TD2 : Encore des pointeurs...

CSH : Initiation au C et au shell

★ **Exercice 1:** Un programme C contient la déclaration suivante :

```
char *couleur[6] = {"rouge", "vert", "bleu", "blanc", "noir", "jaune"};

char **pcoul = couleur;
```

▷ **Question 1:** Que représente `couleur` ? Et `pcoul` ?

▷ **Question 2:** Que désigne `couleur + 2` ?

▷ **Question 3:** Quelle est la valeur de `*couleur` ?

▷ **Question 4:** Quelle est la valeur de `*couleur + 2` ?

▷ **Question 5:** Quelle est la valeur de `*(*(couleur + 5) +3)` ? (Même question avec +4, +5, +6).

★ **Exercice 2: Pointeurs et structures.**

Soit le type suivant :

```
typedef struct {
  char nom[MAX_CAR];
  int  rayon;
} planete_t;
```

▷ **Question 1:** Écrire un constructeur pour ce type `planete_t`.

▷ **Question 2:** Écrire une fonction qui saisit les données relatives à une planète, et crée une planète en conséquence.

▷ **Question 3:** Écrire une fonction qui duplique une planète (un *copy constructor*).

▷ **Question 4:** Soit l'extrait suivant permettant de tester les fonctions précédentes ;

```
void main() {

  planete_t p;
  planete_t *ptr_p;
  ptr_p = planete_saisir();
  p = *ptr_p;
  printf("%s %d \n", p.nom, p.rayon);
  ptr_p = planete_dup(p);
  printf("%s %d \n", ptr_p->nom, ptr_p->rayon);
}
```

Comment s'analysent les types des différentes références suivantes :

Référence	Type
`ptr_p`	
`*ptr_p`	
`ptr_p->nom`	
`ptr_p->rayon`	
`p`	
`&p`	
`p.nom`	
`p.rayon`	

★ **Exercice 3: Chaînes de caractères.**

▷ **Question 1:** Écrire une fonction `PremierCar(...)` qui renvoie l'adresse de la première occurence du caractère `c` dans une chaîne de caractères dont l'adresse (du premier caractère) est passé à l'argument `ptrCar` :

```
char *PremierCar(char c, char *ptrCar)
```

▷ **Question 2:** Écrire la fonction `int main(int argc, char *argv[ ])` permettant de tester la fonction. Voici un exemple d'exécution :

```
$ ./occurence o Bonjour
La premiere occurence de 'o' dans 'Bonjour' est en position 1
$ ./occurence z Bonjour
'z' n'est pas dans Bonjour.
```

★ **Exercice 4: Filiation.**

Sous Unix, il existe un ensemble de pointeurs représentant la filiation des processus. Dans la figure 1, `fils A` est le premier processus fils créé par le processus père, `fils B` est le deuxième fils créé par le processus père ...etc. Chaque fils peut évidemment créer des fils, et le père est également le fils d'un autre processus. Chaque processus est caractérisé par un numéro (un entier positif).

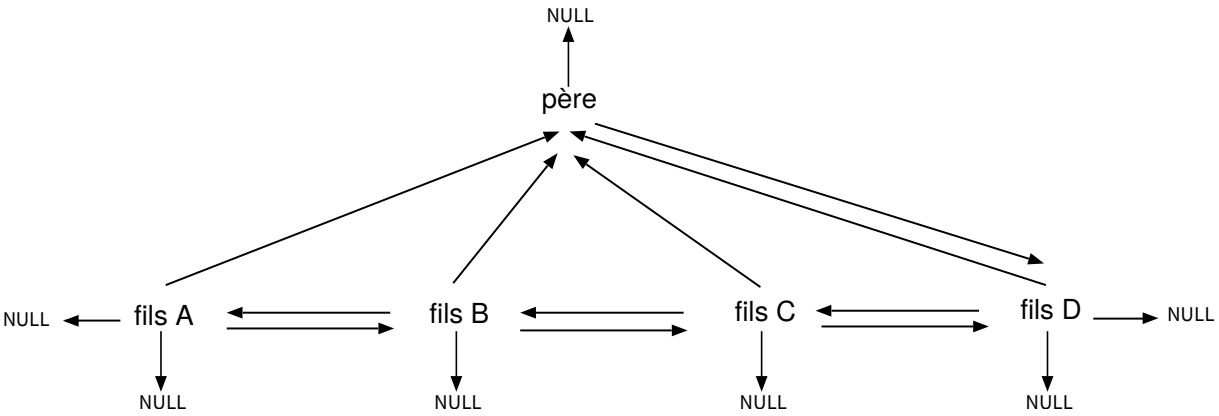

FIGURE 1 – *Filiation sous Unix*

▷ **Question 1:** Représenter schématiquement, sous la forme d'une structure de données, un nœud (c'est à dire un processus) de cette filiation, puis écrire la définition de cette structure en C.

▷ **Question 2:** Le processus repéré par la variable pointeur `p` vient de créer un processus fils de numéro `no`. Écrire en C la fonction `maj` qui met à jour la filiation des processus.