

La ligne de commande

Lorsque l'on lance une application il est possible de lui envoyer des informations :

```
myapp.exe myfile.txt
```

Comment lire ces informations ?

Utilisation de argc et argv

Comme déjà dit, la fonction main est obligatoire et doit respecter une signature imposée par la norme C++. Pour le moment, on utilise :

```
int main() {  
}
```

Correspond à une fonction qui ne reçoit aucun paramètres et retourne un entier. Souvent, on ne mets pas l'entier (0 par défaut) ou on retourne un code d'erreur :

```
int main() {  
    return 123456;  
}
```

Il existe une autre signature, avec paramètres :

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

argc contient le nombre de paramètres passé lors de l'appel de la fonction. Chaque paramètre est séparé par un espace et le premier paramètre est le nom du programme. Par exemple :

```
main.cpp
```

```
#include <iostream>

int main(int argc, char* argv[]) {
    std::cout << argc << std::endl;
}
```

Affiche en fonction de la ligne de commande :

```
./a.out
1

./a.out int.txt
2

./a.out int.txt out.txt
3
```

pas encore vu les tableaux (mais vu les string et accès aux caractères ?)

argv permet d'accéder aux chaînes de caractères correspondant à chaque paramètre. On reconnaît dans la déclaration le type `char*`, qui correspond aux littérales chaînes de caractères. Les crochets droits `[]` permettent de déclarer un tableau. Le type de `argv` peut donc se lire comme étant un tableau de chaînes de caractères.

Pour accéder aux éléments d'un tableau, on utilise également les crochets droits, en indiquant la position dans le tableau (en commençant à 0).

main.cpp

```
#include <iostream>

int main(int argc, char* argv[]) {
    std::cout << argc << std::endl;
    std::cout << argv[0] << std::endl;
    std::cout << argv[1] << std::endl;
    std::cout << argv[2] << std::endl;
}
```

affiche si on l'appelle avec `./a.out in.txt out.txt` :

```
3
./a.out
in.txt
out.txt
```

Si on essaie d'afficher une valeur en dehors du tableau, comportement indéterminé.

L'utilisation des tableaux sera détaillé par la suite, en particulier comment les parcourir automatiquement avec une boucle.

Chapitre précédent	Sommaire principal	Chapitre suivant
---------------------------	---	-------------------------

[Cours, C++](#)