# TP 1 :
# Introduction - Commandes système

Objectif: le but de ce TP est d'aborder la manipulation des commandes système pour la gestion des répertoires et des fichiers.

## Exercice 1 : Quelques commandes Unix

Interpréter le résultat obtenu suite à l'exécution des commandes suivantes:

```
mkdir SYS_TP1
cd SYS_TP1

echo "bonjour"
echo "bonjour"> foo
cat foo
echo "bonsoir"> foo
echo "bonsoir">> foo

cp foo copie_foo
cat foo > copie_foo.txt

cat << fin > du_texte.txt
> je tape du texte autant que je veux
> jusqu'au mot de la fin
> fin

ls
ls -al *txt | sort
(ls -l | grep txt | wc -l)>nb_copie.txt

ln -s foo lien_symbolique_foo
ln foo lien_materiel_foo
cat lien_symbolique_foo
cat lien_materiel_foo
rm foo
cat lien_symbolique_foo
cat lien_materiel_foo

cd ..
rm -r SYS_TP1
```

## Généralités sur les appels système

### Histoire

- Unix a été écrit en langage C $->$ l'accès aux primitives système est facile à partir d'un programme écrit en C.

### Accès à partit du C

- Un appel système est accessible comme une fonction ou procédure.
- Pour obtenir de l'information sur un appel système, utiliser la commande *man* pour accéder à la page du manuel correspondant.
- Ne pas oublier de déclarer l'entête de la fonction avec le type de tous ces paramètres. Pour connaître ces paramètres, consulter la page de manuel correspondant à cette fonction.

### Exemple de petit programme C sous Unix

Soit le source suivant, rangé dans *hello.c*

```
#include <stdio.h>
extern int printf(char *, ...);


int main(void)
{
        printf("Hello world.\n");
}
```

La ligne de commande :

```
gcc -o hello hello.c
```

appelle le compliateur (préprocesseur, génération de code objet, édition de liens)

L'éxecution du programme se fait en tapant la ligne de commande :

```
./hello
```

## Exercice 2 : Mise en majuscule

Sur la base de ce qui a été fait en TD, écrire en C dans le fichier *mettreEnMaj.c* une fonction *mettreEnMajuscule* qui prend une chaîne de caractères en argument et qui rend cette même chaîne avec les minuscules converties en majuscules.

Ensuite, dans le fichier *testMettreEnMaj.c*, écrire un programme qui prend en argument une série de mots et l'affiche sur la sortie standard en convertissant les minuscules en majuscules (en utilisant bien-sûr la fonction *mettreEnMajuscule*).

Écrire un Makefile qui permet d'obtenir à partir de votre programme C l'éxecutable *exeMettreEnMaj*.

## Exercice 3 : Parcours d'une arborescence

Écrire un programme *parcours* qui parcourt un sous-arbre de l'ensemble de fichiers et affiche le nom de tous les répertoires et fichiers rencontrés. La racine des sous-arbres sera donné en argument de la ligne de commande.

Indication: utiliser les fonctions *opendir*, *readdir* et *chdir*.

## Exercice 4 : Parcours modifié

Reprendre l'exercice précédent en n'affichant cette fois que les fichiers ou répertoires dont la date de la dernière modification est antérieure à une date donnée en second paramètre de la ligne de commande.

Indication: utiliser la fonction *stat* ainsi que la fonction *convert* définie dans le fichier *Temps.h*.

Une fois fini, regarder le manuel de la commande *find* et faire les mêmes affichages de sous-arbre soit avec cette commande, soit avec votre programme.