

M2 CCI

UE Programmation et Langages

Cours « Programmation »

Laurent Mounier

Organisation de l'UE « Programmation et Langages »

- Partie **Programmation**

- Cours : L. Mounier *[jeudi 8h-9h30]*
- TPs : C. Ene *[jeudi 9h45 – 11h15]*

- Partie **Langage Machine**

- Cours/TD : G. Kponhinto *[vendredi 13h30 – 16h45]*

Du 20/09 au 20/11 + projet semaine du 23/11

Organisation du cours « Programmation en C »

- Page Moodle du cours :

<https://im2ag-moodle.univ-grenoble-alpes.fr/course/view.php?id=48>

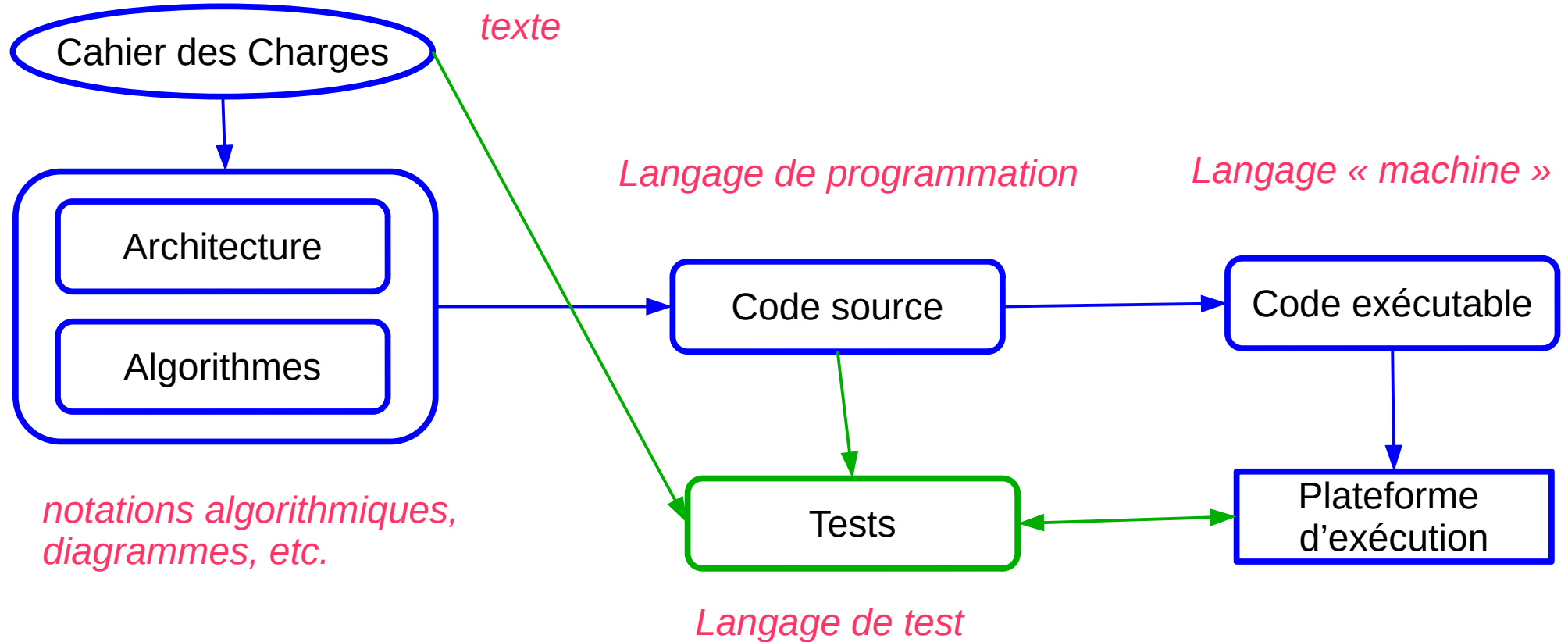
- **Évaluation**

- Examen terminal (2h) [semaine du 30/11]
- Contrôle Continu :
 - Un Devoir Surveillé (1h15) [vendredi 6/11 – à *confirmer*]
 - Une note de projet [semaine du 23/11]

→ les TP ne sont pas évalués ...

Quelques étapes du développement (simplifié) d'un programme

Ajout d'une étape de test ...



Langages de programmation

Grande diversité des « plateforme d'exécution »

ordinateurs « classiques », smartphones, objets connectés, systèmes embarqués, machines virtuelles, « cloud », etc.

Code exécutable

- fortement lié à la plateforme d'exécution
- code **binaire** (base 2, séquence de « 0 » et de « 1 »)
 - doit être produit automatiquement

Algorithme

- niveau abstraction, **expressivité élevée**
- permet de **raisonner** sur le programme (correction, complexité)

Langage de programmation = décrire le programme dans un formalisme :

- lisible, **non ambiguë**, suffisamment expressif
- qui puisse être **traduit automatiquement** vers du code exécutable

Grande variété des langages de programmation

- différents paradigmes
- différents objectifs (systèmes embarqués, applications graphiques, calcul scientifique, machine learning, etc.)
- différents niveaux d'expression
- **compromis** entre : expressivité, efficacité du code produit, sûreté /sécurité, code existant, etc.
- évolution continue ...

Langages de programmation

Grande diversité des « plateforme d'exécution »

ordinateurs « classiques », smartphones, objets connectés/systèmes embarqués, systèmes industriels, etc.

Code exécutable

fortement lié à la plateforme d'exécution

code **binaire** (base 2, séquence de « 0 » et de « 1 ») → doit être produit automatiquement

Algorithme

- niveau abstraction, **expressivité élevée**
- permet de **raisonner** sur le programme (correction, complexité)

Langage de programmation = décrire le programme dans un formalisme :

- lisible, **non ambiguë**, suffisamment expressif
- qui puisse être **traduit automatiquement** vers du code exécutable

Grande variété des langages de programmation

- différents paradigmes
- différents objectifs (systèmes embarqués, applications graphiques, calcul scientifique, machine learning, etc.)
- différents niveaux d'expression

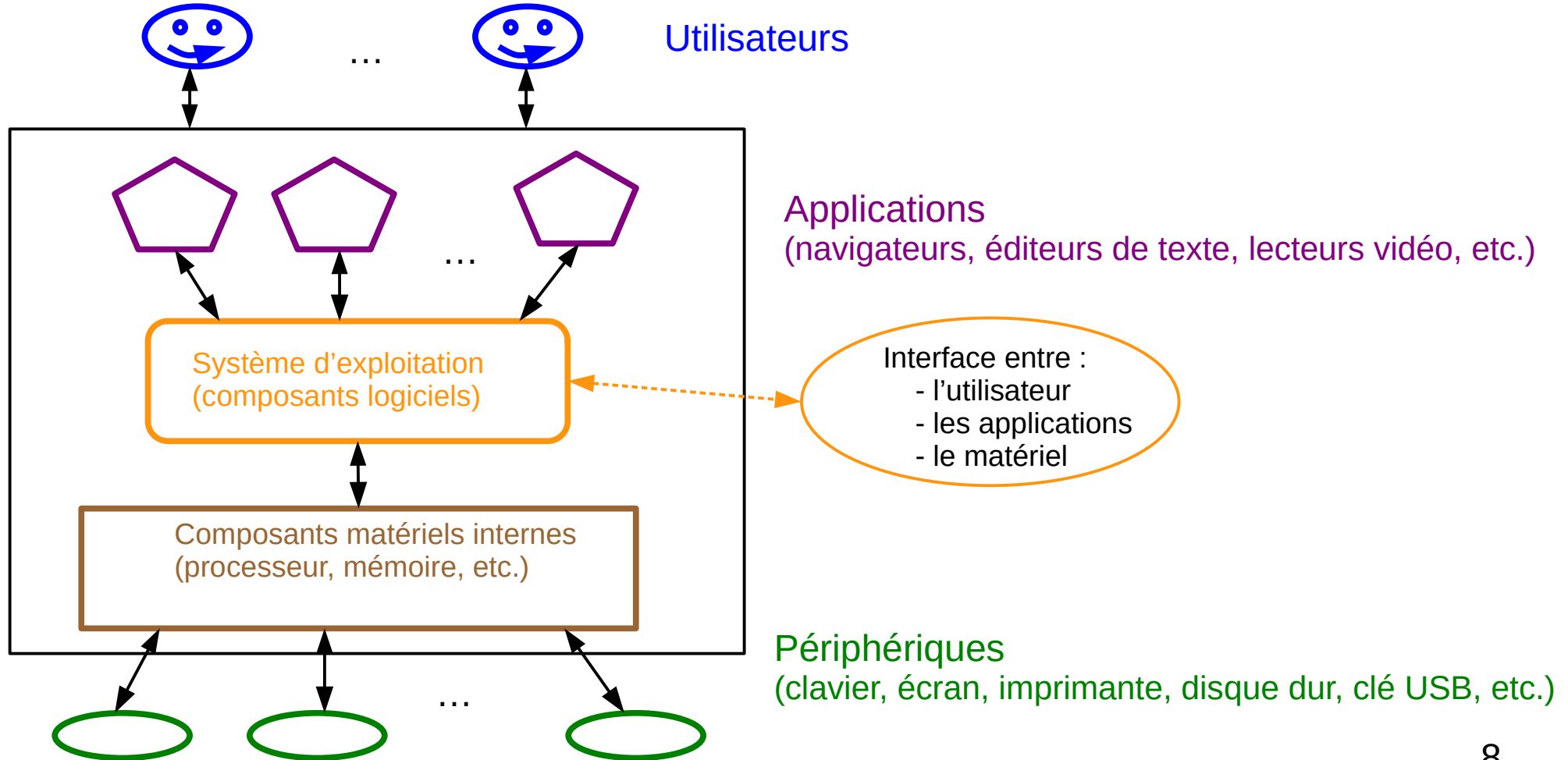
compromis entre : expressivité, efficacité du code produit, sûreté /sécurité, code existant, etc. évolution continue ...

Objectifs de ce cours

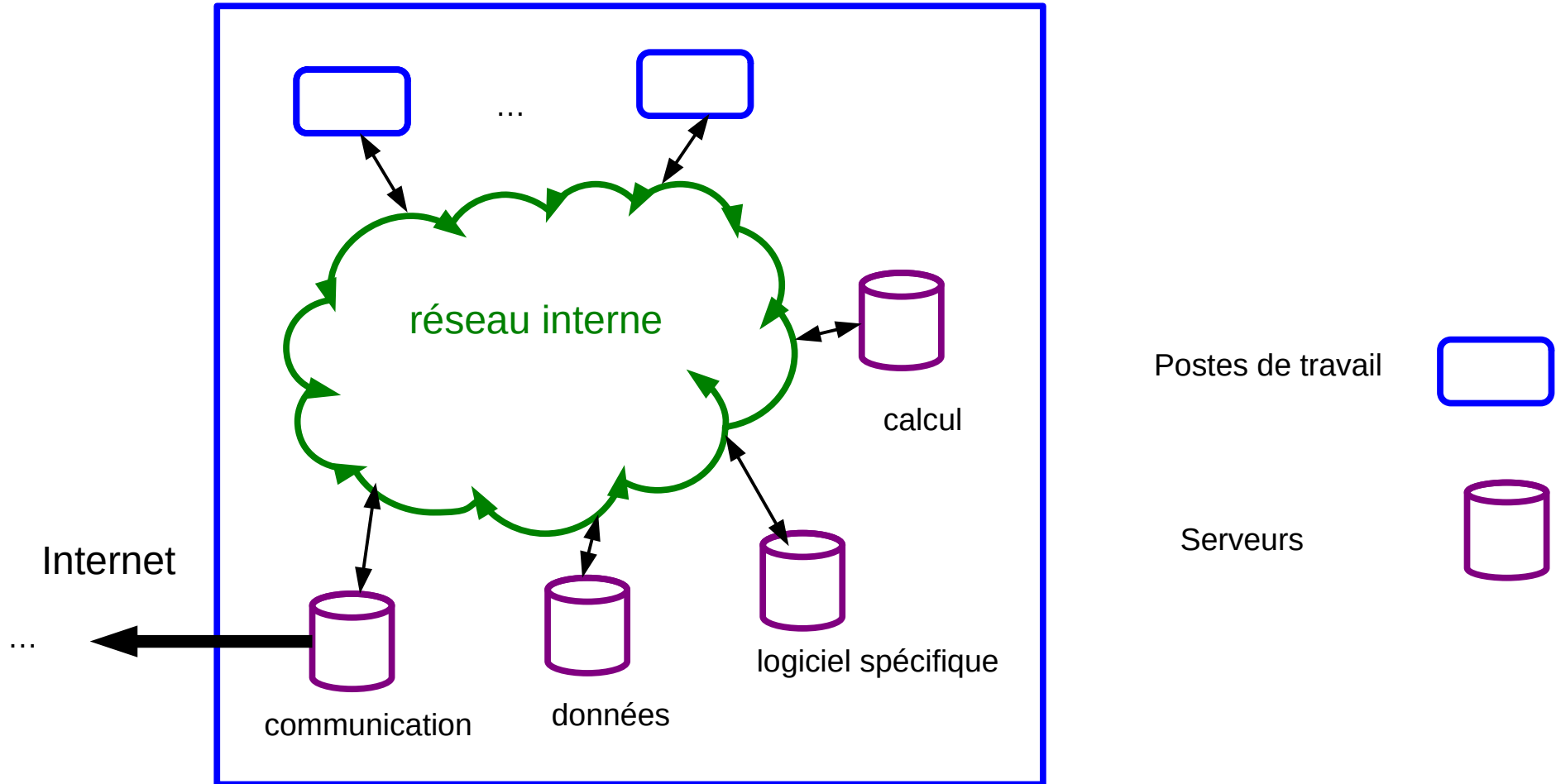
- (Première ?) notion de « langage de programmation »
... à travers l'exemple du langage C ...
- Mise en œuvre des notions vues en cours d'Algo
- Quelques outils et techniques d'aide à la programmation
(mise au point et test de programmes)
- Expérience de « projet de programmation »
développer à plusieurs, code de taille « moyenne »

Liens avec les UEs : algo, génie logiciel, système/réseaux, PLAI

Plateforme d'exécution



Réseau informatique (~ celui du bâtiment F)



Système d'Exploitation (*Operating System*)

→ ensemble de « programmes » (composants logiciels)

Rôle

- Abstraction de la couche matérielle
 - Gestion :
 - Des périphériques
 - Des ressources (matériel, énergie, etc.)
 - Des utilisateurs
 - Des données
 - Etc.
- ne peut être ignoré ni l'utilisateur ... ni par le programmeur

Exemples d'OS :

Windows, Linux, Android, iOS, MacOS, etc.

Interpréteur de commande

- Un des composants (logiciels) de l'OS ...
- Gère les **interactions** avec l'utilisateur
 - Soit via une interface **graphique** (souris, menus, boutons, etc.)
 - Soit via une interface « **textuelle** », via des commandes pré-définies (langage de commande)

Ex : *cmdtool* sous Windows, *terminal de commande* sous Linux/macOS

Algorithme :

1. Lecture d'une suite de caractères terminée par *return*
2. Est-ce une commande « correcte » (= appartient au langage de commande) ?
 - Si oui : l'exécuter ...
 - Si non : afficher un message d'erreur
3. Recommencer en 1.

Remarques :

- sous Linux il existe différents langages de commande (shell-script)
- on peut exécuter plusieurs interpréteurs de commande simultanément

Compte utilisateurs, fichiers, répertoires

L'OS gère les utilisateurs autorisés à accéder à la plateforme d'exécution ...

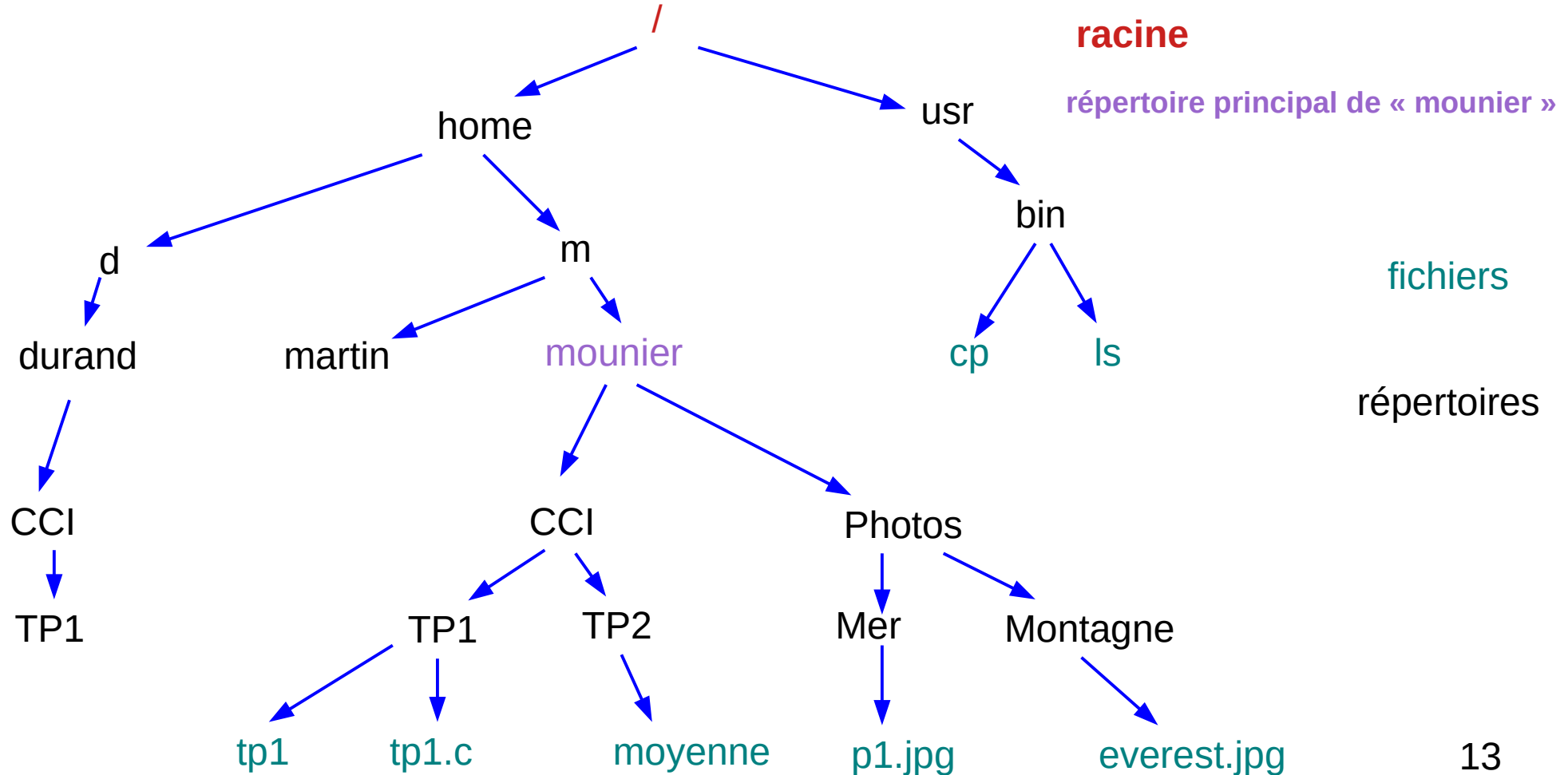
Notion de « compte » utilisateur :

- Un nom d'accès (ou login)
- Un mot de passe (en général ...)
- Un espace mémoire pour stocker des informations
- Diverses informations liées aux compte (« préférences »)

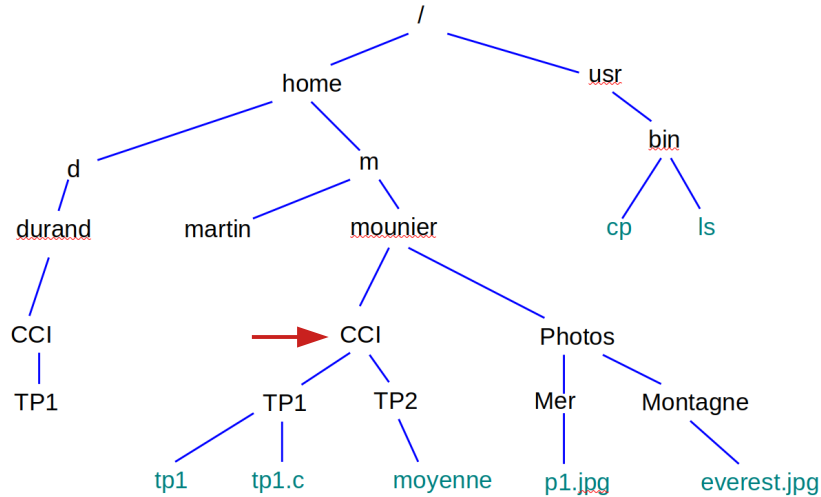
Stockage de l'information en mémoire « permanente » :

- fichier (file) : ensemble « cohérent » d'information
(ex : un texte, une image, un document PDF, une vidéo, un programme exécutable, etc.)
- répertoire (directory) : un ensemble de fichiers ... ou de répertoires
→ une hiérarchie / arborescence de fichiers et répertoires ...

Exemple d'arborescence de fichiers/répertoires



Désignation des fichiers/répertoires sous Linux



Noms absolus : depuis la racine

/home/m/mounier/CCI/TP1/tp1.c

/usr/bin

/home/d/durand/CCI/TP1

Noms relatifs : depuis le répertoire courant (→)

TP2/moyenne

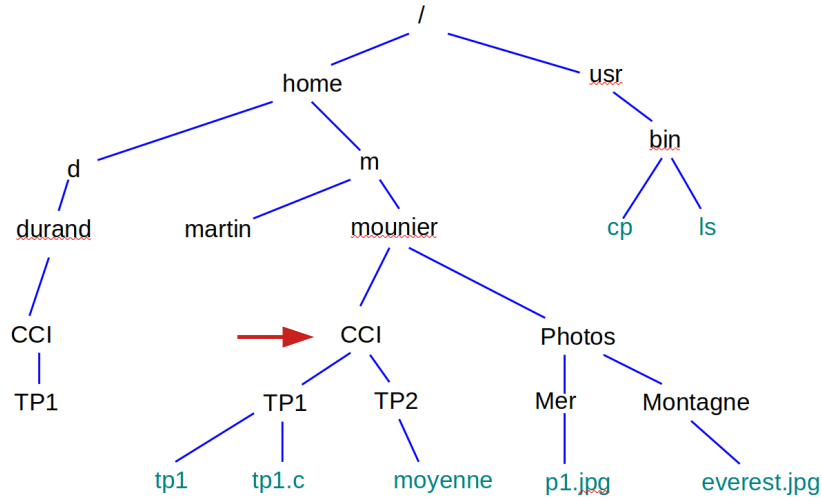
../Photos/Mer/p1.jpg

../CCI/TP1/tp1.c

Quelques notations utiles :

- . répertoire courant
- .. répertoire « père »
- ~ répertoire principal
- * tous les fichiers et répertoires du répertoire courant

Quelques commandes utiles ...



pwd
cd
ls
mkdir
rmdir
cp
rm
mv

print working directory
change directory
list
make directory
remove directory (s'il est vide)
copy (de fichiers, de répertoires)
remove (un fichier)
move (déplacer/renommer un fichier ou répertoire)

Exercice :

1. supprimer le fichier *tp1.c*
2. copier le fichier *p1.jpg* dans *Montagne*
3. effacer l'ensemble du répertoire *durand*

Un mot sur les droits d'accès (Linux)

Attributs associés à un fichier ou répertoire :

Taille, date de création, propriétaire, droits d'accès

Droits associés à un fichier ou répertoire : *read* (r), *write* (w), *execute* (x)

Catégorie d'utilisateur par rapport à un fichier ou répertoire donné :

user (u), *group* (g), *others* (o), *all* (a)

Visualiser les attributs d'un fichier : `ls -l`

---|---|---
user group others

```
im2ag-mandelbrot:~/CCI_PL1/Rentree1/PL1/Rentree1
[08:24:47]mounlaur$ ls -l
total 0
-rwxr-xr-- 1 mounlaur 21003 23 Aug 18 2015 hello.c
-rwxr-xr-- 1 mounlaur 21003 28 Aug 18 2015 ping
-rwxr-xr-- 1 mounlaur 21003 28 Aug 18 2015 pong
```

Modifier les droits d'accès : `chmod`

`chmod u+x fichier1`

ajout des droits d'exécution au propriétaire sur le fichier

`chmod g-wx fichier2`

enlève les droits ecriture/exécution au groupe sur le fichier

`chmod a+r fichier3`

ajout des droits de lecture à tous sur le fichier

`chmod -R a+r repertoire1`

ajout des droits de lecture à tous sur l'ensemble du répertoire