

TD1 Bash

1. Que font ces oneliners ou ces scripts BASH ? écrivez ce que vous pensez voir en sortie de la commande ou du script bash

1. Quel résultat donne cette ligne ?

```
DATE=$(date); echo $DATE
```

```
bastien_fedora@fedora:~$ DATE=$(date); echo $DATE
jeu. 04 avril 2024 10:27:39 CEST
bastien_fedora@fedora:~$
```

Elle affiche la date courante.

2. quel résultat donne cette ligne ?

```
cat /etc/passwd |wc -l
```

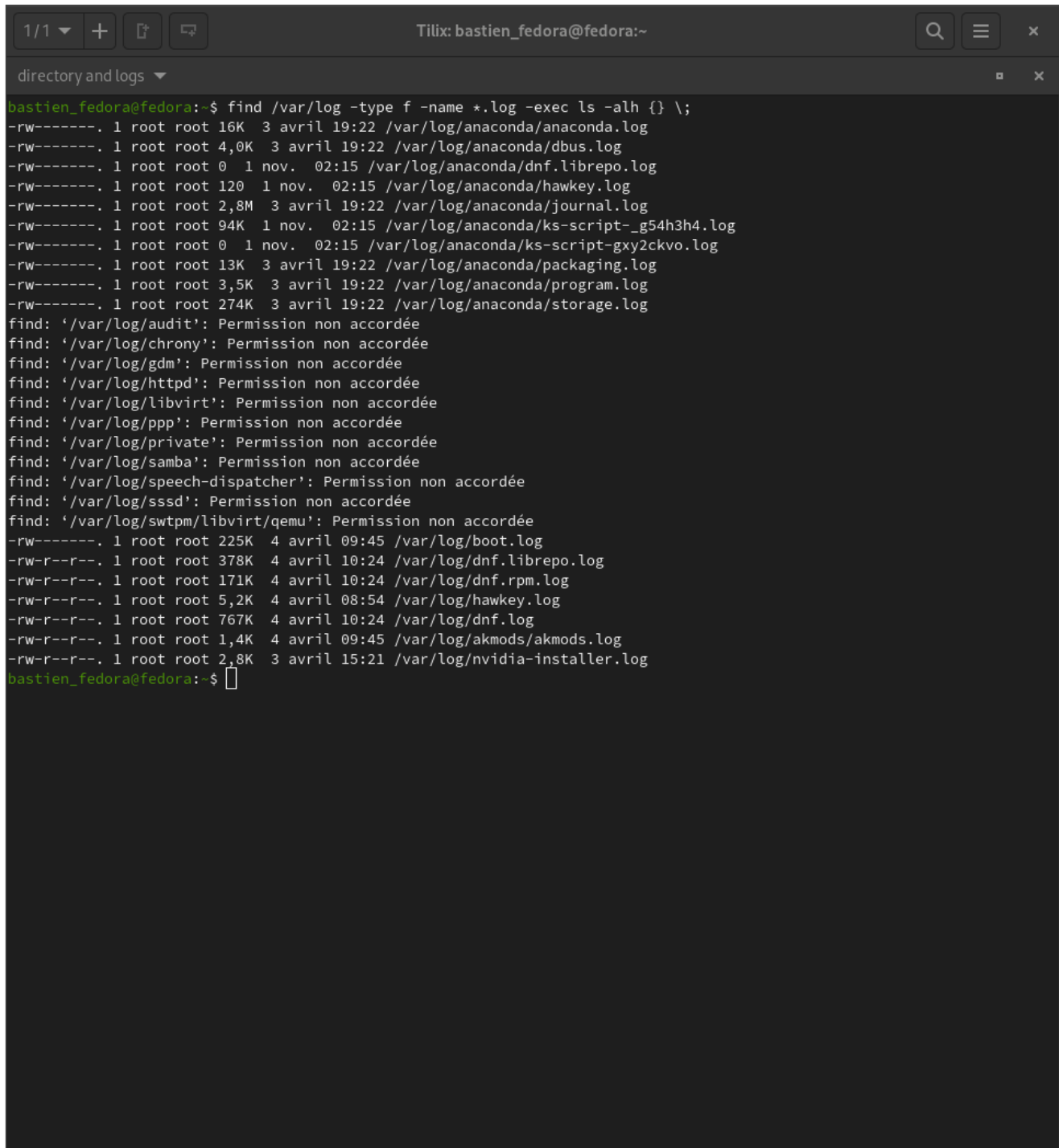
Elle affiche le nombre de lignes du fichier /etc/passwd.

```
bastien_fedora@fedora:~$ cat /etc/passwd |wc -l
50
bastien_fedora@fedora:~$
```

3.

```
find /var/log -type f -name *.log -exec ls -alh {} \;
```

Affiche les fichiers log de /var/log avec les droits d'accès.

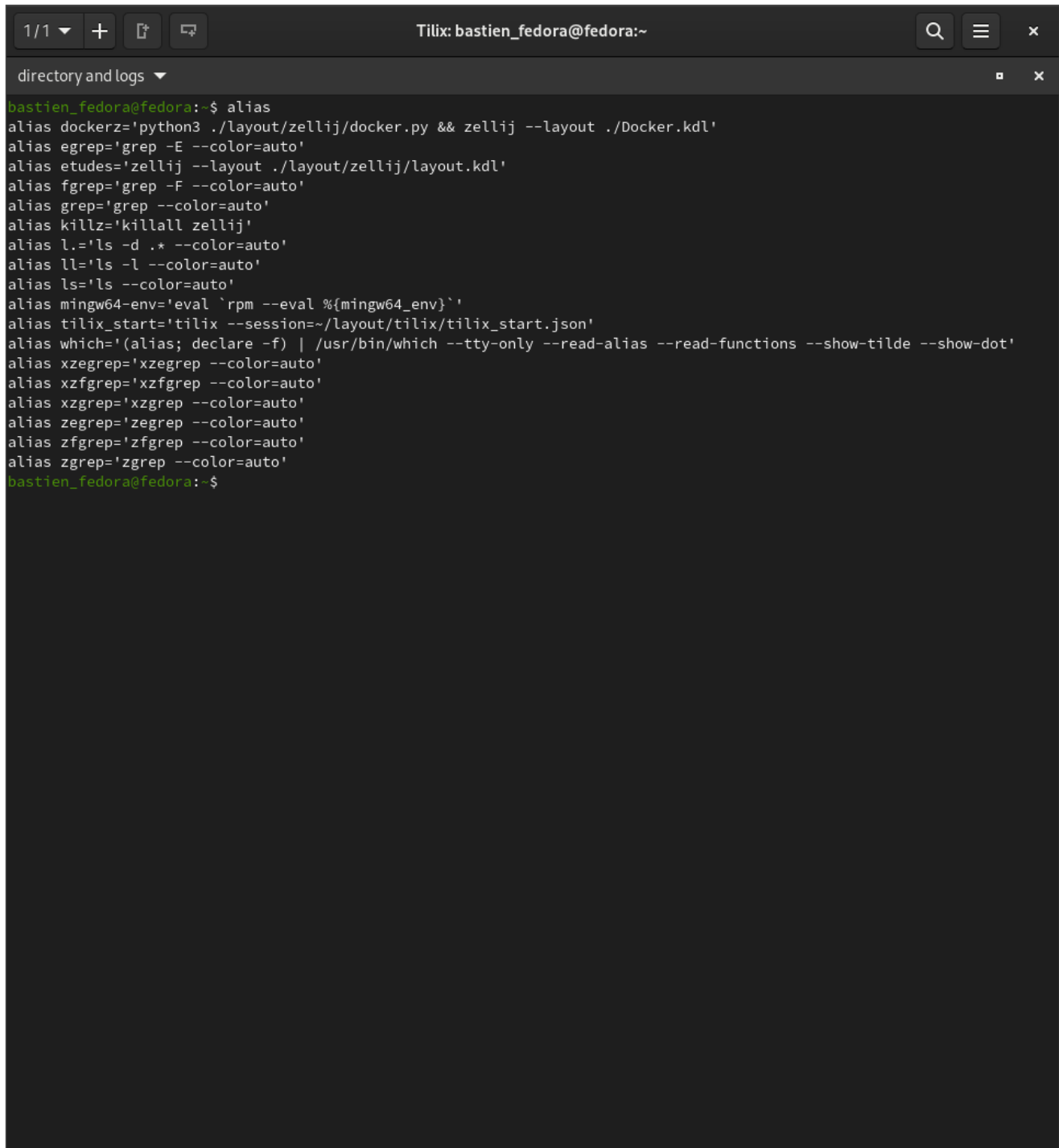


```
1/1 + [ ] [ ] TiliX: bastien_fedora@fedora:~  
directory and logs ▾  
bastien_fedora@fedora:~$ find /var/log -type f -name *.log -exec ls -alh {} \;  
-rw-----. 1 root root 16K  3 avril 19:22 /var/log/anaconda/anaconda.log  
-rw-----. 1 root root 4,0K  3 avril 19:22 /var/log/anaconda/dbus.log  
-rw-----. 1 root root 0  1 nov.  02:15 /var/log/anaconda/dnf.librepo.log  
-rw-----. 1 root root 120  1 nov.  02:15 /var/log/anaconda/hawkey.log  
-rw-----. 1 root root 2,8M  3 avril 19:22 /var/log/anaconda/journal.log  
-rw-----. 1 root root 94K  1 nov.  02:15 /var/log/anaconda/ks-script-_g54h3h4.log  
-rw-----. 1 root root 0  1 nov.  02:15 /var/log/anaconda/ks-script-gxy2ckvo.log  
-rw-----. 1 root root 13K  3 avril 19:22 /var/log/anaconda/packaging.log  
-rw-----. 1 root root 3,5K  3 avril 19:22 /var/log/anaconda/program.log  
-rw-----. 1 root root 274K  3 avril 19:22 /var/log/anaconda/storage.log  
find: '/var/log/audit': Permission non accordée  
find: '/var/log/chrony': Permission non accordée  
find: '/var/log/gdm': Permission non accordée  
find: '/var/log/httpd': Permission non accordée  
find: '/var/log/libvirt': Permission non accordée  
find: '/var/log/ppp': Permission non accordée  
find: '/var/log/private': Permission non accordée  
find: '/var/log/samba': Permission non accordée  
find: '/var/log/speech-dispatcher': Permission non accordée  
find: '/var/log/sss': Permission non accordée  
find: '/var/log/swtpm/libvirt/qemu': Permission non accordée  
-rw-----. 1 root root 225K  4 avril 09:45 /var/log/boot.log  
-rw-r--r--. 1 root root 378K  4 avril 10:24 /var/log/dnf.librepo.log  
-rw-r--r--. 1 root root 171K  4 avril 10:24 /var/log/dnf.rpm.log  
-rw-r--r--. 1 root root 5,2K  4 avril 08:54 /var/log/hawkey.log  
-rw-r--r--. 1 root root 767K  4 avril 10:24 /var/log/dnf.log  
-rw-r--r--. 1 root root 1,4K  4 avril 09:45 /var/log/akmods/akmods.log  
-rw-r--r--. 1 root root 2,8K  3 avril 15:21 /var/log/nvidia-installer.log  
bastien_fedora@fedora:~$
```

4. expliquez ce que fait cet alias. Quels sont les autres alias sur votre compte ?

```
alias cds='cd /Users/pouchou/ownCloud/cours\ iut\ 2/cours\  
iut/latex/inc/scripts'
```

Cet alias permet de se déplacer dans le répertoire donné. Les autres alias sont :



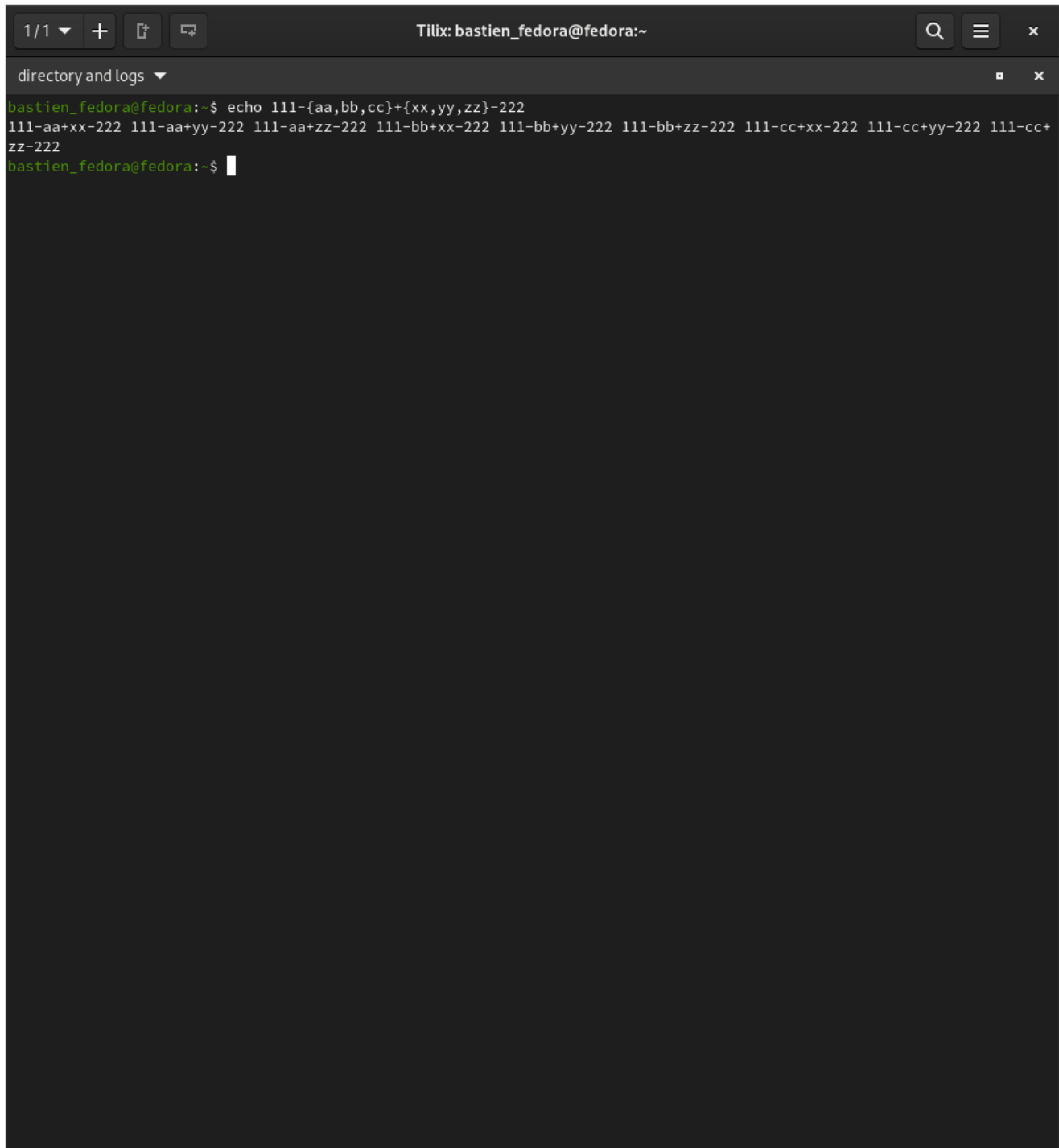
The screenshot shows a Tilix terminal window titled "Tilix: bastien_fedora@fedora:~". The terminal displays a series of alias definitions entered at the prompt. A sidebar on the left shows "directory and logs" with a search icon and a close button. The aliases are as follows:

```
bastien_fedora@fedora:~$ alias
alias dockerz='python3 ./layout/zellij/docker.py && zellij --layout ./Docker.kdl'
alias egrep='grep -E --color=auto'
alias etudes='zellij --layout ./layout/zellij/layout.kdl'
alias fgrep='grep -F --color=auto'
alias grep='grep --color=auto'
alias killz='killall zellij'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mingw64-env='eval `rpm --eval %{mingw64_env}`'
alias tilix_start='tilix --session=~/.layout/tilix/tilix_start.json'
alias which='(alias; declare -f) | /usr/bin/which --tty-only --read-alias --read-functions --show-tilde --show-dot'
alias xzgrep='xzgrep --color=auto'
alias xzfgrep='xzfgrep --color=auto'
alias xzgrep='xzgrep --color=auto'
alias zegrep='zegrep --color=auto'
alias zfgrep='zfgrep --color=auto'
alias zgrep='zgrep --color=auto'
bastien_fedora@fedora:~$
```

5. Que fait cette ligne ?

```
echo 111-{aa,bb,cc}+{xx,yy,zz}-222
```

Affiche les combinaisons possibles entre les accolades.



The screenshot shows a terminal window titled "Tmux: bastien_fedora@fedora:~". The prompt is "bastien_fedora@fedora:~\$". The command entered is `echo 111-{aa,bb,cc}+{xx,yy,zz}-222`. The output is a single line: `111-aa+xx-222 111-aa+yy-222 111-aa+zz-222 111-bb+xx-222 111-bb+yy-222 111-bb+zz-222 111-cc+xx-222 111-cc+yy-222 111-cc+zz-222`. The prompt is now "bastien_fedora@fedora:~\$".

6. Pourquoi le résultat diffère-t-il entre les deux boucles ? que pourriez vous en retenir comme conseil systématique ?

```
MESSAGE="hello world"
for i in "$MESSAGE"; do echo $i; done
hello world
for i in $MESSAGE; do echo $i; done
hello
word
```

Le premier for boucle sur une seule valeur, le second sur deux valeurs (sans-guillemet).

7. Utilisez l'expansion de variables afin de simplifier cette commande :

```
cp dirname-et-basename.sh dirname-et-basename.sh.bak
```

```
cp dirname-et-basename.sh{,.bak}
```

en ajoutant {,.bak} on ajoute le suffixe .bak au fichier dirname-et-basename.sh sans avoir à le réécrire.

8. Déterminez quel est le PID de votre bash via la commande `ps -ef`. comparer avec la variable `$$` ? Que fait cette commande ?

```
ps -ef | grep bash
```

```
```bash  
echo $$
```

```
bastien_fedora@fedora:~$ ps -ef |grep bash
bastien+ 3567 3557 0 09:45 pts/0 00:00:00 /bin/bash
bastien+ 3569 3557 0 09:45 pts/1 00:00:00 /bin/bash
bastien+ 3592 3557 0 09:45 pts/2 00:00:00 /bin/bash
bastien+ 3812 3784 0 09:45 pts/4 00:00:00 /bin/bash
bastien+ 3851 3784 0 09:45 pts/5 00:00:00 /bin/bash
bastien+ 3878 3784 0 09:45 pts/6 00:00:00 /bin/bash
bastien+ 3905 3784 0 09:45 pts/7 00:00:00 /bin/bash
bastien+ 3932 3784 0 09:45 pts/8 00:00:00 /bin/bash
bastien+ 3965 3784 0 09:45 pts/9 00:00:00 /bin/bash
bastien+ 4521 3878 0 09:46 pts/6 00:00:00 /bin/bash /usr/bin/brave-br
wser
bastien+ 41967 3557 0 10:46 pts/10 00:00:00 /bin/bash
bastien+ 42639 41967 0 10:48 pts/10 00:00:00 grep --color=auto bash
bastien_fedora@fedora:~$ echo $$
41967
bastien_fedora@fedora:~$
```

```
kill -9 $$
```

supprime le bash courant.

9.

```
ping -c1 localhost && { echo succes;} || { echo pasglop; }
```

```
bastien_fedora@fedora:~$ ping -c1 localhost && { echo succes;} || { echo pasglop; }
PING localhost(localhost (:::1)) 56 octets de données
64 octets de localhost (:::1) : icmp_seq=1 ttl=64 temps=0.078 ms

--- statistiques ping localhost ---
1 paquets transmis, 1 reçus, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.078/0.078/0.078/0.000 ms
succes
bastien_fedora@fedora:~$
```

## 2. Explorons \$\* et @\$ : quels résultats donnent les scripts suivants ?

1. Quel résultats pour la commande suivante : ./loopargs1.sh un deux trois quatre rappel du cours : Les variables \$\* et @\$ contiennent la liste des arguments d'un script shell. Lorsqu'elles ne sont pas entourées par des guillemets, elles sont équivalentes. \$\* ou @\$ ensemble des paramètres positionnels, équivalent à \$1 \$2 ... \$n "\$\*" ensemble des paramètres positionnels, équivalent à "\$1 \$2 ... \$n" "\$@" ensemble des paramètres positionnels, équivalent à "\$1" "\$2" ... "\$n"

```
#!/bin/bash
for i in $*
do
echo $i
done
echo -e '\n'
for i in @$
do
echo $i
done
echo -e '\n'
for i in "$*"
do
echo $i
done
echo -e '\n'
for i in "$@"
do
echo $i
done
```

```
bastien_fedora@fedora:~$ bash ./loopargs1.sh

bastien_fedora@fedora:~$
```

le programme affiche les arguments passés en paramètre (un deux trois quatre rappel du cours) en les séparant par des espaces.

### 3. Solutions pour lire un fichier.

1. Découverte de la variable IFS. a. Expliquez les résultats de ce script :

```
mkdir "repertoire avec espace"{1,2} &>/dev/null && touch "fichier "{1,2} \
& > /dev/null || echo -e "rep et fichiers presents"
for n in $(ls) ; do echo $n ; done
echo -e "-----"
echo -e "Avec IFS modifié"
IFS=$'\n'; for n in $(ls) ; do echo $n ; done
./test-ifs.sh
fichier
1
fichier
2
repertoire
avec
espace1
repertoire
avec
espace2
test-ifs.sh

Avec IFS modifie
fichier 1
fichier 2
repertoire avec espace1
repertoire avec espace2
test-ifs.sh
```

Le programme affiche le texte "rep et fichiers presents" puis les fichiers et répertoires présents dans le répertoire courant. En modifiant IFS, on affiche les fichiers et répertoires en prenant en compte les espaces.

Avant chaque for met un retour a la ligne, contrairement à la commande IFS=\$'\n'

b. Que fait ce script ?

```
#!/bin/bash
_file="${1:-/dev/null}" # sécurité en cas d'erreur
while IFS= read -r line
do
echo "$line"
done < "$_file"
```

Le script affiche le contenu du fichier passé en paramètre, en respectant la disposition ligne par ligne.

```

bastien_fedora@fedora:~$ bash ./loopargs1.sh /etc/passwd
root:x:0:0:Super User:/root:/bin/bash
bin:x:1:1:bin:/bin:/usr/sbin/nologin
daemon:x:2:2:daemon:/sbin:/usr/sbin/nologin
adm:x:3:4:adm:/var/adm:/usr/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/usr/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/usr/sbin/nologin
operator:x:11:0:operator:/root:/usr/sbin/nologin
games:x:12:100:games:/usr/games:/usr/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/usr/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/usr/sbin/nologin
dbus:x:81:81:System Message Bus:/usr/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
tss:x:59:59:Account used for TPM access:/usr/sbin/nologin
systemd-coredump:x:998:998:systemd Core Dumper:/usr/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/usr/sbin/nologin
systemd-oom:x:997:997:systemd Userspace OOM Killer:/usr/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/usr/sbin/nologin
systemd-timesync:x:996:996:systemd Time Synchronization:/usr/sbin/nologin
qemu:x:107:107:qemu user:/sbin/nologin
polkitd:x:114:114:User for polkitd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
geoclue:x:995:994:User for geoclue:/var/lib/geoclue:/sbin/nologin
nm-openconnect:x:994:993:NetworkManager user for OpenConnect:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/sbin/nologin
gluster:x:993:992:GlusterFS daemons:/run/gluster:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
pipewire:x:992:990:PipeWire System Daemon:/run/pipewire:/usr/sbin/nologin
saslauth:x:991:76:Saslauthd user:/run/saslauthd:/sbin/nologin
chrony:x:990:989:chrony system user:/var/lib/chrony:/sbin/nologin
dnsmasq:x:989:988:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/usr/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
openvpn:x:988:987:OpenVPN:/etc/openvpn:/sbin/nologin
nm-openvpn:x:987:986:Default user for running openvpn spawned by NetworkManager:/sbin/nologin
colord:x:986:985:User for colord:/var/lib/colord:/sbin/nologin
unbound:x:985:984:Unbound DNS resolver:/var/lib/unbound:/sbin/nologin
abrt:x:173:173:/etc/abrt:/sbin/nologin
flatpak:x:984:982:Flatpak system helper:/usr/sbin/nologin
gdm:x:42:42:GNOME Display Manager:/var/lib/gdm:/usr/sbin/nologin
gnome-initial-setup:x:983:981:/run/gnome-initial-setup:/sbin/nologin
vboxadd:x:982:1:/var/run/vboxadd:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/usr/share/empty.sshd:/usr/sbin/nologin
tcpdump:x:72:72:tcpdump:/usr/sbin/nologin
bastien_fedora:x:1000:1000:Bastien_Fedora:/home/bastien_fedora:/bin/bash
clevis:x:981:976:Clevis Decryption Framework unprivileged user:/var/cache/clevis:/usr/sbin/nologin
dhcpcd:x:980:975:Minimalistic DHCP client:/var/lib/dhcpcd:/usr/sbin/nologin
akmods:x:979:973:User is used by akmods to build akmod packages:/var/cache/akmods:/sbin/nologin
bastien_fedora@fedora:~$

```

## 2. Que fait ce script ?

```

#!/bin/bash
while IFS= read -r line
do
echo "$line"
done < "/etc/passwd"

```

On rappelle que les champs de /etc/passwd sont séparés par ' : ' A quoi sert IFS ?



Le script affiche le contenu du fichier /etc/passwd en respectant la disposition ligne par ligne. IFS est une variable d'environnement qui définit le séparateur de champ interne. Par défaut, il est défini sur les espaces, les tabulations et les retours à la ligne.

### 3. Que fait ce script ?

```
#!/bin/bash
IFS=:
while read login mdp uid gid gecoss home shell;
do echo "${gecos:=undef} > login $login (home : $home, shell : $shell)" ;
done < /etc/passwd
```

Le script affiche le contenu du fichier /etc/passwd en respectant la disposition ligne par ligne. Il affiche le login, le home et le shell de chaque utilisateur.

A quoi sert IFS ? IFS =: permet de définir le séparateur de champ interne à ":".

A quoi sert le undef ? il permet de ne pas enregistrer via gecoss \*

### 4. A quoi sert la commande set -x affiche les options de ligne lisibles.