

TP n°1 - Découverte de l'environnement de travail

I. Système d'exploitation Linux

Cette année, nous allons travailler sur des machines sous Linux. Linux est un système d'exploitation, comme le sont Mac OS et Windows. Un **système d'exploitation** est une couche logicielle (*software*) intermédiaire permettant de faire l'interface entre l'utilisateur d'une machine et le matériel physique (*hardware*).

I. 1. Chemin d'accès d'un fichier ou d'un répertoire

L'ensemble des fichiers et répertoires (aussi appelés dossiers) sont rangés sous la forme d'un arbre. On parle de l'**arborescence de fichiers**. Les répertoires sont les nœuds de l'arbre et les fichiers sont les feuilles. Nous reviendrons sur la structure d'arbre informatique au deuxième semestre. La Figure 1 montre une représentation de cette arborescence. Il y a un unique chemin dans l'arbre pour accéder à un fichier ou à un répertoire donné.

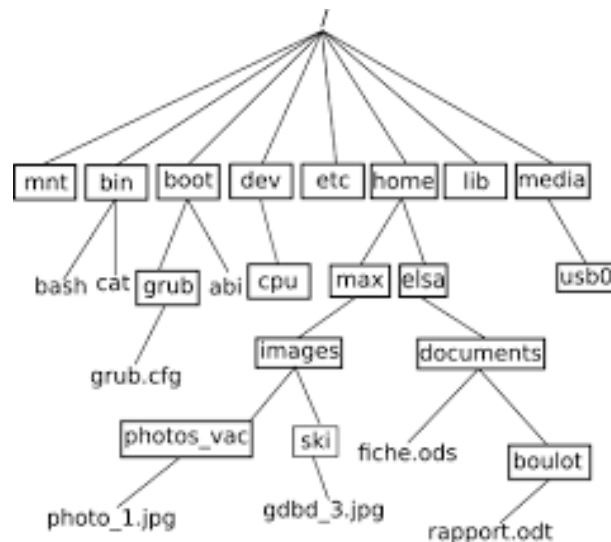


FIGURE 1 – Représentation de l'arborescence de fichiers sous Linux.

Pour faire référence sans ambiguïté à un répertoire ou à un fichier, il ne faut pas seulement donner son nom, il faut donner son **chemin d'accès** (*path* en anglais) dans l'arborescence de fichiers. Il existe deux façons de donner un chemin d'accès :

Chemin d'accès absolu : On part de la racine de l'arborescence de fichiers, puis on donne tous les nœuds par lesquels il faut passer pour atteindre le fichier ou le répertoire ciblé. Par exemple, pour accéder au fichier `photo_1.jpg`, le chemin d'accès absolu est :

`/home/max/images/photos_vac/photo_1.jpg`

Chemin d'accès relatif : On part du répertoire où l'on se trouve (on appelle cela le répertoire courant ou répertoire de travail), puis on donne tous les nœuds par lesquels il faut passer en partant du répertoire courant pour arriver jusqu'au fichier ou jusqu'au répertoire ciblé. Pour remonter d'un cran dans l'arbre, j'utilise la notation `..`. Par exemple, si je suis dans le répertoire `ski` et que je veux accéder au fichier `photo_1.jpg`, le chemin d'accès relatif est :

`../photos_vac/photo_1.jpg`

Voici un petit lexique anglais-français

directory : répertoire, aussi appelé dossier en français

file : fichier

path : chemin d'accès d'un fichier ou d'un répertoire, permettant de le localiser précisément dans l'arborescence de fichiers

hardware : matériel physique, hard = dur

software : logiciel, soft = doux, mou

I. 2. Arborescence de fichiers sous Linux

Dans la hiérarchie de fichiers, il y a les **fichiers et répertoires utilisateur** (photos, vidéos, documents, programmes ... créés par l'utilisateur) et il y a les **fichiers et répertoires systèmes**, qui sont ceux utiles pour le bon fonctionnement du système d'exploitation.

Ces fichiers et répertoires systèmes sont toujours plus ou moins organisés de la même manière sous Linux.¹

Dans le tableau suivant nous citons les principaux répertoires du système Linux avec une brève description de leur contenu. On peut les retrouver sur toutes les machines fonctionnant sous Linux :

/bin/ : doit contenir toutes les commandes de base nécessaires au démarrage et à l'utilisation d'un système minimaliste (par exemple : `cat`, `ls`, `cp`, `sh`) excluant les commandes systèmes et celles réservées aux administrateurs (qui sont plutôt placées dans **/sbin/**). Le nom du répertoire est tiré de l'abréviation de binaires.

/dev/ : fichiers correspondant (directement ou non) avec un périphérique (abréviation de *device*)

Les fichiers de périphériques : périphériques physiques (disque, réseau, bande, disquette) périphériques virtuels ; **/dev/null** **/dev/zero**

/etc/ : fichiers de configuration (abréviation de *editable text configuration*).

/home/ : répertoires des utilisateurs (exemple : **/home/max** est le répertoire de l'utilisateur nommé 'max' dans la Figure 1). Le répertoire de l'utilisateur qui est en train d'utiliser la machine est également nommé .

/lib/ : bibliothèques logicielles nécessaires pour les exécutables de **/bin/** et **/sbin/** (abréviation de *libraries*, signifiant bibliothèques en français).

/mnt/ : point de montage pour les systèmes de fichiers temporaires (abréviation de *mount* ou point de montage).

/media/ : point de montage pour les médias amovibles. Parmi les médias amovibles, il y a notamment les CD-ROM et les clés USB.

/usr/ : contient certains répertoires semblables à ceux présents à la racine mais qui ne sont pas nécessaires au fonctionnement minimal du système (**/usr/** est l'acronyme de *unix system resources*).

/usr/bin/ : binaires exécutables qui ne sont pas déjà présents dans **/bin** et donc pas indispensables à un système minimaliste.

/usr/include/ : entêtes des bibliothèques partagées

/usr/lib/ : bibliothèques partagées entre différents programmes

Exercice 1 (Découverte de l'arborescence de fichiers Linux).

1. Localisez où se trouvent les fichiers exécutables correspondant aux commandes `more` et `cd` dans l'arborescence de fichiers de votre machine.
2. Trouvez, dans l'arborescence de fichier, le fichier contenant les informations relatives au microprocesseur (CPU *Central Processing Unit*) de votre machine. Afficher son contenu dans le terminal. Notez la marque, le model, la fréquence d'horloge et le nombre de coeurs de votre CPU.

II. Terminal de commande

II. 1. Définition

Chaque système d'exploitation peut recevoir des commandes de l'utilisateur. Ces commandes peuvent être réalisées par l'intermédiaire

1. https://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

d'une interface graphique avancée : souvent sous la forme d'un système de fenêtres graphiques où les clics de la souris à un endroit précis dans une fenêtre donnée permettent de lancer une action précise.

d'un terminal de commande : une fenêtre graphique dans laquelle s'exécute un interpréteur en ligne de commande appelé **shell**. Ce dernier fournit un certain nombre de commandes pour interagir avec le système d'exploitation. Cette année, nous allons privilégier le lancement de commande par le terminal de commande, car il permet de comprendre plus en profondeur ce que l'on fait faire à la machine.

II. 2. Fonctionnement du terminal de commande.

Après avoir lancé le terminal, une fenêtre attend que l'utilisateur saisisse l'une de ces commandes. Un **prompt** est affiché, de la forme `nom_utilisateur@nom_machine $` qui invite l'utilisateur à taper une commande à envoyer au système d'exploitation. L'utilisateur doit taper une commande valide et appuyer sur entrée pour lancer son exécution.

Même si de prime abord, l'utilisation du terminal semble moins aisée que celle d'une interface graphique à l'aide d'une souris, elle est généralement beaucoup plus efficace et plus simple pour effectuer des actions qui nécessiteraient de nombreux clics de souris, voire impossible à faire dans une interface graphique.

II. 3. Commandes de base Linux

Une commande Linux s'écrit de la façon générale suivante :

`nom_commande options facultatives paramètres_d'entrée_obligatoires`

Voici quelques commandes de base très utiles :

clear : permet d'effacer le contenu de l'écran pour ne laisser qu'un terminal vide avec son prompt. La combinaison de touches [ctrl] + [l] fait la même chose 7

exit : permet de terminer la session en cours et de fermer la fenêtre du terminal.

help : affiche une liste de commandes shell disponibles. Suivie du nom d'une commande, **help** renvoie une courte description de la commande.

man : suivi du nom d'une commande, affiche une description complète (le **manuel**) d'une commande. Pour quitter l'affichage, appuyer sur la touche q.

which : suivi du nom d'une commande, permet de savoir où se situe le programme exécutable correspondant dans la hiérarchie de fichiers

history : affiche l'historique des 50 dernières commandes saisies. L'utilisation des touches fléchées vers le haut ou vers le bas permet également de parcourir l'historique de ces commandes, de la plus récente à la plus ancienne.

pwd : (**p**rint **w**orking **d**irectory) affiche le nom du répertoire de travail actuel.

ls : affiche le contenu du répertoire courant : fichiers et dossiers. Elle peut être suivie d'options pour préciser la nature des informations à afficher : `ls -l` , `ls -a` , `ls -la` , `ls -rla` etc.

cd : permet la navigation dans l'arborescence des répertoires. suivie du nom d'un répertoire, la commande permet de déplacer le répertoire courant vers ce nouveau répertoire.

- suivie d'aucun argument, elle permet de revenir au répertoire principal de l'utilisateur (répertoire home).
- suivie de `-` , elle renvoie au répertoire précédent.
- suivie d'une espace et de deux points, elle fait remonter d'un niveau dans l'arborescence des fichiers.

mkdir : (**m**ake **d**irectory) suivie d'un nom de répertoire, cette commande crée un nouveau répertoire.

rmdir : (**r**emove **d**irectory) suivi du nom d'un répertoire vide, supprime ce répertoire. Pour supprimer un répertoire et tous les fichiers et sous-dossiers qu'il contient, il faut rajouter l'option de commande **-r** (r pour récursivement) : `rm -r nom_repertoire`

sudo : (**s**uper **u**tilisateur **d**o) permet d'exécuter une commande avec les droits d'un autre utilisateur. Le mot de passe de l'utilisateur ciblé est alors demandé. Si la commande est saisie sans nom d'utilisateur, c'est le super-utilisateur **root** qui est ciblé. Suivi de l'option `-u` et du nom de l'utilisateur, ce dernier est ciblé. Noter qu'un changement d'utilisateur n'est autorisé que si l'administrateur de la machine vous en donne les droits

cat : suivi du nom d'un fichier, affiche son contenu
touch : suivi d'un nom, crée un nouveau fichier vide portant ce nom
cp : (copy) **cp chemin1 chemin2** permet de copier le contenu du fichier ayant le chemin d'accès **chemin1** dans un nouveau fichier qui aura le chemin d'accès **chemin2**
mv : (move) **mv chemin1 chemin2** permet de déplacer le fichier ayant le chemin d'accès **chemin1** pour qu'il ait le nouveau chemin d'accès **chemin2**
rm : (remove) **rm chemin** efface le fichier de ayant le chemin d'accès **chemin**

Exercice 2 (Découverte des commandes Linux).

1. Déplacez vous à la racine de votre répertoire utilisateur ou assurez vous que vous y êtes déjà
2. Créer le répertoire TP1
3. Aller lire la page de manuel de la commande **ls**
4. Vérifiez que ce répertoire a bien été crée en lisant tous les éléments contenus dans votre répertoire utilisateur.
5. Déplacez vous dans le dossier TP1
6. Créer un fichier vide en le nommant **TP1-toplevel.ml**
7. Créer un autre fichier vide en le nommant **TP1.c**
8. Revenir au répertoire du dessus et listez tous les éléments de votre arborescence en partant de ce point.

A retenir : les fichiers texte correspondant à des programmes écrits en langage C porteront l'extension **.c**, et ceux correspondant à des programmes écrits en langage OCaml porteront l'extension **.ml**.

III. L'éditeur de texte emacs

Emacs est un éditeur de texte très utilisé par les programmeurs sous Linux. Ce n'est pas un logiciel de traitement de texte comme Word, dont le but est de mettre en forme du texte. Il s'agit d'un logiciel permettant simplement d'écrire du texte. Emacs possède aussi des fonctionnalités d'IDE (*Integrated Development Environment*) puisqu'il permet de développer du code de façon confortable (coloration syntaxique adaptée à de nombreux langages de programmation, possibilité d'exécuter du code ligne à ligne par exemple)

Dans une fenêtre Emacs (une instance), on peut ouvrir plusieurs fichiers en même temps dans ce que l'on nomme des **buffer**²

Emacs ne fonctionne quasiment pas avec des clics de souris mais avec des raccourcis claviers. Nous donnons ci-dessous la liste des principales commandes, à s'approprier rapidement car elles vous serviront durant toute la prépa et même après!

C-a se lit « *contrôle a* » et signifie que le raccourci clavier consiste à appuyer en même temps sur la touche **Ctrl** du clavier et sur la touche **A**.

M-x se lit « *alt x* » et signifie que le raccourci clavier consiste à appuyer en même temps sur la touche **Alt** du clavier et sur la touche **x**.

2. *buffer* est un mot anglais signifiant tampon, car ce que l'on modifie dans emacs est en fait un fichier temporaire qui va servir de tampon entre la version initiale du fichier et la version finale que nous enregistrerons à la fin de nos modifications.

Abréviation	Nom de la commande	Commande
C-x C-f	find-file	ouvrir un fichier
C-x C-s	save-buffer	sauvegarder le tampon courant
C-x C-w	write-file	sauvegarder le tampon sous un nouveau nom
C-a	beginning-of-line	aller en début de ligne
C-e	end-of-line	aller en fin de ligne
C-v	scroll-up	descendre d'une page écran
M-v	scroll-down	remonter d'une page écran
M-<	beginning-of-buffer	Aller au début du document
M->	end-of-buffer	aller à la fin du document
C-k	kill-line	couper la ligne
C-ESPACE		permet de démarrer une sélection au point où se trouve le curseur
C-w	kill-region	couper la région sélectionnée
C-y	yank	coller la région sélectionnée
C- -	undo	annuler l'action précédente
C-s	isearch-forward	rechercher en aval
C-r	isearch-backward	rechercher en amont
M-%	query-replace	rechercher et remplacer
C-x b	switch-to-buffer	changer de tampon
C-x k	kill-buffer	détruire un tampon
C-x 1	delete-other-windows	ne garder qu'une seule fenêtre
C-x 2	split-window-vertically	découper la fenêtre horizontalement
C-x 3	split-window-horizontally	découper la fenêtre verticalement
C-x 5 2	make-frame	ouvrir une nouvelle fenêtre
C-x 5 0	delete-frame	fermer la fenêtre
M-g	goto-line	aller à la ligne désirée
C-x C-c	save-buffers-kill-emacs	quitter emacs
C-g	keyboard-quit	interrompre la commande en cours
M-x		saisir une commande depuis son nom
C-h ?	help	aide

FIGURE 2 – Principales commande emacs, avec leur raccourci clavier.

Exercice 3 (Découverte d'Emacs).

1. Pour commencer, téléchargez le fichiers `emacs` que vous avez utilisé pour votre installation sur votre machine personnelle.
2. Copier ce fichier à la racine de votre répertoire utilisateur en le renommant `.emacs`
3. Créez un dossier `doc` à la racine de votre répertoire utilisateur
4. Dans ce nouveau dossier, tapez la commande suivante, pour à la fois créer le fichier `TP1_tests_emacs.txt` et l'ouvrir avec Emacs.

```
emacs TP1_tests_emacs.txt
```

5. Faites en sorte qu'il n'y ait plus qu'un seul buffer visible dans votre fenêtre Emacs
6. Tapez le texte suivant : *J'aime la prépa !*
7. Utilisez les raccourcis clavier d'emacs pour copier-coller cette même phrase sur les 3 lignes suivantes
8. Annulez la dernière action pour n'avoir plus que trois lignes avec ce même texte
9. Remplacez toutes les occurrences (= les apparitions) du mot *prépa* par le mot *fête* en utilisant les raccourcis clavier d'Emacs
10. Enregistrez les modifications effectuées dans votre fichier
11. Quittez le logiciel Emacs

Exercice 4 (Un premier programme en Python exécuté en ligne de commande).

1. Assurez vous d'abord que Python3 est bien installé sur votre machine en tapant simplement

```
> python3
```

Si ce n'est pas le cas, installez-le avec le gestionnaire de paquets `apt-get`

```
> sudo apt-get install python3
```

Installez le gestionnaire de paquets (bibliothèques) Python appelé `pip` en tapant cette ligne de commande :

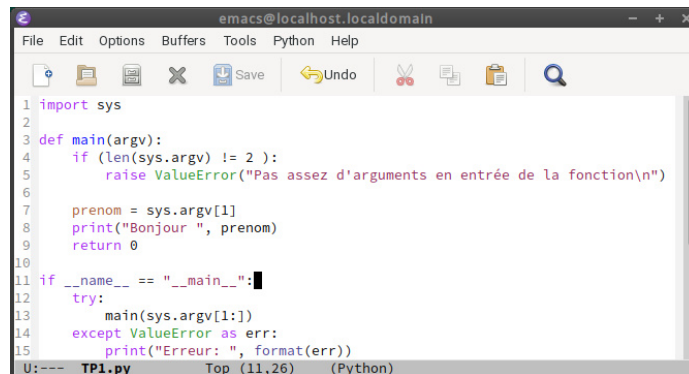
```
> sudo apt install python3-pip
```

2. Installez ensuite grâce à `pip` les bibliothèques `numpy` (calcul scientifique) et `matplotlib` (création de graphiques) qui vous serviront beaucoup ces deux années (informatique, physique, TIPE...)

```
> pip install numpy
```

```
> pip install matplotlib
```

3. A partir de votre instance d'Emacs, ouvrez un nouveau fichier de chemin `/TP1/TP1.py`
4. Taper le code suivant en essayant de deviner le rôle de chaque ligne de code :



```
1 import sys
2
3 def main(argv):
4     if (len(sys.argv) != 2):
5         raise ValueError("Pas assez d'arguments en entrée de la fonction\n")
6
7     prenom = sys.argv[1]
8     print("Bonjour ", prenom)
9     return 0
10
11 if __name__ == "__main__":
12     try:
13         main(sys.argv[1:])
14     except ValueError as err:
15         print("Erreur: ", format(err))
```

Observez également la coloration syntaxique (mise en couleur de certains éléments particuliers du texte). Sur vos notes, indiquez quelle couleur correspond à quels éléments du langage.

5. Nous allons maintenant faire **exécuter** le code par l'**interpréteur** python3 présent sur votre machine.

Pour faire cela, tapez la commande Linux suivante dans le terminal après vous être placé dans le dossier où se trouve votre programme Python :

```
python3 ./TP1.py votre_prenom
```

6. Dupliquez le fichier de code grâce à une commande Linux et ouvrez le fichier dupliqué directement depuis Emacs en utilisant un raccourci clavier
7. Modifiez le code Python pour que l'utilisateur puisse également donner son nom de famille et que le programme l'affiche dans son message de bonjour. Testez bien sûr votre code !

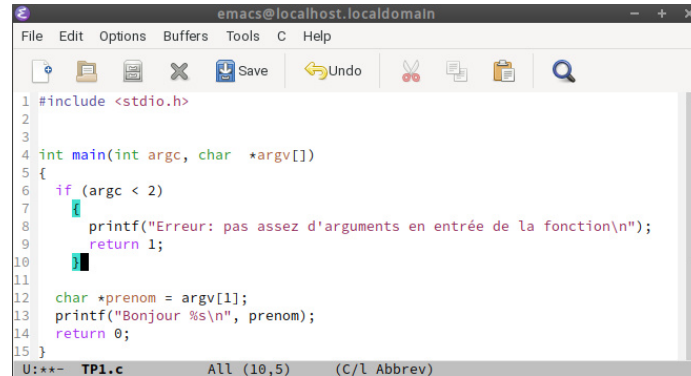
Exercice 5 (Un premier programme en C de A à Z).

Nous allons écrire le même programme mais cette fois-ci en langage C.

1. Assurez-vous d'abord que le compilateur `gcc` est bien installé sur votre compte. Si ce n'est pas le cas, installez-le, ainsi que toutes les bibliothèques essentielles et le débogueur `gdb`, en tapant :

```
> sudo apt install build-essential
```

2. A partir de votre instance d'Emacs, ouvrez le fichier vide de chemin `/TP1/TP1.c` que vous avez créé dans un exercice précédent.
3. Tapez le code suivant en essayant de deviner le rôle de chaque ligne de code :



```
1 #include <stdio.h>
2
3
4 int main(int argc, char *argv[])
5 {
6     if (argc < 2)
7     {
8         printf("Erreur: pas assez d'arguments en entrée de la fonction\n");
9         return 1;
10    }
11
12    char *prenom = argv[1];
13    printf("Bonjour %s\n", prenom);
14    return 0;
15 }
```

Observez également la coloration syntaxique (mise en couleur de certains éléments particuliers du texte). Sur vos notes, indiquez quelle couleur correspond à quels éléments du langage.

4. Nous allons maintenant **compiler** le code, c'est-à-dire le transformer en langage binaire compréhensible par le processeur de l'ordinateur. Nous allons utiliser pour cela le compilateur `gcc`, qui a été installé sur votre machine.

Pour compiler votre petit code en C, taper la commande Linux suivante dans le répertoire TP1 :

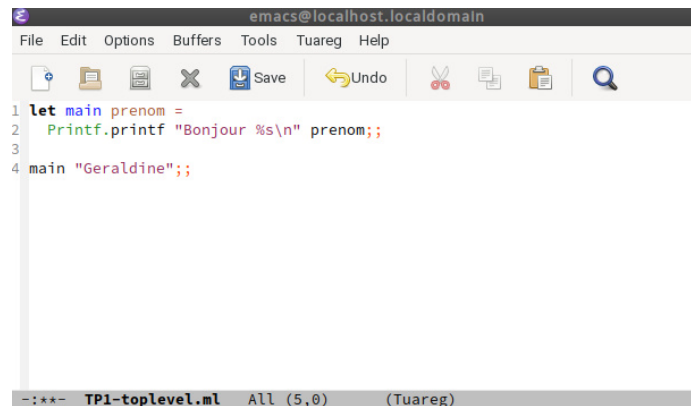
```
gcc ./TP1.c -o TP1
```

Cette ligne va générer un nouveau fichier appelé TP1 qui est un fichier exécutable par la machine. Notez que l'option de commande `-o` permet de spécifier le nom du fichier exécutable créé en sortie de la commande.

5. Enfin, nous allons exécuter ce programme. Pour cela, il faut taper le nom du fichier exécutable suivi du ou des arguments d'entrée attendus par le programme. A vous de jouer.
6. Dupliquez le fichier de code grâce à une commande Linux et ouvrez le fichier dupliqué directement depuis Emacs en utilisant un raccourci clavier
7. Modifiez le code C pour que l'utilisateur puisse également donner son nom de famille et que le programme l'affiche dans son message de bonjour.

Exercice 6 (Un premier programme en OCaml exécuté pas à pas en mode interactif (mode console = REPL = toplevel)).

1. A partir de votre instance d'Emacs, ouvrez un nouveau fichier de chemin `/TP1/TP1-toplevel.ml`
2. Tapez le code suivant en essayant de deviner le rôle de chaque ligne de code :



```
1 let main prenom =
2   Printf.printf "Bonjour %s\\n" prenom;;
3
4   main "Geraldine";;
```

Observez également la coloration syntaxique (mise en couleur de certains éléments particuliers du texte). Sur vos notes, indiquez quelle couleur correspond à quels éléments du langage.

3. Nous allons maintenant exécuter pas à pas les instructions écrites en OCaml en utilisant la console interactive Ocaml. Celle-ci peut être lancée dans un Terminal en appelant la commande `ocaml`.

Comme l'interpréteur est en mode interactif, le toplevel répond à chaque demande d'exécution en indiquant l'effet de l'action exécutée. D'une certaine façon, le toplevel d'OCaml ressemble beaucoup à la console interactive Python que vous connaissez bien. Par contre, le langage de programmation OCaml est très différent de Python !

La console interactive est aussi appelée **toplevel** ou REPL (*Read Eval Print Loop*). Elle peut être lancée de deux manières :

Dans un Terminal en ligne de commande en appelant simplement la commande `ocaml` sans arguments.

Directement dans Emacs : grâce au module additionnel Tuareg qui est installé sur votre machine.

Dans Emacs, le mode Tuareg permet d'appeler la commande `ocaml` sur chaque ligne de code pour exécuter pas à pas le programme écrit en OCaml. Pour cela, il faut se placer au début de chaque instruction dans l'ordre et appuyer à chaque fois sur les touches **C-x C-e**.

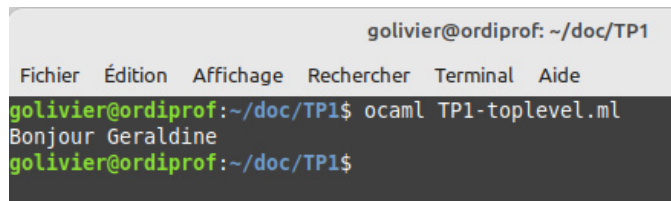
Testez ces deux méthodes sur le code OCaml que vous venez d'écrire.

4. Modifiez le code pour qu'il affiche votre prénom et non le mien.
5. Modifiez le code pour qu'il puisse afficher également le nom, que l'on pourra modifier dans la dernière commande.

Exercice 7 (Le même code OCaml exécuté en non-interactif par l'interpréteur).

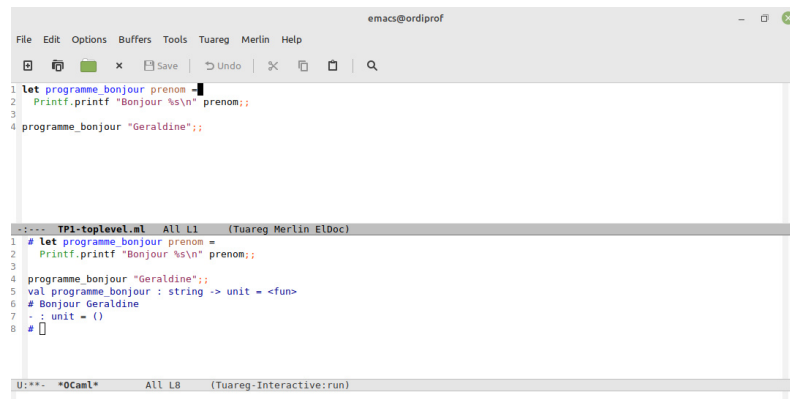
Tout comme dans le cas de l'interpréteur `python3`, on peut aussi lancer le script OCaml en une seule fois, en mode non-interactif. Cela peut être fait de deux manières :

Dans un Terminal de commande, tapez la commande suivante pour exécuter le script en une seule fois en mode non-interactif (même syntaxe qu'avec `python3`) :



```
golivier@ordiprof: ~/doc/TP1
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
golivier@ordiprof:~/doc/TP1$ ocaml TP1-toplevel.ml
Bonjour Geraldine
golivier@ordiprof:~/doc/TP1$
```

Directement dans Emacs : il est possible d'exécuter le script « tout d'un coup » grâce au mode Tuareg en utilisant le raccourci clavier `C-c C-b`.



```
File Edit Options Buffers Tools Tuareg Merlin Help
[Icons] Save Undo [Icons] Search

1 let programme_bonjour prenom =
2   Printf.printf "Bonjour %s\n" prenom;;
3
4 programme_bonjour "Geraldine";;

-:--- TP1-toplevel.ml All L1 (Tuareg Merlin ElDoc)
1 # let programme_bonjour prenom =
2   Printf.printf "Bonjour %s\n" prenom;;
3
4 programme_bonjour "Geraldine";;
5 val programme_bonjour : string -> unit = <fun>
6 # Bonjour Geraldine
7 -: unit = ()
8 #

U:*** *OCaml* All L8 (Tuareg-Interactive:run)
```

Testez ces deux méthodes sur le code OCaml que vous venez d'écrire.

Exercice 8 (Un premier programme OCaml exécuté après compilation).

1. A partir de votre instance d'Emacs, ouvrez un nouveau fichier de chemin `/TP1/TP10CamlCompile.ml`
2. Tapez le code suivant en essayant de deviner le rôle de chaque ligne de code :



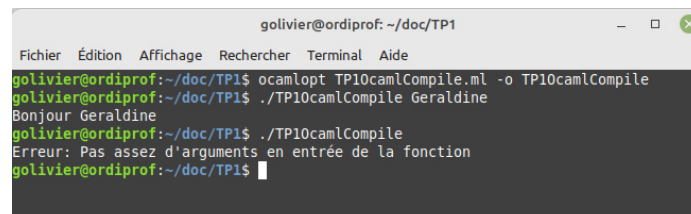
```
1 let main() =
2   let n_arg = Array.length Sys.argv in
3   if n_arg != 2 then
4     failwith "Pas assez d'arguments en entrée de la fonction\n"
5   else
6     let prenom = Sys.argv.(1) in Printf.printf "Bonjour %s\n" prenom;
7   in
8   if !Sys.interactive == true then
9     ()
10  else
11    try
12      main ()
13    with Failure error_string -> Printf.printf("Erreur: "); print_string(error_string);;
```

3. Nous allons maintenant **compiler** le code comme nous l'avons fait pour notre code écrit en C. Nous allons utiliser pour cela le compilateur `ocamlopt`, qui a été installé sur votre machine. Pour compiler votre petit code du langage OCaml vers le langage machine binaire, tapez la commande Linux suivante :

```
ocamlopt ./TP10camlCompile.ml -o TP10camlCompile
```

Cette ligne va générer un nouveau fichier appelé `TP10camlCompile` qui est un fichier exécutable binaire. Notez que l'option de commande `-o` permet de spécifier le nom du fichier exécutable créé en sortie de la commande, comme pour la commande `gcc`. Il est fréquent que certaines options ayant des effets similaires soient nommées de la même manière pour des commandes différentes, ceci afin de rendre intuitif les commandes Linux.

4. Enfin, nous allons exécuter ce programme. Pour cela, il faut taper le nom du fichier exécutable suivi du ou des arguments d'entrée attendus par le programme. A vous de jouer.



```
golivier@ordipprof: ~/doc/TP1
golivier@ordipprof:~/doc/TP1$ ocamlopt TP10camlCompile.ml -o TP10camlCompile
golivier@ordipprof:~/doc/TP1$ ./TP10camlCompile Geraldine
Bonjour Geraldine
golivier@ordipprof:~/doc/TP1$ ./TP10camlCompile
Erreur: Pas assez d'arguments en entrée de la fonction
golivier@ordipprof:~/doc/TP1$
```

5. Dupliquez le fichier de code grâce à une commande Linux et ouvrez le fichier dupliqué directement depuis Emacs en utilisant un raccourci clavier.
6. Modifiez le code OCaml pour que l'utilisateur puisse également donner son nom de famille et que le programme l'affiche dans son message de bonjour.