Télécharger l'annexe *annexe-exercice-arbre.zip* et extraire les fichiers.

Exercice 1 :

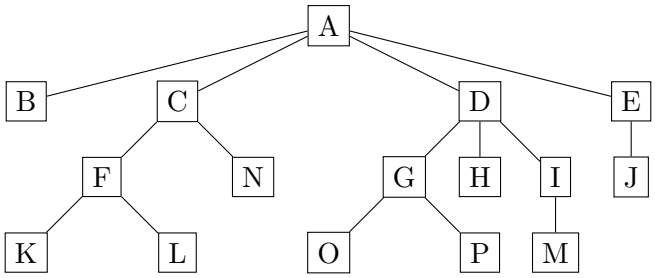
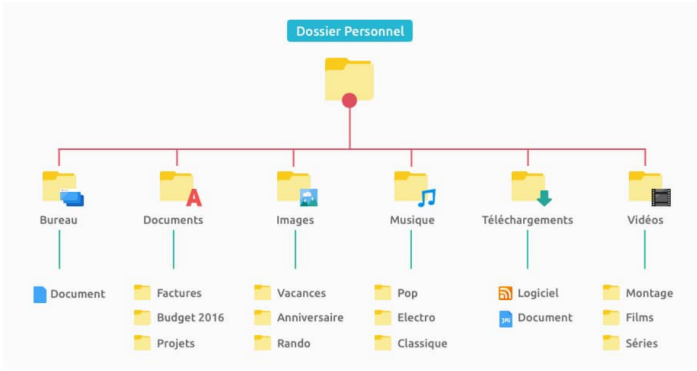


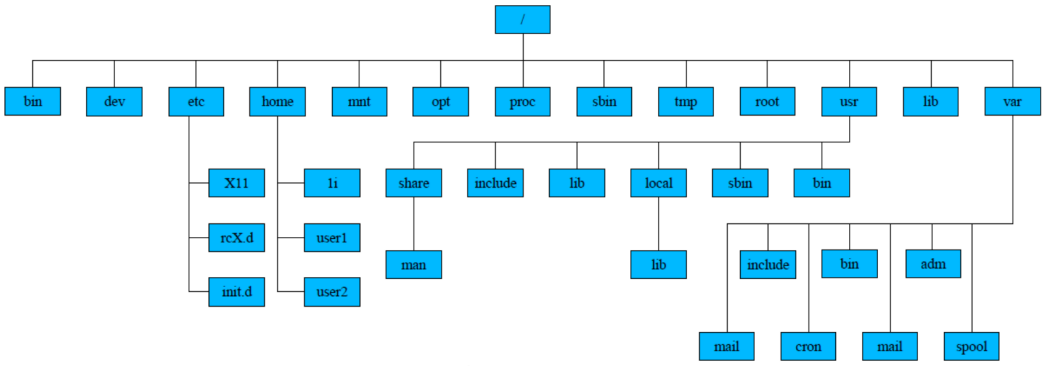
FIGURE 1 – Une structure arborescente

1. Donner le nom de la racine, le nombre de feuilles, la taille et la profondeur de l'arbre figure 1. On considère qu'un arbre qui n'a qu'une racine est de profondeur 0.
2. Donner un parcours en largeur puis un parcours en profondeur de cette structure arborescente.

Exercice 2 : Voici deux arborescences de fichiers. Pour chacune d'elles, donner la taille et la hauteur de l'arbre correspondant.



Arborescence Windows



Arborescence Linux

FIGURE 2 – Systèmes d'exploitation

Exercice 3 : On souhaite réaliser un correcteur orthographique. Pour cela, on a besoin d'un dictionnaire contenant les différents mots de la langue française. Pour l'instant, on dispose seulement de quelques mots : *arbre, arbitre, arbitrer, binaire, binette, bio, empiler, exact.*
On souhaite stocker ces mots par ordre alphabétique dans une liste Python. Chaque nom/adjectif qualificatif sera présent au singulier et au pluriel, les adjectifs qualificatifs seront également présents au féminin et les verbes seront également conjugués au présent de l'indicatif.

Christophe Viroulaud

1. Quelle sera la taille de cette liste ? En déduire le nombre de tests nécessaires (mot à mot et pas lettre à lettre) pour trouver le mot « exactes » en effectuant une recherche linéaire.
2. Représenter ce dictionnaire rudimentaire sous la forme d'un arbre dans lequel chaque nœud est une lettre. La racine et la fin des mots seront notées avec le symbole *. Combien de tests sont nécessaires pour trouver le mot *« exactes »* ?

Exercice 4 : Un site web de cuisine stocke chaque recette sous forme d'un fichier *json (JavaScript Object Notation)*.

1. Ouvrir le fichier *recette-fondant.json* à l'aide d'un éditeur de texte et construire sur papier l'arbre correspondant à la recette. Il faut remarquer qu'il n'y a pas de nœud racine.
2. Ouvrir le fichier *recette.py*. L'arbre a été construit avec la classe *Noeud*.
3. Écrire la fonction **BFS(arbre : Noeud)** $\rightarrow$ **list** qui effectue un parcours en largeur de l'arbre et renvoie la liste des valeurs (attribut *valeur)* des nœuds visités.
4. Écrire la fonction **affiche(arbre : Noeud, decalage : str = "", affichage : str = "")** $\rightarrow$ **str** qui affiche l'arbre avec un décalage de quatre espaces entre chaque niveau (code 1).

```
None
    recette
        Fondant au chocolat
    difficulté
        facile
    temps
        préparation
            unite
                min
            valeur
                15
```

Code 1 – Début d'affichage de l'arbre

Christophe Viroulaud