

SHELL LINUX

TP Final



livecampus

Yoan Reghal

10/11/2022

Sommaire

<u>Introduction</u>	3
<u>Théorie</u>	4
<u>Script de gestion Nginx</u>	6
Installation	6
Mise à jour	10
Désinstallation	10
<u>Ansible</u>	11
<u>Scripts supplémentaires</u>	12
<u>PlusOuMoins</u>	12
<u>Pendu</u>	13
<u>Images Html</u>	14

Introduction

Le tp suivant aura pour but de valider les compétences apprises durant le cours sur la shell Linux.

La première partie sera un ensemble de questions théoriques, la deuxième partie sera quant à elle portée sur le scripting et la troisième partie consistera en la configuration d'Ansible.

Les scripts, en plus des captures d'écran présentes dans le rendu, sont également disponibles dans [ce répertoire git](#).

Théorie

1. Qu'est ce qu'un shell ?

Le shell est le programme qui permet à l'ordinateur d'interpréter les commandes qu'il reçoit de la part des utilisateurs et applications.

2. Quel est le shell racine disponible sur tous les systèmes Linux ?

Bash.

3. Quelle est la différence entre un langage de script et un langage de programmation ?

Un programme a besoin d'être compilé avant de pouvoir être exécuté.

Un script est interprété par le shell comme une chaîne de commandes lors de son exécution.

4. Comment indiquer à un script l'interpréteur de commandes à utiliser ?

Grâce au shebang présent au début du script.

5. Quelle est la différence entre une variable locale et une variable d'environnement ?

Les variables d'environnement sont disponibles à l'ensemble des shell "enfants" engendrés par le shell. Les variables locales sont limitées à leur shell de création.

6. Comment configurer le shell de manière persistante ?

En modifiant le fichier `bashrc`, présent dans le dossier `home` de l'utilisateur.

7. Quelle est l'utilité d'une condition ?

Les conditions permettent à un script ou programme d'effectuer différentes tâches selon différents critères définis.

8. Dans quels cas va-t-on préférer utiliser la boucle foreach plutôt que la boucle while ? Vice-versa ?

La boucle While permet l'exécution de commandes selon une ou plusieurs conditions, La boucle For permet l'itération des commandes pour chacune des valeurs dans une liste par exemple. La boucle while pourra par exemple echo un message tant qu'un service est en cours, et la boucle for servira par exemple à ping une liste d'IP s'arrêter à la fin.

9. Qu'est-ce que l'exit status d'un programme ou d'une commande ? Comment y accède-t-on ?

L'exit status est un nombre entier de 0 à 255. Une valeur égale à 0 signifie que la commande a été exécutée avec succès, un autre nombre (1-255) indique une erreur. L'exit status est stocké dans la variable '\$?'. Pour afficher le status de la commande précédente, on pourra 'echo \$? '.

10. Comment afficher la liste de tous les arguments envoyés au programme ?

La commande 'echo \$@' dans un script permettra d'afficher tous les arguments qui lui sont passés.

Script de gestion Nginx

Le script présenté a pour ambition de permettre à l'utilisateur de télécharger les sources d'une version particulière de Nginx, puis de les compiler, de les installer puis de créer automatiquement les fichiers systemd.

Le script permet également la mise à jour avec ou sans archivage des fichiers de configuration, et la désinstallation avec ou sans archivage de l'installation précédente. [git](#).

```
> sudo ./gestion_nginx.sh
1) Installer
2) Mettre a jour
3) Desinstaller
4) Quitter
Choisir une operation: 
```

a) Installation

Si l'utilisateur choisit l'option 1, le script installera les dépendances spécifiques à Debian ou RedHat selon les fichiers systèmes trouvés.

```
select op in "${ops[@]}"; do
  case $op in
    "Installer")
      if [ -f /etc/debian_version ]; then
        echo -e "\033[0;32mInstallation des dépendances Debian... \033[0m "
        apt install -q build-essential libpcre3 libpcre3-dev zlib1g zlib1g-dev libssl-dev libgd-dev \
          libxml2 libxml2-dev uuid-dev devscripts -y 2>/dev/null >/dev/null \
          /
      elif [ -f /etc/redhat-release ]; then
        echo -e "\033[0;32mInstallation des dépendances Redhat... \033[0m "
        yum install -q pcre pcre-devel zlib zlib-devel openssl openssl-devel -y 2>/dev/null >/dev/null
      fi
    ;;
  esac
done
```

Le recours à la suppression des erreurs et messages de commande n'est qu'à but esthétique.

Il est dans un second temps demandé à l'utilisateur la version du logiciel qu'il aimerait installer. Pour faciliter son choix, les versions accessibles sont listées avec `rmadison nginx`

```
echo -e "\033[0;34mVersions de Nginx possibles : \033[0m "
rmadison nginx
read -p $'\e[34mQuelle version installer ? ( ex: \e[32m1.22.1 \e[34mpour installer la version 1.22.1-1 \e[34m):\e[0m ' ver
```

```

Versions de Nginx possibles :
nginx      | 1.6.2-5+deb8u5      | oldoldstable      | source, all
nginx      | 1.10.3-1+deb9u4     | oldoldstable      | source, all
nginx      | 1.10.3-1+deb9u4     | oldoldstable-debug | source
nginx      | 1.14.1-1~bpo9+1     | stretch-backports | source, all
nginx      | 1.14.1-1~bpo9+1     | stretch-backports-debug | source
nginx      | 1.14.2-2+deb10u4    | oldstable          | source, all
nginx      | 1.14.2-2+deb10u4    | oldstable-debug    | source
nginx      | 1.18.0-6.1+deb11u2  | stable              | source, all
nginx      | 1.18.0-6.1+deb11u2  | stable-debug        | source
nginx      | 1.18.0-6.1+deb11u3  | stable-new          | source, all
nginx      | 1.22.1-1             | testing             | source, all
nginx      | 1.22.1-1             | unstable            | source, all
nginx      | 1.22.1-1             | unstable-debug      | source
Quelle version installer ? ( ex: 1.22.1 pour installer la version 1.22.1-1 ):

```

La version indiquée sera stockée dans la variable *ver* et utilisée pour télécharger l'archive depuis le site officiel, puis un fichier de configuration standard sera créé automatiquement

```

echo -e "\033[0;32mTéléchargement de Nginx... \033[0m "
wget https://nginx.org/download/nginx-$ver.tar.gz
tar xzf nginx-$ver.tar.gz
cd nginx-$ver
echo -e "\033[0;32mConfiguration de Nginx... \033[0m "
./configure --sbin-path=/usr/bin/nginx \
            --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log \
            --http-log-path=/var/log/nginx/access.log --with-pcre --pid-path=/var/run/nginx.pid \
            --with-debug && /dev/null \

echo -e "\033[0;32mInstallation de Nginx... \033[0m "
make install && /dev/null
cd ..
rm nginx-$ver.tar.gz
nginx -v

```

Le fichier est ensuite compilé à l'aide de la commande *make*, l'archive est supprimée et la version installée est affichée à l'écran.

```

Téléchargement de Nginx...
--2022-11-18 17:37:26-- https://nginx.org/download/nginx-1.22.1.tar.gz
Resolving nginx.org (nginx.org)... 2a05:d014:edb:5702::6, 2a05:d014:edb:5704::6, 3.125.197.172, ...
Connecting to nginx.org (nginx.org)|2a05:d014:edb:5702::6|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1073948 (1.0M) [application/octet-stream]
Saving to: 'nginx-1.22.1.tar.gz'

nginx-1.22.1.tar.gz      100%[=====] 1.02M  --.-KB/s  in 0.1s

2022-11-18 17:37:26 (6.92 MB/s) - 'nginx-1.22.1.tar.gz' saved [1073948/1073948]

Configuration de Nginx...
Installation de Nginx...
nginx version: nginx/1.22.1

```

Le script créera ensuite les fichiers systemd grâce à la fonction suivante, lisible dans son entièreté dans le répertoire [git](#).

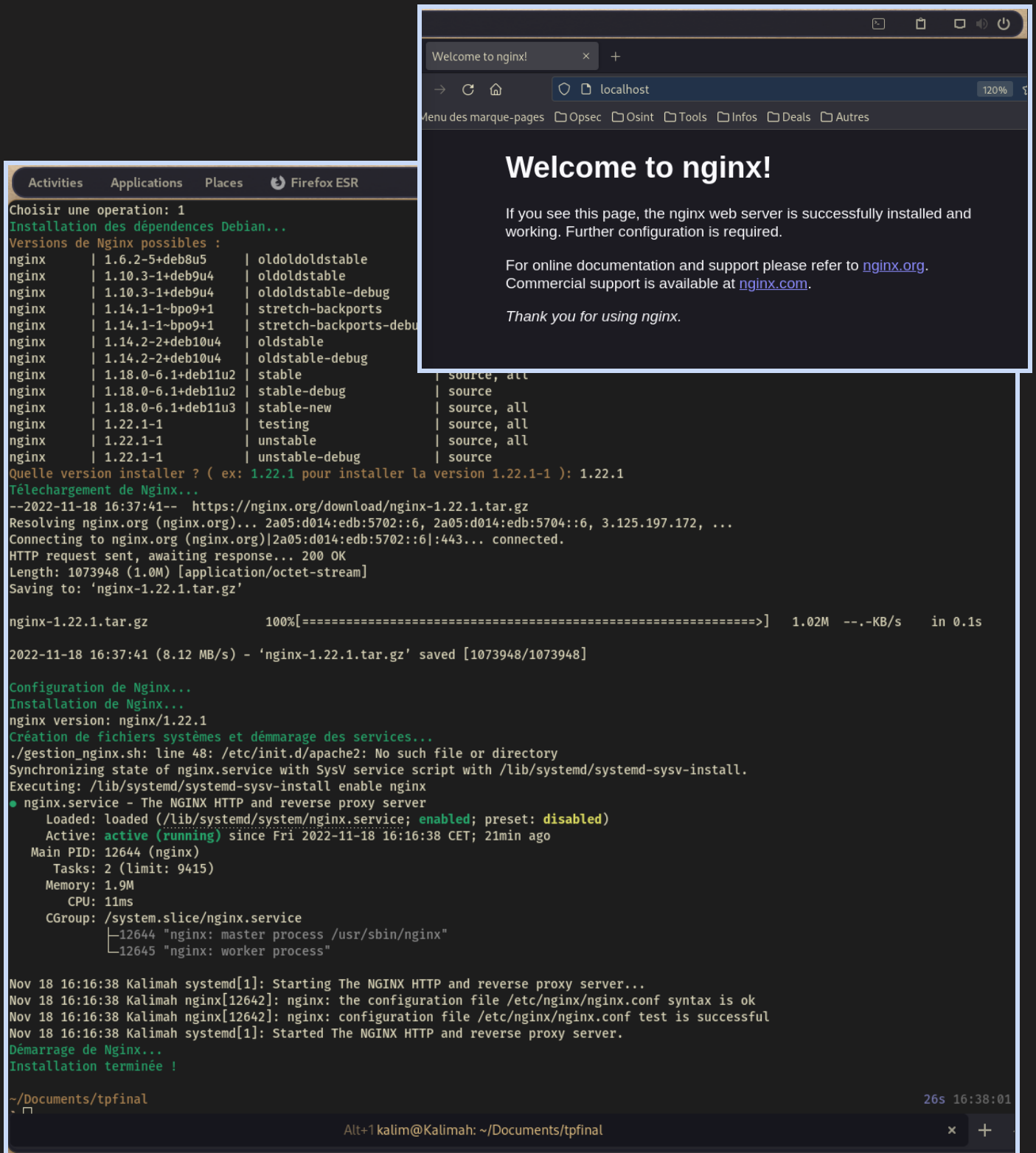
```
mksmd () {  
    bash -c "printf '%s\n' '[Unit]' 'Description=The NGINX HTTP and reverse proxy server '  
}
```

Les services sont ensuite redémarrés, Apache est stoppé pour éviter tout conflit, les restrictions sont levées sur Nginx et ses services sont démarrés.

```
echo -e "\033[0;32mCréation de fichiers systèmes et démarrage des services... \033[0m "  
touch /lib/systemd/system/nginx.service  
mksmd  
systemctl daemon-reload  
/etc/init.d/apache2 stop  
systemctl unmask nginx.service  
systemctl enable nginx.service  
systemctl start nginx.service  
systemctl status nginx.service  
echo -e "\033[0;32mDémarrage de Nginx... \033[0m "  
nginx  
echo -e "\033[0;32mInstallation terminée ! \033[0m "  
break  
;;
```

```
Configuration de Nginx...  
Installation de Nginx...  
nginx version: nginx/1.22.1  
Création de fichiers systèmes et démarrage des services...  
./gestion_nginx.sh: line 48: /etc/init.d/apache2: No such file or directory  
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable nginx  
● nginx.service - The NGINX HTTP and reverse proxy server  
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: disabled)  
   Active: active (running) since Fri 2022-11-18 16:16:38 CET; 21min ago  
 Main PID: 12644 (nginx)  
    Tasks: 2 (limit: 9415)  
  Memory: 1.9M  
     CPU: 11ms  
   CGroup: /system.slice/nginx.service  
           └─12644 "nginx: master process /usr/sbin/nginx"  
             └─12645 "nginx: worker process"
```


Si la connexion à localhost:80 mène à la page d'accueil Nginx alors l'installation est réussie.



b) Mise à jour

Si l'utilisateur sélectionne l'option 2, alors le script sauvegardera le fichier de configuration actuel en backup avant de mettre à jour le paquet à partir de la dernière version stable.

```
"Mettre a jour")

echo -e "\033[0;32mSauvegarde des fichiers de configuration... \033[0m "
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup

echo -e "\033[0;32mMise à jour... ! \033[0m "

if [ -f /etc/debian_version ]; then
    add-apt-repository ppa:nginx/stable
    apt update
    apt dist-upgrade -y nginx
elif [ -f /etc/redhat-release ]; then
    yum-config-manager --add-repo ppa:nginx/stable
    yum update nginx nginx-common nginx-core
fi
break
;;
```

c) Désinstallation

L'option 3 donne le choix d'archiver la version installée de nginx avant sa désinstallation, puis supprime ensuite l'ensemble Nginx ainsi que les dépendances maintenant inutiles.

```
"Desinstaller")

    read -p "Archiver nginx ? (o/n): " arc
    if [ "$arc" == [0/o] ];then
        echo -e "\033[0;32mArchivage... \033[0m "
        tar -zcvf nginx-backup.tar.gz /etc/nginx/
    else

        echo -e "\033[0;32mDesinstallation... \033[0m "
        if [ -f /etc/debian_version ]; then
            apt purge nginx nginx-common nginx-core -y
            apt autoremove -y
        elif [ -f /etc/redhat-release ]; then
            yum remove nginx nginx-common nginx-core -y
        fi
    fi
```

Ansible

Le but du playbook créé est d'installer les utilitaires de base sur chacun des postes et d'y installer les services Nginx, selon leur distribution.

```
---
- name: Installation d'utilitaires pour CentOs
  hosts: centos
  become: yes
  become_user: root
  tasks:
    - name: Installation de git
      yum: name=git update_cache=yes
    - name: installation de Vim
      yum: name=vim

    - name: Installation de Nginx pour CentOs
      become: yes
      yum: name=nginx state=latest
    - name: Enable
      become: yes
      service: name=nginx enabled=true
```

```
- name: Installation d'utilitaires pour Ubuntu
  hosts: ubuntu
  become: yes
  become_user: root
  tasks:
    - name: Installation de git
      apt: name=git update_cache=yes
    - name: installation de Vim
      apt: name=vim

    - name: Installation de Nginx pour Ubuntu
      become: yes
      apt: name=nginx state=latest
    - name: Enable
      become: yes
      service: name=nginx enabled=true
```

```
PLAY RECAP *****
centos1      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
centos2      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
centos3      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu1      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu2      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu3      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Scripts supplémentaires

a) [PlusOuMoins](#)

```
> ./plusoumoins.sh
Entre quoi ( minimum ) ? 20
Entre et quoi ( maximum ) ? 30
Devine ? 21
+ !
Devine ? 29
- !
Devine ? 25
- !
Devine ? 23
+ !
Devine ? 24
Bien joué, c'est bien 24 !
```

Ce jeu de + ou - commence par demander à l'utilisateur entre quel et quel chiffre démarrer le jeu. Un chiffre est aléatoirement choisi dans cette plage, puis est comparé avec chaque nouvelle tentative.

Si la tentative est inférieure au chiffre, un + est affiché, si elle est supérieure alors un - est affiché.

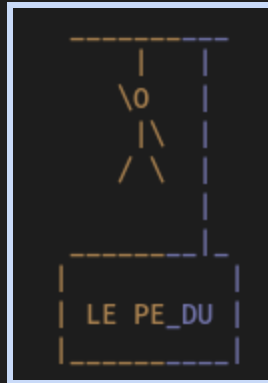
Cela continue jusqu'à ce que la tentative soit correcte.

```
read -p "Entre quoi ( minimum ) ? " min
read -p "Et quoi ( maximum ) ? " max
num=$(shuf -i $min-$max -n 1)

for((;;))
do
    read -p "Devine ? " dev

    if
    [ $dev -lt $num ]
    then
        echo " + ! "
        i=-1; continue;
    elif
    [ $dev -gt $num ]
    then
        echo " - ! "
        i=-1; continue
    elif
    [ $dev == $num ]
    then
        echo " Bien joué, c'est bien $num ! "
```

b) Pendu



La fonction *main* du jeu ouvre le fichier *PetitLarousse.txt*, en tire un mot au hasard et l'associe à la variable *mot*.

Si présents, les chiffres sont supprimés et les majuscules réduites.

Des tirets bas sont ajoutés pour chaque lettre du mot, et le nombre d'essais est affiché au dessus des lettres déjà tentées.

La fonction *rejouer* permet simplement de continuer à jouer sans quitter le script.

```
main() {  
  
    readarray -t a <PetitLarousse.txt  
    randind=`expr $RANDOM % ${#a[@]}`  
    mot=${a[$randind]}  
  
    essais=()  
    essaislist=()  
    guin=0  
  
    mot=`echo $mot | tr -dc '[:alnum:] \n\r' | tr '[:upper:]' '[:lower:]`'  
    long=${#mot}  
  
    for ((i=0;i<$long;i++)); do  
        essais[$i]="_"  
    done  
  
    mo=()  
  
    for ((i=0;i<$long;i++)); do  
        mo[$i]="${mot:$i:1}"  
    done  
  
    for ((j=0;j<$long;j++)); do  
        if [[ ${mo[$j]} == " " ]]; then  
            essais[$j]=" "  
        fi  
    done  
  
    rejouer(){  
        echo  
        echo -n "Rejouer? (o/n) "  
        read -n 1 choix  
        case $choix in  
            [oO]) clear  
                   main  
                   ;;  
            esac  
        exit  
    }  
}
```

Une boucle est créée pour dessiner un pendu évolutif à chaque erreur, jusqu'à 9 erreurs.

[illegible]

```
while [[ $faux -lt 9 ]]; do
    case $faux in
        0) echo " "
            ;;
        1) faux1
            ;;
        2) faux2
            ;;
        3) faux3
            ;;
        4) faux4
            ;;
        5) faux5
            ;;
        6) faux6
            ;;
        7) faux7
            ;;
        8) faux8
            ;;
    esac

    if [[ faux -eq 0 ]]; then
        for i in {1..9}
        do
            echo
        done
    fi
done
```

```

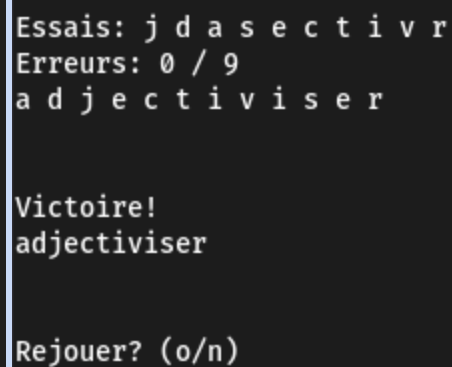
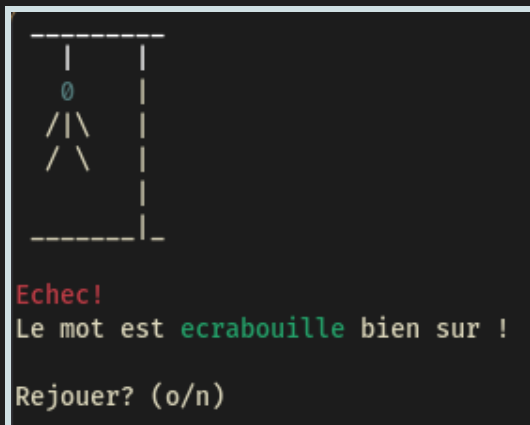
faux7 () {
    echo "          "
    echo "  |         |  "
    echo "  0         |  "
    echo "  |         |  "
    echo "             |  "
    echo "             |  "
    echo "             |  "
    echo "          _____|_"
}

faux8 () {
    echo "          "
    echo "  |         |  "
    echo "  0         |  "
    echo "  /\        |  "
    echo "             |  "
    echo "             |  "
    echo "             |  "
    echo "          _____|_"
}

faux9 () {

```

Le reste de la fonction permet d'incrémenter les essais ainsi que d'afficher les lettres utilisées, et également d'incrémenter le compteur de *faux* si la lettre n'est pas trouvée dans le mot. Si il n'y a plus de lettres à trouver, alors le jeu est gagné.



```

bon=0
for ((j=0;j<$long;j++)); do
    if [[ ${essais[$j]} == "_" ]]; then
        bon=1
    fi
done

echo Essais: ${essaislist[@]}
echo Erreurs: $faux / 9
for ((k=0;k<$long;k++)); do
    echo -n "${essais[$k]} "
done
echo
echo

if [[ bon -eq 1 ]]; then
    echo -n "Tapez une lettre : "
    read -n 1 -e lettre
    lettre=$(echo $lettre | tr [A-Z] [a-z])
    essaislist[$guin]=$lettre
    guin=`expr $guin + 1`
fi

f=0;
for ((i=0;i<$long;i++)); do
    if [[ ${mo[$i]} == $lettre ]]; then
        essais[$i]=$lettre
        f=1
    fi
done
if [[ f -eq 0 ]]; then
    faux=`expr $faux + 1`
fi

if [[ bon -eq 0 ]]; then
    echo
    echo Victoire!
    echo $mot
    echo
    rejouer
fi
clear

done

faux9
echo -e "\033[0;31mEchec!\033[0m"
echo -e "Le mot est \033[0;32m$mot\033[0m bien sur !"
rejouer
}

```

c) Images Html

Le but de ce script est de créer une page HTML contenant l'ensemble des images du répertoire courant. Si bien optimisé, il est possible de créer cela en quelques simples lignes, comme suit.

```
#!/usr/bin/bash  
  
for f in /*.jpg; do echo "<img src=\"$f\" />" >> all.html  
done
```

Pour chaque fichier finissant par *.jpg* dans le dossier courant, le script écrit une balise html dans laquelle l'image est sourcée, et l'ajoute au fichier *all.html*.

```
> ls  
imageshtml.sh magritte.jpg mgt.jpg oasis.jpg  
  
> ./imageshtml.sh  
  
> ls  
all.html imageshtml.sh magritte.jpg mgt.jpg oasis.jpg
```

Lancer la page avec un navigateur permet de vérifier son bon fonctionnement.



Merci de votre attention.