TEST LAB :
Linux Shell/GNU Make

Théo PIERRE, Delphine MALAVAUD, Chloé GAUTHERON - ECE Paris

1 Command line/Environment variables/User permissions

1. In the bash prompt, what is the meaning of the character ~ ?

La ~présente dans le bash prompt signifie que nous sommes dans notre répertoire HOME.

2. Explain the behavior of running in the order VV=3, export VV, bash, unset VV, exit and finally echo $VV ?

VV = 3 ; On affecte la valeur 3 à la variable VV
export VV ; La variable locale VV est exportée pour devenir une variable d'environnement
bash ; Crée un child bash process
unset VV ; Supprime la variable d'environnement en cours d'exécution
exit ; Termine le child bash
echo $VV ; Affiche la variable locale VV

3. How to run /home/user/ls instead of /usr/bin/ls automatically by typing ls without changing the behavior of the other commands ?

Le répertoire /usr/bin est une application ou distribution binaire qui est destinée à être accessible seulement lorsque l'utilisateur est connecté localement. Comme nous l'avons noté dans la question 1, le symbole ~nous permet d'accéder au dossier personnel (home directory). Il faut donc entrer la commande ~**/ls** qui est équivalente à **/home/user/ls.**

4. How to provide file.txt as input of the command flex and copy its stderr to resu.txt ?

Pour rediriger le stderr on utilise la commande « 2> ». Soit :
./file.txt 2> resu.txt

5. Propose a command (in one line) that displays the middle line of /etc/passwd ?

```
chloe@pimped:~$ head -n $((`wc -l </etc/passwd` / 2)) /etc/passwd | tail -n 1
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
```

FIGURE 1 – Displaying middle line

Head commande -n option donne le nombre de ligne que nous voulons voir. $wc -l va chercher la valeur du nombre de ligne dans /etc/passwd que l'on divise par 2. Tail affiche la dernière ligne indiquée.

6. How to change the owner of a first file to the owner of a second one by using a command substitution based on the commands ls and cut ?

Pour changer l'owner du fichier, on va tout d'abord lire le fichier.

Command : ls –l file1
Output : -rw –rw –r – 1 user group . . . file1

Nous savons que *-rw –rw –r –* a pour valeurs binaires *110 110 100* donc notre valeur décimale sera *664*. Soit :

Command : cut –d –file2 664 file1

Ce qui nous permet d'écrire, grâce à 664, dans le file 1 l'owner que l'on a récupéré dans le file 2.

2 The commands grep/sed and regular expressions

1. Propose a grep command that displays user names in /etc/passwd whom UID is multiple of 2 ?

```
chloe@pimped:~$ grep '^[^:]*:[^:]*:[0-9]*[0 2 4 6 8]$*:.*' /etc/passwd | cut -d: -f1
root
bin
sync
man
mail
uucp
backup
list
nobody
systemd-timesync
systemd-resolve
syslog
messagebus
lightdm
avahi
colord
hplip
pulse
rtkit
usbmux
chloe
```

FIGURE 2 – UID multiple 2

Cette commande permet d'afficher tous les UID étant un multiple de 2. On va chercher tous les login x qui ne sont pas :. Lorsque l'on arrive à l'UID on entre [0-9]*[0 2 4 6 8]$ qui va chercher tous les UID dont le premier chiffre est entre 1-9, et finissant par un chiffre pair. Le $ nous permet de prendre tout jusqu'à la fin, * :.* de n'importe quel GID.
Comme nous cherchons simplement les noms de ces UID pairs, on ajoute la commande cut -d : -f1.

2. Propose a grep command that displays, from the output of ifconfig, the WiFi IP address ?

```
attali@attali-VirtualBox:~$ ifconfig lo | grep "inet addr" | cut -d: -f2 | awk '{print $1}'
127.0.0.1
attali@attali-VirtualBox:~$ ifconfig enp0s3 | grep "inet addr" | cut -d: -f2 | awk '{print $1}'
10.0.2.15
```

FIGURE 3 – Wifi IP address

On demande de se concentrer sur le wifi (WLAN 0) et de filtrer parmi les informations sur ce réseau celles concernant l'adresse IP (inet adresse). La commande cut permet ensuite d'afficher ce qui est sélectionné. On ajoute un séparateur de champ « - » à l'aide de la commande –d. Pour finir, awk va simplement afficher toutes les lignes sélectionnées précédemment.

3 Write a Shell script that displays the entries of /etc/passwd, using sed and for loops.

4 GNU Make

Dans cette partie, nous allons détailler étape par étape les actions qui nous ont permises de compiler le programme.
Premièrement, on télécharge le fichier zip du Lab 3 et on y extrait les fichiers C suivants :

FIGURE 4 – MinMax file

Nous les sauvegardons dans un projet MinMax

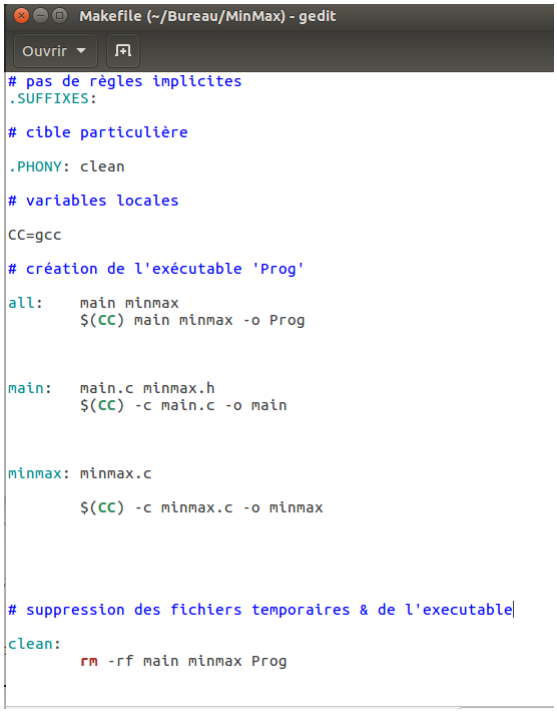

FIGURE 5 – The Makefile

Ensuite on propose et teste le Makefile suivant :

Native compile :

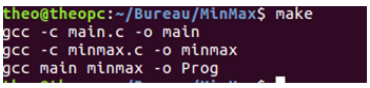

FIGURE 6 – Command make to build

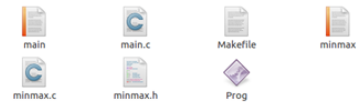

FIGURE 7 – MinMax après le build

On teste notre programme puis on clean le fichier MinMax :

```
theo@theopc:~/Bureau/MinMax$ ./Prog 42 69
Max(42,69)=69
Min(42,69)=42
```

FIGURE 8 – Test MinMax file

```
theo@theopc:~/Bureau/MinMax$ make clean
rm -rf main minmax Prog
```

FIGURE 9 – Clean MinMax file

Arm-cross compile :

On envoie le fichier MinMax à notre RPi.

```
theo@theopc:~/Bureau/MinMax$ sudo scp -r /home/theo/Bureau/MinMax pi@192.168.0.21:/home/pi/newfile
pi@192.168.0.21's password:
minmax.c
minmax.h
Makefile
main.c
```

FIGURE 10 – commande pour l'envoi du fichier MinMax

Nous compilons, testons le programme. Pour finir on nettoie le fichier.

```
pi@raspberrypi:~/newfile/MinMax $ make
gcc -c main.c -o main
gcc -c minmax.c -o minmax
gcc main minmax -o Prog
pi@raspberrypi:~/newfile/MinMax $ ./Prog 69 42
Max(69,42)=69
Min(69,42)=42
pi@raspberrypi:~/newfile/MinMax $ make clean
rm -rf main minmax Prog
```

5 Managing processes/System call fork()

Write a C code where a parent process creates a child process that creates a grandson process using the system call fork().

Code C où un parent process crée un child process qui crée un grandson process

```c
#include <stdlib.h>
#include <stdio.h>
/* Pour fork() */
#include <unistd.h>
/* Pour le type pid_t */
#include <sys/types.h>

const char* jesuis=NULL;

int main()
{
    pid_t pid;
    jesuis="père";
    pid=fork();
    if(pid==0){
        jesuis="fils";
        printf("Je suis %s PID=%d PPID=%d\n",jesuis,getpid(),getppid());
        pid=fork();
        if(pid==0){
            jesuis="petit fils";
            printf("Je suis %s PID=%d PPID=%d\n",jesuis,getpid(),getppid());
        }
        else{
            wait(NULL);
        }
    }
    else{
        printf("Je suis%s PID=%d\n",jesuis,getpid());
        wait(NULL);
    }
    return 0;
}
```

FIGURE 11 – Code C

Ensuite on teste le programme (cf Figure 10) et on remarque que le PID est correctement défini.

```
theo@theopc:~/Bureau$ ./forktest
Je suispère PID=4839
Je suis fils PID=4840 PPID=4839
Je suis petit fils PID=4841 PPID=4840
```

FIGURE 12 – Test PID

La figure ci-dessous explique le programme. On peut remarquer que le parent process attend le child pour finir, grâce à "wait(NULL)". Les process sont donc synchronisés.

FIGURE 13 – Explication du programme