Episode 1
Bases shell Unix et C

Programmation système unix

BOOTSTRAP

Cet épisode est une introduction au fonctionnement du shell Unix et de la programmation élémentaire en C.

Nous commencerons avec des commandes Unix essentielles, leur imbrication en chaîne afin de créer des commandes puissantes. Nous en profiterons pour créer nos premiers scripts shell.

Enfin, nous poserons de bonnes bases pour se familiariser avec le C et enchaîner sur des concepts plus complexes dans les jours à venir.

Dennis et Ken ont créé le C il y a plus de 50 ans. Pour autant, il reste aujourd'hui le langage le plus executé (attention, on a pas dit le plus populaire…) en 2023.
Il est orienté minimisation de la mémoire nécessaire et maximisation de la performance.
Il s'exécute partout, sur toutes les architectures.
Sa structure est simple et extrêmement claire : cela le rend un excellent langage pour s'approprier les bons réflexes de développeur. Son aspect bas-niveau, au plus prêt de la mémoire, permet de comprendre exactement ce que l'on fait, pourquoi on le fait et comment bien le faire.

Evidemment, il est plein de défauts : https://fr.wikipedia.org/wiki/C_(langage) / Qualités et défauts.

Un développeur qui maîtrise le C peut appréhender n'importe quel langage.
Un développeur qui maîtrise le C comprend mieux que les autres "ce qu'il se passe à l'intérieur".

Mangez du C.

PREAMBULE

Le livre Wuzazu contient la première mention connue du jeu "Papier-Caillou-Ciseau". Il a été écrit par l'écrivain de la dynastie Ming Xie Zhaozhi, qui indique alors que ce jeu date de la dynastie Han (206 BC - 220 AD). Dans le livre, le jeu s'appellait shoushiling. Le livre Note of Liuyanzhai mentionne également le jeu, l'appelant shoushiling, huozhitou, or huoquan.

Au travers de l'histoire japonaise, nous retrouverons des références fréquentes au jeux "sansukumi-ken", "ken" signifiant jeu de poing, avec une impasse "sukumi" à trois voies "san". C'est à lire dans le sens A bat B, B bat C et C bat A. Ce jeu est originaire de Chine avant d'être importé au Japon et de devenir populaire.

Au début du 20ème siècle, papier caillou ciseaux s'est répandu au delà de l'Asie, notamment grâce au contact augmenté entre le Jpaon et l'Ouest. Son nom anglais est alors pris par traduction du nom japonais des gestes utilisés. Dans le reste de l'Asie, le papier est remplacé par tissu. La forme des ciseaux est également adoptée du style japonais.

En 1927, "La vie au patronage", un magazine pour en enfants en France, le décrivait en détail, et le considérant comme un "jeu japonais" Son nom français alternatif ("chi-fou-mi"), est basé sur les anciens mots japonais pour "un, deux, trois" ("hi, fu, mi").

Un article du New York Times de 1932 explique les règles pour les lecteurs américains, indiquant que le jeu n'était alors pas très répandu. L'édition de 1933 du magazine "Compton's Pictured Encyclopedia" le décrivait comme un moyen commun de résolution des conflits entre enfants lors de son article sur le Japon : "This is such a good way of deciding an argument that American boys and girls might like to practice it too."

EXERCICES SHELL UNIX
(c'est pas encore le C)

EXERCICE 1 ⭐

Créer un fichier **z** dans le répertoire **episode1/ex1,** qui affiche le caractère **'Z'** suivi d'un retour à la ligne **'\n'** quand le binaire **cat** est utilisé pour le lire.

Terminal

```
~/> cat episode1/ex1/z
Z
~/> cat -e episode1/ex1/z
Z$
```

Tips :

man touch, man mkdir, man echo
Google "echo to a file"

EXERCICE 2 ⭐

Trouver une commande qui permet d'afficher la liste des fichiers du répertoire /**etc**/ avec les features suivantes :

- N'affiche pas les fichiers cachés
- Tri par ordre de dernière modification
- Affiche de la plus ancienne à la plus récente modification

Terminal

```
~/> YOUR_COMMAND /etc/
total 1476
-rw-r--r--  1 root      root        887 avril  1  2013 rpc
-rw-r--r--  1 root      root       2932 avril  1  2013 protocols
-rw-r--r--  1 root      root        604 sept. 16  2018 deluser.conf
-rw-r--r--  1 root      root       1816 déc.  27  2019 ethertypes
```

```
drwxr-xr-x  2 root      root        4096 févr. 24  2020 usb_modeswitch.d
[...]
-rw-r--r--  1 root      root        9454 déc.  17 12:54 locale.gen
drwxr-xr-x  2 root      root        4096 déc.  17 13:01 vim
drwxr-xr-x  2 root      root        4096 déc.  17 13:03 ld.so.conf.d
-rw-r--r--  1 root      root       89339 déc.  17 13:03 ld.so.cache
-rw-r--r--  1 root      root       25385 déc.  17 13:03 mailcap
```

EXERCICE 3 ⭐⭐

Ecrire un script, nommé **count_files.sh** qui affiche le nombre de fichier (“regular files”) et le nombre de dossier dans le répertoire courant (pas ses sous-répertoires).

Terminal

```
~/> ./count_files.sh
Regular files: 74
Directory: 9
```

Tips :

man find, man wc. Il faut chaîner ces commandes.
Pensez aux fichiers cachés (commençant par un .)
Essayer dans un répertoire avec suffisamment de fichier / dossier pour tester. Votre home par exemple (~/ ou /home/VOTRE_UTILISATEUR/)
Rendre votre script executable à l’aide de `chmod +x count_files.sh`

EXERCICE 4 ⭐⭐

Trouver une commande qui permet d’afficher les adresses IP des cartes réseaux de votre machine. Seules les adresses IPv4 sont affichées, par les adresses IPv6. Chaque adresse

sera suivie d’un retour à la ligne.

Terminal

```
~/> YOUR_COMMAND
127.0.0.1
192.168.1.10
192.168.122.1
172.17.0.1
172.19.0.1
172.18.0.1
```

Tips :

man ifconfig ou man ip

EXERCICES C

EXERCICE 5 ⭐

Fichier : my_print_alpha.c

Écrire une fonction **my_print_alpha** qui affiche l'alphabet en minuscule dans l'ordre ascendant sur une seule ligne.

Le prototype de votre fonction sera :

```
int     my_print_alpha(void);
```

La fonction main utilisée sera :

```
int     main(void)
{
    my_print_alpha();
}
```

Terminal

```
~/> gcc my_print_alpha.c
~/> ./a.out
abcdefghijklmnopqrstuvwxyz
~/>
```

Tips :

While plutôt que For (pas besoin de For en C).
printf() interdit, utilisez putchar()
man 7 ascii

EXERCICE 6 ⭐

Fichier : my_print_revalpha.c

Écrire une fonction **my_print_revalpha** qui affiche l'alphabet en majuscule dans l'ordre descendant sur une seule ligne.

Le prototype de votre fonction sera :

```
int     my_print_revalpha(void);
```

La fonction main utilisée sera :

```
int     main(void)
{
    my_print_revalpha();
}
```

Terminal

```
~/> gcc my_print_revalpha.c
~/> ./a.out
ZYXWVUTSRQPONMLKJIHGFEDCBA
~/>
```

EXERCICE 7 ⭐

Fichier : my_print_digit.c

Écrire une fonction **my_print_digit** qui affiche les chiffres de 0 à 9 sur une seule ligne.

Le prototype de votre fonction sera :

```
int     my_print_digit(void);
```

La fonction main utilisée sera :

```
int     main(void)
{
    my_print_digit();
}
```

Terminal

```
~/> gcc my_print_digit.c
~/> ./a.out
0123456789
~/>
```

EXERCICE 8 ⭐

Fichier : my_isneg.c

Écrire une fonction **my_isneg** qui affiche N si l'entier passé en argument est négatif et P s'il est positif ou nul.

Le prototype de votre fonction sera :

```
int     my_isneg(int n);
```

La fonction main utilisée sera :

```
int     main(void)
{
    int pos;
    int neg;

    pos = 42;
    neg = -84;
    my_isneg(pos);
    my_isneg(21);
    my_isneg(neg);
}
```

Terminal

```
~/> gcc my_isneg.c
~/> ./a.out
P
P
```

```
N
~/>
```

EXERCICE 9 ⭐

Fichier : my_swap.c

Écrire une fonction **my_swap** qui échange le contenu de 2 entiers (int), dont les adresses sont données en paramètres.

Le prototype de votre fonction sera :

```
int     my_swap(int *a, int *b);
```

La fonction main utilisée sera :

```
int     main(void)
{
    int my_int1;
    int my_int2;

    my_int1 = 42;
    my_int2 = 84;
    printf("Value of int1 = %i", my_int1);
    printf("Value of int2 = %i", my_int2);
    my_swap(&my_int1, &my_int2);
    printf("Value of int1 = %i", my_int1);
    printf("Value of int2 = %i", my_int2);
}
```

Terminal

```
~/> gcc my_swap.c
~/> ./a.out
Value of int1 = 42
Value of int2 = 84
Value of int1 = 84
Value of int2 = 42
~/>
```

Tips :

Venez poser une question sur les pointeurs. Qu'est-ce qu'un pointeur ? Qu'est-ce que "l'adresse de x" ?

EXERCICE 10 ⭐

Fichier : my_putstr.c

Écrire une fonction **my_putstr** qui affiche, un par un, les caractères d'une chaîne de caractères (char *).
L'adresse de la chaîne de caractères sera trouvée dans le pointeur passé en paramètres de la fonction.

Le prototype de votre fonction sera :

```
int     my_putstr(char *str);
```

La fonction main utilisée sera :

```
int     main(void)
{
    char str[] = "I like ponies";

    my_putstr(str);
}
```

Tips :

Par quoi se termine une chaîne de caractères ?

EXERCICE 11 ⭐

Fichier : my_strlen.c

Écrire une fonction **my_strlen** qui retourne le nombre de caractères d'une chaîne de

caractères.

Le prototype de votre fonction sera :

```
int     my_strlen(char *str);
```

La fonction main utilisée sera :

```
int      main(void)
{
    char str[] = “I like ponies”;
    int  size;

    size = my_strlen(str);
    printf(“%d”, size);
}
```

EXERCICE 12 ⭐⭐

Fichier : my_sort_int_array.c

Écrire une fonction **my_sort_int_array** qui tri un tableau d’entier dans l’ordre ascendant à partir d’un pointeur sur le premier élément du tableau et de la taille du tableau.

Le prototype de votre fonction sera :

```
void     my_sort_int_array(int *tab, int size);
```

La fonction main utilisée sera :

```
int      main(void)
{
    int  tab[5] = {64, 58, -52, 42, 5};

    my_sort_int_array(tab, 5);
    printf(“%d”, tab[0]);
    printf(“%d”, tab[4]);
}
```

EXERCICE 13 ⭐

Fichier : my_strcpy.c

Écrire une fonction **my_strcpy** qui copie une string dans une autre. La string de destination dispose déjà d'assez de mémoire (vous ne gérer pas l'allocation). La fonction retourne la string de destination.

Le prototype de votre fonction sera :

```
char	*my_strcpy(char* dest, char const *src);
```

La fonction main utilisée sera :

```
int	main(void)
{
    char  src[256] = "Hello hello";
    char  dest[256];

    my_strcpy(dest, src);
    my_putstr(dest);
}
```

EXERCICE 14 ⭐⭐

Fichier : my_strcapitalize.c

Écrire une fonction **my_strcapitalize** qui transforme en majuscule la première lettre de chaque mot. La fonction renvoie la string.
La string **hey, how are you? 42WORds forty-two; fifty+one**
va devenir **Hey, How Are You? 42words Forty-Two; Fifty+One**

Le prototype de votre fonction sera :

```
char	*my_strcapitalize(char* str);
```

La fonction main utilisée sera :

```
int	main(void)
{
    char  *str[] = “hey, how are you? 42WORds forty-two; fifty+one”;

    my_strcpy(str);
    my_putstr(str);
}
```

EXERCICE 15 ⭐⭐

Fichier : my_print_params.c

Écrire un programme qui affiche les arguments passés. La fonction main revoie 42.

Terminal

```
~/> ./my_print_params Hello guys
Hello
guys
~/> ./my_print_params Ma chanson préférée est Les kéké-boys Train pour Bandol
Ma
chanson
préférée
est
Les
kéké-boys
Train
pour
Bandol
~/> ./my_print_params “Ma chanson préférée” est “Les kéké-boys Train pour Bandol”
Ma chanson préférée
est
Les kéké-boys Train pour Bandol
~/> ./my_print_params
~/> echo $?
42
```

```
~/>
```

Tips :

```
int     main(int argc, char* argv[])
```

man 3 exit

EXERCICE 16 ⭐⭐

Fichier : my_convert_bin.c

Écrire un programme qui convertit en binaire la chaîne de caractères passée en paramètre. La valeur binaire est affichée sur 8 bits (1 octet).

Terminal

```
~/> ./my_convert_bin Pastis
01010000
01100001
01110011
01110100
01101001
01110011
```

EXERCICE 17 ⭐⭐⭐

Fichier : my_read_and_stack.c

Écrire un programme lit sur l'entrée standard une chaîne de caractères (maximum 31 caractères) et qui ajoute cette chaîne à sa pile de mots. A chaque nouvelle lecture, la pile de mots est affichée.

Terminal (en rose, les mots tapés par l'utilisateur)

```
~/> ./my_read_and_stack
Add new word:
HELLO

My stack is:
HELLO

Add new word:
how

My stack is:
HELLO
how

Add new word:
are you?

My stack is:
HELLO
how
are you?
```

Tips :

La difficulté de cet exercice consiste à gérer la taille du tableau de chaîne de caractères : man malloc.

Quelques étapes :

- Ecrire une fonction qui lit une phrase sur l'entrée standard : man 3 scanf et scanf("%31s", string);
- Ecrire une fonction qui affiche un tableau de chaine de caractère
- Ecrire une fonction qui ajoute une string à un tableau de chaine de caractère : man malloc, man strcpy ou man strncpy