SAé 1.03

GOKCEN Bayram
SAVARY Marian

Installation du poste :

Dans le but d'installer le poste debian, de nombreux choix spécifiques ont été faits. C'est pourquoi nous allons, dès à présent, entrer dans le détail, expliquer, les choix particuliers qui ont été entrepris :

La mémoire RAM à allouer, il s'agit ici d'une partie assez compliquée. En effet, le but de notre machine virtuelle est d'être le plus efficace possible, c'est-à-dire qu'elle doit être rapide. De ce fait, allouer le maximum de mémoire RAM pourrait paraître être une bonne idée, cependant il ne faut pas oublier que la machine virtuelle tourne sur une machine hôte et donc que la Machine hôte (celle physique) nécessite aussi de la mémoire vive : Il faut alors trouver un équilibre. De plus, dès le démarrage, une machine, utilise déjà entre un quart et la moitié de sa mémoire vive pour des tâches de fond.
C'est pourquoi, le juste milieu paraît être un quart de la RAM alloué à la machine virtuelle. En effet, dès lors, entre ¼ et ½ de la mémoire vive serait alloué pour les tâches de fond, ¼ serait alloué pour la Machine virtuelle et le reste pourra servir pour d'autres tâches, il restera même assez de RAM disponible pour éviter une surutilisation de celle-ci.
Finalement, sachant que toutes les machines ne possèdent pas une quantité de mémoire RAM identiques à la base. La quantité en elle-même peut varier fortement, en effet pour un PC avec 16GO de RAM, il serait d'après notre réflexion, avantageux d'allouer 4 GO de RAM, pour un PC avec 8GO : 2GO etc.. Pour les PCs possédant à peu près 4GO de RAM initial, il serait recommandé d'allouer 1GO de RAM pour la machine virtuelle, celà fonctionnera cependant la Machine hôte sera incapable de réaliser d'autres tâches.

La taille du disque dur virtuel, il s'agit en somme de l'espace stockage virtuel dont dispose la Machine Virtuelle. Cet espace dépend de l'utilisation que l'on compte faire de la Machine. En effet, dans le cas où l'on souhaite installer quelques applications pas très lourdes, 16 GO peuvent suffir, si au contraire on souhaite installer de nombreuses applications, on peut allouer plus 32 voire 64 GO par exemple. En effet, les applications pèsent de plus en plus lourd, l'espace de stockage devient de plus en plus rare. Au contraire, si l'on ne souhaite quasiment rien installer, ou seulement utiliser la machine virtuelle pour de petites tâches, 12GO peuvent suffir sur le court terme. En somme, celà dépend de l'utilisation que l'on compte faire de la machine et si on tient la garder sur le long terme.

Le nombre de processeurs alloués dépend de ce que l'on souhaite faire sur la machine virtuelle ainsi que sur la machine physique. Si l'on doit utiliser la machine virtuelle pendant que le système d'exploitation hôte est utilisé pour une autre activité, il peut être préférable d'allouer un petit nombre de processeurs pour la machine virtuelle afin que l'hôte ait ceux restants même si la machine virtuelle est en pleine charge. En revanche, si l'on travaille principalement sur la machine virtuelle, il est alors plus avantageux d'allouer plus de processeurs à celle-ci.
VirtualBox permet dans un premier temps de savoir grâce à une bande de couleur jusqu'à combien de processeurs, il est conseillé, d'aller au maximum.
En somme, si l'on souhaite utiliser la machine virtuelle de façon raisonnable, il est recommandé de mettre 2 voire 3 cœurs de processeur sur la Machine virtuelle. Cependant si l'on souhaite utiliser la machine virtuelle de façon poussée, sans utiliser en même temps la machine hôte, il est

possible de mettre jusqu'à 5 ou 6 cœurs de processeurs. La machine pourra donc être un peu plus efficace mais ceci aux dépens de l'efficacité de la machine hôte.

Une des étapes indispensables de l'installation d'un poste est le partitionnement du disque dur. En effet, il est important de scinder le disque dur en 2 parties, une partie servant de disque de stockage et une deuxième partie "swap".
La première partie représente le stockage brut, l'espace dédié à stocker les données, il est donc important de mettre une grosse part de l'espace total dans cette partie. Le reste de l'espace total est alors utilisé dans une partie swap.
La partition swap est une zone d'un disque dur faisant partie de la mémoire virtuelle de la machine, il s'agit d'un espace d'échange. En effet, la mémoire RAM de la machine virtuelle est utilisée pour stocker des données en cours de traitement. Si celle-ci se remplit presque entièrement mais que la machine a tout de même besoin de ressources pour procéder à des traitements, les pages mémoire seront temporairement déplacées vers l'espace d'échange (swap) défini dans le disque dur afin de libérer des ressources mémoire. L'espace d'échange agit ainsi en tant qu'extension de votre mémoire vive physique : elle récupère, au besoin, des blocs mémoire en excès de la mémoire RAM.
Il est donc avantageux de mettre comme partition swap approximativement 10% de l'espace total. En effet, celà suffit à décharger temporairement la RAM et permet d'avoir suffisamment d'espace de stockage (les 90% restants).

L'environnement de bureau est à choisir en fonction des attentes que l'on a de la machine virtuelle. Il faut alors choisir celui le plus adéquat à l'utilisation que l'on va faire du PC :
Par exemple, si l'on souhaite avoir un environnement de bureau avec beaucoup de personnalisation possible, l'environnement "*KDE PLASMA*" est très extensible, cependant il utilise beaucoup de ressources, il n'est donc pas très efficace en termes de performance. A contrario, si l'on souhaite avoir un environnement efficace, qui n'utilise pas énormément de ressources, l'environnement "*Xfce*" est efficace, c'est celui que l'on a choisi. Néanmoins, en contrepartie, l'aspect graphique de la debian est moins joli, il y a aussi moins de choix de personnalisation. Il existe bien entendu de nombreux autres environnements qui sont plus ou moins efficaces et plus ou moins personnalisables.
En somme, ici, le choix de l'environnement dépend donc des besoins et des ressources de l'utilisateur. En effet, plus la machine virtuelle est puissante, plus elle peut gérer des interfaces lourdes sans forcément avoir de problèmes de lenteur. Cependant la VM ne possède pas de carte graphique, les interfaces surchargées ne sont donc, dans tous les cas, pas très conseillées.

Installer, si l'utilisateur souhaite utiliser un BIOS, le programme de démarrage Grub qui permet quand la machine démarre de choisir quelle système d'exploitation choisir. En soit, il n'est pas très utile de l'installer dans une Machine Virtuelle car il permet à la base de faire du Dual Boot, c'est-à-dire d'utiliser, à la fois, plusieurs systèmes d'exploitation, mais cela peut permettre, au démarrage, de vérifier que c'est le bon système d'exploitation qui se lance.

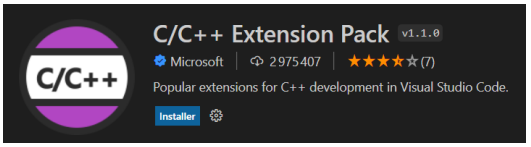
Visual Studio Code :

Nous devons installer dans notre machine des applications comme VsCode. Nous allons alors expliquer rapidement comment l'installer, puis surtout comment le mettre en œuvre :
Pour l'installer, après avoir téléchargé la version ".deb" de VSCode sur le site officiel nous exécutons quelques commandes :
"sudo dpkg -i" suivi du nom du fichier pour nous : "code_1.8.1-1482158209_amd64.deb"

```
sudo dpkg -i code_1.8.1-1482158209_amd64.deb
```

Ici, le nom du fichier peut être un peu différent mais on peut le retrouver dans les téléchargements.

Pour utiliser le langage C++ ou C, nous avons installé le pack “C++ Extension Pack” qui possède toutes les extensions utiles.



- choisir les compilateurs : “Ctrl+Shift+B”

Pour les programmes en C : “/usr/bin/gcc-10”
Pour les programmes en C++ : “/usr/bin/g++”

Nous pouvons alors tester ces langages :

C

```c
int main()
{
    printf("Hello World !\n");
    return 0;
}
```

->

```
Hello World !
```

C++

```cpp
#include <iostream>
using namespace std;

int main()
{
   cout << "Hello world!" <<endl;

   return 0;
}
```

->

```
Hello world!
```

Pour les langages comme le HTML, il n’y a pas d’extension obligatoire à installer. Cependant, il y a de nombreuses extensions utiles, simples à installer si besoin.

LaTex :

Installation :

1 - Télécharger le dossier “install-tl.zip”, disponible sur la page :
http://mirror.ctan.org/systems/texlive/tlnet
(Tous les autres fichiers ne nous servent à rien dans ce cas)

install-tl-windows.exe
install-tl-windows.exe.sha512
install-tl-windows.exe.sha512.asc
install-tl.zip
install-tl.zip.sha512
install-tl.zip.sha512.asc

2 - Aller dans le dossier Téléchargements de la machine virtuelle, puis décompresser le dossier précédemment installé.

3 - dans le terminal du dossier, exécuter la commande “sudo ./install-tl -gui”, il s’agit d’une commande en mode administrateur, il faut donc indiquer son mot de passe.

```
goksav@DebGOKSAV:~/Téléchargements/install-tl-20211129$ sudo ./install-tl -gui
```

4 - Le processus d’installation se lance donc :

1 - aller dans les paramètres avancés et cocher la case “Créer des liens symboliques dans les répertoires systèmes”.

2 - Lancer l’installation.

Utilisation :

Nous pouvons donc essayer d’utiliser LateX pour vérifier qu’il soit bien installer :

```
root@DebGOKSAV:/home/goksav/Téléchargements# pdflatex fichier_pour_latex.tex
```

Nous utilisons pour ça le fichier “fichier_pour_latex.tex” donné précédemment. Il s’agit d’un fichier test qui permet de savoir si nous avons bien compilé le fichier en un fichier pdf.

```
This is pdfTeX, Version 3.141592653-2.6-1.40.23 (TeX Live 2021) (preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./fichier_pour_latex.tex
LaTeX2e <2021-11-15>
L3 programming layer <2021-11-22>
(/usr/local/texlive/2021/texmf-dist/tex/latex/base/article.cls
Document Class: article 2021/10/04 v1.4n Standard LaTeX document class
(/usr/local/texlive/2021/texmf-dist/tex/latex/base/size12.clo))
(/usr/local/texlive/2021/texmf-dist/tex/latex/l3backend/l3backend-pdftex.def)
(./fichier_pour_latex.aux) [1{/usr/local/texlive/2021/texmf-var/fonts/map/pdfte
x/updmap/pdftex.map}] (./fichier_pour_latex.aux) )</usr/local/texlive/2021/texm
f-dist/fonts/type1/public/amsfonts/cm/cmr12.pfb></usr/local/texlive/2021/texmf-
dist/fonts/type1/public/amsfonts/cm/cmr8.pfb>
Output written on fichier_pour_latex.pdf (1 page, 22785 bytes).
Transcript written on fichier_pour_latex.log.
```

Nous pouvons donc nous rendre dans le répertoire où se trouve le fichier .tex. Nous voyons alors qu'un fichier pdf vient d'apparaître !

Nous pouvons alors voir que le fichier a bien été compilé ! Le nouveau fichier pdf est bien le bon.

C'est bien joué, vous venez de compiler votre premier fichier en LaTeX.
Bravo !!!

Création des groupes d'utilisateurs

Nous allons, dans un premier temps, créer 3 groupes d'utilisateurs pour les 3 années. On les nommera "groupe_1", "groupe_2" et "groupe_3"
Pour ceci nous utilisons la commande groupadd sous la forme de root.

```
root@DebGOKSAV:/home/goksav# sudo groupadd groupe_1
root@DebGOKSAV:/home/goksav# sudo groupadd groupe_2
root@DebGOKSAV:/home/goksav# sudo groupadd groupe_3
```

```
groupe_1:x:1001:
groupe_2:x:1002:
groupe_3:x:1003:
```

Nous obtenons, alors, 3 groupes d'utilisateurs avec le gid "1001" pour le groupe 1, "1002" pour le groupe 2 et "1003" pour le groupe 3.

Nous pouvons désormais créer les 3 utilisateurs "tests" qui appartiendront chacun à un groupe d'année :

- Dans un premier temps, nous créons les utilisateurs test :

```
sudo useradd test1 --groups groupe_1 --gid groupe_1 --home /home/test1 --shell /bin/bash
sudo useradd test2 --groups groupe_2 --gid groupe_2 --home /home/test1 --shell /bin/bash
sudo useradd test3 --groups groupe_3 --gid groupe_3 --home /home/test1 --shell /bin/bash
```

- Nous pouvons voir que nous avons bien pensé à les mettre dans le bon groupe.
- Nous pouvons alors vérifier :

```
test1:x:1014:1001::/home/test1:/bin/bash
test3:x:1015:1003::/home/test1:/bin/bash
test2:x:1016:1002::/home/test1:/bin/bash
```

```
root@DebGOKSAV:/home/goksav# id test1
uid=1014(test1) gid=1001(groupe_1) groupes=1001(groupe_1)
root@DebGOKSAV:/home/goksav# id test2
uid=1016(test2) gid=1002(groupe_2) groupes=1002(groupe_2)
root@DebGOKSAV:/home/goksav# id test3
uid=1015(test3) gid=1003(groupe_3) groupes=1003(groupe_3)
```

Nous voyons donc que les utilisateurs "test1", "test2" et "test3" sont bien dans les bons groupes. (groupe_1, groupe_2 et groupe_3)

Nous allons donc, à présent, créer les répertoires propres à chaque année :

Nous allons, dans un premier temps, créer les 3 répertoires :

```
root@DebGOKSAV:/home/goksav# mkdir Annee1
root@DebGOKSAV:/home/goksav# mkdir Annee2
root@DebGOKSAV:/home/goksav# mkdir Annee3
```

Il faut, maintenant, donner aux répertoires le bon groupe pour qu'ils soient utilisables par ces groupes :

```
root@DebGOKSAV:/home/goksav# chgrp groupe_1 Annee1
root@DebGOKSAV:/home/goksav# chgrp groupe_3 Annee3
root@DebGOKSAV:/home/goksav# chgrp groupe_2 Annee2
root@DebGOKSAV:/home/goksav# ls -l
total 52
-rwxr-xr-x 1 root   root      136 15 déc.  16:55 addUsers
d-----xrwx 2 root   groupe_1 4096 15 déc.  17:08 Annee1
d-----xrwx 2 root   groupe_2 4096 15 déc.  17:08 Annee2
d-----xrwx 2 root   groupe_3 4096 15 déc.  17:08 Annee3
drwxr-xr-x 2 goksav goksav   4096 30 nov.  16:37 Bureau
```

Finalement il faut donner aux répertoires les bons droits :

```
root@DebGOKSAV:/home/goksav# chmod 750 Annee1
root@DebGOKSAV:/home/goksav# chmod 750 Annee2
root@DebGOKSAV:/home/goksav# chmod 750 Annee3
root@DebGOKSAV:/home/goksav# ls -l
total 52
-rwxr-xr-x 1 root   root      136 15 déc.  16:55 addUsers
drwxr-x--- 2 root   groupe_1 4096 15 déc.  17:08 Annee1
drwxr-x--- 2 root   groupe_2 4096 15 déc.  17:08 Annee2
drwxr-x--- 2 root   groupe_3 4096 15 déc.  17:08 Annee3
drwxr-xr-x 2 goksav goksav   4096 30 nov.  16:37 Bureau
```

Nous avons alors décidé de donner aux répertoires les droits rwxr - x- - - grâce à la commande "chmod 750"

En effet, de ce fait l'utilisateur qui a créé le groupe (root) peut tout faire, il est alors pour l'administrateur réseaux possible de tout changer dans tous les groupes (car en effet il est censé être le seul ayant possession de compte root).
Les personnes du bon groupe peuvent lire et exécuter les fichiers ou scripts du répertoire, ils peuvent donc, dans leur répertoire, se déplacer, lire et utiliser différents fichiers. En effet, ils ont besoin de ces droits pour utiliser les différents fichiers. Dans notre cas, on ne leur accorde pas le droit d'écriture car il ne leur est pas utile et cela serait dangereux, mais cela peut être changé.
Finalement les personnes autres (qui n'appartiennent pas au groupe) ne peuvent pas interagir, en effet il est inutile pour une personne d'année 1 de regarder dans les fichiers du répertoire Année 2. C'est pourquoi on ne leur accorde aucun droit !
Ces droits nous paraissent en effet judicieusement choisis, mais cela peut être un peu différent pour chaque cas.

Le script de création des comptes utilisateurs :

Une des missions de ce projet est de créer les comptes utilisateurs des différents étudiants. Pour ceci, nous devons écrire un script qui, à l'aide du fichier des nouveaux inscrits, qui nous a été donné, crée les comptes des nouveaux inscrits dans le répertoire adéquat et avec un mot de passe proprement créé sur un schéma défini.

Nous allons donc détailler chaque partie du script :

```
#!/bin/bash
#vérification du nombre de paramètres
if (($#!=1))
        then echo "Erreur : Nombre d'arguments incorrects"
        exit 1
fi
```

Nous commençons par écrire la ligne shebang. Cette ligne sert à indiquer au système quel interpréteur de script il doit utiliser pour exécuter le code contenu dans le fichier.
Par la suite, nous vérifions qu'il y a bien un seul paramètre, en effet nous voulons qu'un seul paramètre puisse entrer dans la fonction (il s'agit du fichier contenant les nouveaux inscrits). Pour ceci, nous vérifions que le nombre de paramètres ($#) soit bien égal à 1. Si il y a plus d'un paramètre ou si il n'y a pas de paramètre un message d'erreur apparaît "Erreur : nombre d'arguments incorrects" et on sort de la fonction avec un code d'erreur 1.

```
#lecture ligne par ligne
while read ligne
        do
```

Etant donné que dans le fichier les inscrits sont écrits ligne par ligne, nous avons décidé de faire une lecture ligne par ligne. Pour ceci, nous utilisons la commande while read. Grâce à ça, chaque ligne rentre dans la variable "ligne" et ce à tour de rôle pour chaque ligne. La boucle s'opère donc à chaque fois pour chaque utilisateur (chaque ligne) jusqu'à la fin du fichier que l'on nous donne.

```
#initialisation de problème (il n'y a aucun problème au début)
((probleme=0))
```

Nous avons décidé, dans le but d'éviter les problèmes, de créer une variable "problème" qui fonctionne comme un booléen. En effet, nous l'avons initialisé à 0 : ce qui signifie qu'il n'y a pas de problème. Plus tard dans le script, si il y a un problème cette variable prend une autre valeur, finalement le compte utilisateur de cette ligne fausse n'est pas créé.

Avant de commencer, la présentation de la boucle, il faut comprendre comment fonctionne le fichier que l'on nous donne. En effet celui-ci est sous la forme :

"nom:prénom:année:numéro téléphone:date_naissance"

On peut donc y retrouver toutes les informations importantes, séparées par des ":"

```
Gokcen:Bayram:3:0777777777:12/04/2003
```

La première étape est donc la création du login de l'utilisateur, il s'agit de son identifiant. Il faut que le login soit constitué de la première lettre du prénom suivi du caractère "_" et ensuite du nom. Pour ce faire nous devons dans un premier temps séparer le nom et le prénom.
Pour ceci nous pouvons couper la ligne et prendre ce qui nous intéresse. Nous coupons donc la ligne à tous les ":", il nous suffit alors de prendre la première partie (-f1) qui est le nom et la deuxième (-f2) qui est le prénom.
Pour obtenir la première lettre du nom, il suffit alors de couper le nom par caractère et de prendre le premier (-c1).
Finalement pour créer le login il suffit de mettre à la suite la première lettre du nom que l'on a précédemment obtenue, suivie de "_", puis finalement le prénom.

```
#création login
nom=$(echo $ligne|cut -d':' -f1)
prenom=$(echo $ligne|cut -d':' -f2)
pl=$(echo $nom|cut -c1)
login=($pl"_"$prenom)
```

Il faut à présent créer le mot de passe :

Nous avons donc décomposé cette étape en plusieurs parties.

Le premier caractère du mot de passe est une lettre du nom au hasard en majuscule. Nous avons, pour obtenir une lettre du nom au hasard, compter, grâce à la commande "wc -c", le nombre de caractères dans le nom. Puis nous avons choisi un chiffre au hasard, entre 0 et le nombre de caractères moins un dans le nom, grâce à la commande RANDOM. Le programme choisit alors le caractère qui se situe à la place d'après dans le nom de la personne (en effet vu que RANDOM prend en compte le 0, nous avons pris le nombre de caractères dans le nom - 1, cependant on ajoute bien un "1 + RANDOM" car il n'existe pas de lettre numéro 0. Il s'agit donc d'une lettre au hasard (parmi toutes les lettres). Finalement, cette lettre passe dans un système qui transforme toutes les lettres minuscules en majuscules. On obtient alors la première partie du mot de passe.

```
#première partie du mdp
ppmdp=$(echo $nom|cut -c $(echo $((1 + RANDOM % $(($(echo $nom|wc -c)-1))))|tr '[a-z]' '[A-Z]')
```

Le deuxième caractère du mot de passe est une lettre du prénom au hasard en minuscule. Cette étape fonctionne presque comme la première. En effet, il suffit de compter, toujours grâce à la commande "wc -c" le nombre de caractères dans le prénom cette fois-ci, puis de choisir un nombre au hasard parmi le nombre de lettres, pour finalement choisir le caractère situé à cette position. Finalement il suffit de cette fois-ci transformer toutes les lettres majuscules en minuscules. On a alors la deuxième partie du mot de passe.

```
#deuxième partie du mdp
dpmdp=$(echo $prenom|cut -c $(echo $((1 + RANDOM % $(($(echo $prenom|wc -c)-1))))|tr '[A-Z]' '[a-z]')
```

La troisième partie du mot de passe est la somme des cinq groupes de deux chiffres constituants le numéro de téléphone.
Pour réaliser cette somme, nous avons dans un premier temps, récupéré le numéro de téléphone de l'utilisateur. Pour ce faire, nous avons comme pour le prénom ou nom, séparé la ligne par les ":", puis récupéré la 4ème partie qui est celle qui nous intéresse.
Puis nous avons fait la somme des groupes, pour ceci nous avons rassemblé les chiffres par groupe. Nous avons dès lors fait un groupe avec les deux premiers chiffres, puis le 3ème avec le 4ème, puis le 5ème avec le 6ème, le 7ème avec le 8ème, finalement les deux derniers chiffres. (cut -c1,2 / cut -c3,4 ... cut -c9,10). Il nous suffit alors de faire la somme de tous ces groupes. Néanmoins petite subtilité, on obtient parfois des nombres comme 08 ou 09, ce qui pose problème, c'est pourquoi nous mettons tous les groupes *"base 10"* grâce à la commande "10#". En effet, nos groupes sont déjà écrits sur une base décimale mais ceci permet surtout d'enlever le 1er "0" qui peut gêner la manipulation.

```
#troisème partie du mdp
num=$(echo $ligne|cut -d':' -f4)
((tpmdp= 10#$(echo $num|cut -c1,2) + 10#$(echo $num|cut -c3,4) + 10#$(echo $num|cut -c5,6) + 10#$(echo $num|cut -c7,8) + 10#$(echo $num|cut -c9,10) ))
```

La 4ème partie du mot de passe est un caractère spécial au hasard. Pour ce faire, nous avons décidé de nous faire notre propre liste de 19 caractères spéciaux, parmi lesquels un va en choisir un. (En effet, on choisit bien un entier entre 0 et 18 auquel on ajoute 1, donc entre 1 et 19 soit 19 choix). Ce caractère constitue alors la 4ème partie du mot de passe.

```
#quatrième partie du mdp
qpmdp=$(echo '%/!?,;*+@\#&~[-].}{'|cut -c $(echo $((1 + RANDOM % 18))))
```

Finalement la dernière partie du mot de passe est la première lettre du mois naissance de l'utilisateur en minuscule. Pour réaliser cette étape, nous avons commencé par récupérer le mois de naissance de l'utilisateur qui est en chiffre. Pour ceci, après avoir comme pour le prénom ou le nom, récupéré la date de naissance séparé par des ":", à la 5ème partie, nous avons coupé une nouvelle fois cette date par rapport aux "/" puis nous avons récupéré la deuxième partie où il y a les mois. Nous avons mis ce mois en base 10 pour enlever le 0 devant le mois qui peut gêner. Nous avons ensuite créé un petit algorithme, une petite case, qui permet de renvoyer la première lettre du mois qu'on lui donne en chiffre. Par exemple, si on lui donne "2", il renvoie "f". Cet algorithme fonctionne de façon à ce que la 5ème partie du mot de passe prend la valeur adéquate par rapport au mois qu'on lui donne. Il faut donc le programmer pour tous les mois.

```
#cinquième partie du mdp
mois=10#$(echo $ligne|cut -d':' -f5|cut -d'/' -f2)
if (($mois==1 || $mois==6 || $mois==7))
        then cpmdp="j"
elif (($mois==2))
        then cpmdp="f"
elif (($mois==3 || $mois==5))
        then cpmdp="m"
elif (($mois==4 || $mois==8))
        then cpmdp="a"
elif (($mois==9))
        then cpmdp="s"
elif (($mois==10))
        then cpmdp="o"
elif (($mois==11))
        then cpmdp="n"
elif (($mois==12))
        then cpmdp="d"
else ((probleme=1))
fi
```

Par exemple, si le mois vaut 1, 6 ou 7, le programme renvoie “j”, si le mois vaut 3 ou 5, il renvoie “m”, etc.. De plus si le moins ne vaut pas une valeur normale donc un entier entre 1 et 12, la variable problème intervient, elle prend alors la valeur 1.

Une des variables importantes est l’année d’étude de l’étudiant, en effet il s’agit de ce qui va déterminer dans quel répertoire va aller l’étudiant.
Nous pouvons alors, simplement, récupérer la 3ème partie de la ligne séparée par des “:” (*cut -d’:’ -f3*) que nous mettons dans une base décimale, pour enlever le premier “0”, par exemple si l’année est écrite de cette façon : “01”.

```
((annee=10#$(echo $ligne|cut -d':' -f3)))
```

Nous avons alors pensé à mettre un petit algorithme qui vérifie que l’année est bien utilisable. En effet, les élèves vont soit en Année 1, 2 ou 3 pas 4 ou 0. Si il y a un problème la variable problème prend la valeur 2.

```
#vérification d'une bonne année
if (($annee!=1 && $annee!=2 && $annee!=3))
        then ((probleme=2))
fi
```

Le login et le mot de passe étant maintenant réalisés, nous allons désormais créer s’ils n’existent pas les fichiers qui contiennent toutes les informations par rapport aux utilisateurs.

```
#création des fichiers (qui contiendront les mdps)
if test ! -f /home/etudiants/Annee"$annee"/Utilisateurs_A"$annee" && (($probleme==0))
        then touch /home/etudiants/Annee"$annee"/Utilisateurs_A"$annee"
                chmod 600 /home/etudiants/Annee"$annee"/Utilisateurs_A"$annee"
fi
```

Pour ce faire, nous vérifions, dans un premier temps, s' il existe un fichier nommé Utilisateurs_A“année” dans le répertoire Annee“année” où “année” est l’année d’étude de l’étudiant. Dans un seconde temps, si ce n’est pas le cas et si problème vaut 0 donc qu’il n’y a pas de problèmes, un fichier Utilisateurs_A“année” est créé avec chmod 600, de ce fait seul root peut interagir avec le fichier où il y a les mots de passe.
De ce fait les fichiers contenant les mots de passe des étudiants de différentes années sont créés, cependant ceci est vrai que dans le cas où il y a au moins un étudiant dans l’année en question sinon rien ne se crée.

```
#assemblage du mdp
mdp=("$ppmdp$dpmdp$tpmdp$qpmdp$cpmdp")
```

Le mot de passe est ensuite, simplement, assemblé grâce aux différentes parties et il est enregistré dans la variable “mdp”. Cette variable est une des clés du script car c’est elle qui va permettre de donner leur mot de passe aux utilisateurs et c’est aussi elle qui est écrite dans le fichier contenants les mots de passe de tous les utilisateurs (d’un certain groupe)

Finalement, la dernière étape et la plus importante est l'étape de création des comptes, en effet, il faut désormais créer des comptes, les ajouter à un groupe, leur créer un répertoire, leur attribuer un mot de passe et finalement entrer leurs coordonnées dans le fichier créé précédemment.

```
#vérification des problèmes
#création des comptes
if ((probleme==0))
then echo "Ajout de l'utilisateur $login"
        useradd $login --groups groupe_"$annee" --gid groupe_"$annee" -m -d /home/etudiants/Annee"$annee"/$login
        echo "$login:$mdp" | chpasswd
        echo "$nom" "$prenom" "$login" "$mdp" >> /home/etudiants/Annee"$annee"/Utilisateurs_A"$annee"
        chmod 700 /home/etudiants/Annee"$annee"/$login
#message d'erreur
else echo "Erreur dans l'ajout de $login"
fi
```

Nous commençons, dans un premier temps, par faire une vérification des problèmes, en effet la partie de création des comptes se lance que si probleme=0, donc si il n'y a aucun problème. Dans le cas contraire, un message d'Erreur apparaît disant qu'il y a erreur dans l'ajout de l'utilisateur et le script passe au prochain utilisateur.

Si la variable probleme vaut bien 0 et qu'il n'y a alors aucun problème, nous pouvons ajouter l'utilisateur. Pour ceci nous utilisons la commande useradd :
Nous donnons dans un premier temps, à l'utilisateur, son login que l'on a créé plus haut. Puis nous lui donnons le groupe et le gid de son groupe d'année il s'agit des groupes 1, 2 ou 3 créés précédemment. (il suffit d'utiliser la variable annee que l'on a créé juste avant). Finalement nous créons, à l'utilisateur, un répertoire home dans le bon répertoire "/home/etudiants/Annee … / "login". Nous avons aussi mis un "chmod 700" dans les répertoires des utilisateurs pour que seul l'utilisateur puisse y accéder, ça parait logique que tout le monde ne puisse pas se balader chez tout le monde.

La seconde étape est de donner à l'utilisateur son mot de passe. pour ceci nous utilisons la commande echo "$login:$mdp"|chpasswd. Elle permet de donner un mot de passe à l'utilisateur depuis un script, ici, login est bien le nom de l'utilisateur et mdp est bien le mot de passe assemblé précédemment. L'utilisateur précédemment ajouté reçoit donc bien son mot de passe adéquat

Finalement, la dernière étape consiste à écrire, dans le fichier créé dans l'étape d'avant, le nom, le prénom, le login et le mot de passe des utilisateurs, ceci doit être fait à la suite les uns des autres. Nous avons donc décidé de faire un "echo" avec toutes les informations demandées (donc les variables "nom" "prénom" "login" et "mdp") et ceci en mode ajout dans le fichier adéquat. En effet, on utilise ">>" pour ajouter les utilisateurs à la ligne et non pas supprimer les précédents et on repère le bon fichier grâce à la variable "annee"

La dernière partie du script est une partie qui donne l'entrée du script. Ici l'entrée est bien le seul paramètre possible. Il s'agit du fichier que l'on nous donne au début avec les nouveaux inscrits.

```
#entrée du script
done < $1
```