,6 +9,7 @@ Vagrant.configure("2") do |config|  
    set -ex  
    apt-get update  
    apt-get -y install libnss-mdns  
+   useradd --shell /bin/bash --create-home patrick || true  
    useradd --shell /bin/bash --create-home alice || true  
    useradd --shell /bin/bash --create-home bob || true  
    useradd --shell /bin/bash --create-home carol || true
```

```
user@debian:~/testgit$ git commit -m "testbranch1"  
[nouvelle-branche-test ded0a2e] testbranch1  
1 file changed, 2 insertions(+)
```

on retourne sur le master et on voit que rien n'a bougé

```
user@debian:~/testgit$ git checkout master
Basculement sur la branche 'master'
```

pour mettre les modifs sur le main on a juste a faire git merge nouvelle-branche-test
Dans le git log on voit bien les modif, il reste plus qu'a supprimé la branche

```
user@debian:~/testgit$ git branch -d nouvelle-branche-test
Branche nouvelle-branche-test supprimée (précédemment 615cdf0).
user@debian:~/testgit$ git branch
* master
```

3.3 git conflit

Sur la nouvelle branche on met le port 8081 dans le Vagrantfile

```
user@debian:~/testgit$ git checkout -b forward-new-port
Basculement sur la nouvelle branche 'forward-new-port'
user@debian:~/testgit$ nano Vagrantfile
user@debian:~/testgit$ git add Vagrantfile
user@debian:~/testgit$ git commit
Abandon de la validation dû à un message de validation vide.
user@debian:~/testgit$ git commit -m "8081"
[forward-new-port 85613c7] 8081
1 file changed, 1 insertion(+)
```

Et sur le main on met le port 8080

```
user@debian:~/testgit$ git checkout master
Basculement sur la branche 'master'
user@debian:~/testgit$ nano Vagrantfile
user@debian:~/testgit$ git add Vagrantfile
user@debian:~/testgit$ git commit -m "8080.2"
[master 3b73fbe] 8080.2
1 file changed, 1 insertion(+), 1 deletion(-)
```