> **Ce cours est une mise à jour du cours C++ de OpenClassRoom pour le mettre à jour pour le C++11/14. Le cours d'origine est consultable sur la page suivante : Programmez avec le langage C++, par Mathieu Nebra et Matthieu Schaller. Ce cours est sous licence CC-BY-NC-SA.**

Retourner au sommaire principal

Lire et modifier des fichiers

Pour l'instant, les programmes que nous avons écrits étaient encore relativement simples. C'est normal, vous débutez. Mais avec un peu d'entraînement, vous seriez capables de créer de vraies applications. Vous commencez à connaître les bases du C++ mais il vous manque quand même un élément essentiel : l'interaction avec des fichiers.

Jusqu'à maintenant, vous avez appris à écrire dans la console et à récupérer ce que l'utilisateur avait saisi. Vous serez certainement d'accord avec moi, ce n'est pas suffisant. Pensez à des logiciels comme le bloc-note, votre IDE ou encore un tableur : ce sont tous des programmes qui savent lire des fichiers et écrire dedans. Et même dans le monde des jeux vidéo, on a besoin de cela : il y a bien sûr les fichiers de sauvegardes, mais aussi les images d'un jeu, les cinématiques, les musiques, etc. En somme, un programme qui ne sait pas interagir avec des fichiers risque d'être très limité.

Voyons donc comment faire ! Vous verrez : si vous maîtrisez l'utilisation de cin et de cout, alors vous savez déjà presque tout.

Écrire dans un fichier

La première chose à faire quand on veut manipuler des fichiers, c'est de les ouvrir. Eh bien en C++, c'est la même chose. Une fois le fichier ouvert, tout se passe comme pour cout et cin. Nous allons, par exemple, retrouver les chevrons « et ». Faites-moi confiance, vous allez rapidement vous y retrouver.

On parle de flux pour désigner les moyens de communication d'un programme avec l'extérieur. Dans ce chapitre, nous allons donc parler des flux vers les fichiers. Mais dites simplement « lire et modifier des fichiers » quand vous n'êtes pas dans une soirée de programmeurs. ;)

L'en-tête fstream

Comme d'habitude en C++, quand on a besoin d'une fonctionnalité, il faut commencer par inclure le bon fichier d'en-tête. Pour les fichiers, il faut spécifier #include <fstream> en-haut de notre code source.

> Vous connaissez déjà iostream qui contient les outils nécessaires aux entrées/sorties vers la console. iostream signifie en réalité input/output stream, ce qui veut dire « flux d'entrées/sorties » en français. fstream correspond à file stream, « flux vers les fichiers » en bon français.

La principale différence est qu'il faut un flux par fichier. Voyons comment créer un flux sortant, c'est-à-dire un flux permettant d'écrire dans un fichier.

Ouvrir un fichier en écriture

Les flux sont en réalité des objets. Souvenez-vous que le C++ est un langage orienté objet. Voici donc un de ces fameux objets. N'ayez pas peur, il y aura plusieurs chapitres pour en parler. Pour l'instant, voyez cela comme de grosses variables améliorées. Ces objets contiennent beaucoup d'informations sur les fichiers ouverts et proposent des fonctionnalités comme fermer le fichier, retourner au début et bien d'autres encore.

L'important pour nous est que l'on déclare un flux exactement de la même manière qu'une variable, une variable dont le type serait ofstream et dont la valeur serait le chemin d'accès du fichier à lire.

Comme pour les variables, il y a quelques règles à suivre pour le choix du nom du flux:

- les noms des flux sont constitués de lettres, de chiffres et du tiret-bas _ uniquement ;
- le premier caractère doit être une lettre (majuscule ou minuscule) ;
- on ne peut pas utiliser d'accents ;
- on ne peut pas utiliser d'espaces dans le nom.

Vous l'aurez remarqué, ce sont exactement les mêmes règles que pour les variables. Je ne vous ferai donc pas l'offense de répéter les règles que nous utilisons dans ce cours et qui sont très souvent adoptées par les programmeurs. Tout a déjà été dit au chapitre 4. Dans la suite de ce chapitre nous utiliserons le nom monFlux comme nom pour les exemples de flux. Il satisfait tous les critères; j'espère que vous en conviendrez.

main.cpp

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
   ofstream monFlux("C:/Nanoc/scores.txt");
   //Déclaration d'un flux permettant d'écrire dans le
fichier
   // C:/Nanoc/scores.txt
   return 0;
}
```

J'ai indiqué entre guillemets le chemin d'accès au fichier. Ce chemin doit prendre l'une ou l'autre des deux formes suivantes :

- Un chemin absolu, c'est-à-dire montrant l'emplacement du fichier depuis la racine du disque. Par exemple : C:/Nanoc/C++/Fichiers/scores.txt.
- Un chemin relatif, c'est-à-dire montrant l'emplacement du fichier depuis l'endroit où se situe le programme sur le disque. Par

exemple : Fichiers/scores.txt si mon programme se situe dans le dossier C:/Nanoc/C++/.

À partir de là, on peut utiliser le flux pour écrire dans le fichier.

> Si le fichier n'existait pas, le programme le créerait automatiquement ! Par contre, il faut que le dossier existe. Dans l'exemple précédent, le dossier C:/Nanoc/C++/Fichiers doit exister. Si ce n'est pas le cas, rien ne sera écrit.

Le plus souvent, le nom du fichier est contenu dans une chaîne de caractères string. Dans ce cas, il faut utiliser la fonction c_str() lors de l'ouverture du fichier.

main.cpp

```
string const nomFichier("C:/Nanoc/scores.txt");

ofstream monFlux(nomFichier.c_str());
//Déclaration d'un flux permettant d'écrire dans un fichier.
```

Des problèmes peuvent survenir lors de l'ouverture d'un fichier, si le fichier ne vous appartient pas ou si le disque dur est plein par exemple. C'est pour cela qu'il faut toujours tester si tout s'est bien passé. On utilise pour cela la syntaxe if(monFlux). Si ce test n'est pas vrai, alors c'est qu'il y a eu un problème et que l'on ne peut pas utiliser le fichier.

```
ofstream monFlux("C:/Nanoc/scores.txt");  //On essaye
d'ouvrir le fichier

if(monFlux)  //On teste si tout est OK
{
    //Tout est OK, on peut utiliser le fichier
}
else
{
    cout << "ERREUR: Impossible d'ouvrir le fichier." <<
endl;
}
```

Tout est donc prêt pour l'écriture. Et vous allez voir que ce n'est pas

vraiment nouveau.

Écrire dans un flux

Je vous avais dit que tout était comme pour cout. C'est donc sans surprise que je vous présente le moyen d'envoyer des informations dans un flux : ce sont les chevrons («) qu'il faut utiliser.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    string const nomFichier("C:/Nanoc/scores.txt");
    ofstream monFlux(nomFichier.c_str());

    if(monFlux)
    {
        monFlux << "Bonjour, je suis une phrase écrite dans
un fichier." << endl;
        monFlux << 42.1337 << endl;
        int age(23);
        monFlux << "J'ai " << age << " ans." << endl;
    }
    else
    {
        cout << "ERREUR: Impossible d'ouvrir le fichier." <<
endl;
    }
    return 0;
}
```

Si j'exécute ce programme, je retrouve ensuite sur mon disque un fichier scores.txt dont le contenu est présenté en figure suivante.

Le fichier une fois qu'il a été écrit

Essayez par vous-mêmes ! Vous pouvez par exemple écrire un programme qui demande à l'utilisateur son nom et son âge et qui écrit ces données dans un fichier.

Les différents modes d'ouverture

Il ne nous reste plus qu'un petit point à régler.

> Que se passe-t-il si le fichier existe déjà ?

Il sera supprimé et remplacé par ce que vous écrivez, ce qui est problématique si l'on souhaite ajouter des informations à la fin d'un fichier pré-existant. Pensez par exemple à un fichier qui contiendrait la liste des actions effectuées par l'utilisateur : on ne veut pas tout effacer à chaque fois, on veut juste y ajouter des lignes.

Pour pouvoir écrire à la fin d'un fichier, il faut le spécifier lors de l'ouverture en ajoutant un deuxième paramètre à la création du flux : ofstream monFlux(“C:/Nanoc/scores.txt”, ios::app);.

> app est un raccourci pour append, le verbe anglais qui signifie « ajouter à la fin ».

Avec cela, plus de problème d'écrasement des données : tout ce qui sera écrit sera ajouté à la fin.

Lire un fichier

Nous avons appris à écrire dans un fichier, voyons maintenant comment fonctionne la lecture d'un fichier. Vous allez voir, ce n'est pas très différent de ce que vous connaissez déjà.

Ouvrir un fichier en lecture…

Le principe est exactement le même : on va simplement utiliser un ifstream au lieu d'un ofstream. Il faut également tester l'ouverture, afin d'éviter les erreurs.

```
ifstream monFlux("C:/Nanoc/C++/data.txt");  //Ouverture d'un fichier en lecture

if(monFlux)
{
    //Tout est prêt pour la lecture.
}
else
{
    cout << "ERREUR: Impossible d'ouvrir le fichier en lecture." << endl;
}
```

Rien de bien nouveau.

… et le lire

Il y a trois manières différentes de lire un fichier :

1. Ligne par ligne, en utilisant getline() ;
2. Mot par mot, en utilisant les chevrons » ;
3. Caractère par caractère, en utilisant get().

Voyons ces trois méthodes en détail.

Lire ligne par ligne

La première méthode permet de récupérer une ligne entière et de la stocker dans une chaîne de caractères.

```
string ligne;
getline(monFlux, ligne); //On lit une ligne complète
```

Le fonctionnement est exactement le même qu'avec cin. Vous savez donc déjà tout.

Lire mot par mot

La deuxième manière de faire, vous la connaissez aussi. Comme je suis gentil, je vous propose quand même un petit rappel.

```
double nombre;
monFlux >> nombre; //Lit un nombre à virgule depuis le
fichier
string mot;
monFlux >> mot;    //Lit un mot depuis le fichier
```

Cette méthode lit ce qui se trouve entre l'endroit où l'on se situe dans le fichier et l'espace suivant. Ce qui est lu est alors traduit en double, int ou string selon le type de variable dans lequel on écrit.

Lire caractère par caractère

Finalement, il nous reste la dernière méthode, la seule réellement nouvelle. Mais elle est tout aussi simple, je vous rassure.

```
char a;
monFlux.get(a);
```

Ce code lit une seule lettre et la stocke dans la variable a.

> Cette méthode lit réellement tous les caractères. Les espaces, retours à la ligne et tabulations sont, entre autres, lus par cette fonction. Bien que bizarres, ces caractères seront néanmoins stockés dans la variable.

Souvenez-vous de ce que nous avions vu au chapitre 5 lorsque nous avons découvert l'utilisation de cin. Nous avions appris qu'il fallait utiliser cin.ignore() lorsque l'on passait de la lecture mot par mot à la lecture ligne par ligne. Il en va de même ici. Il faudra donc écrire:

```
ifstream monFlux("C:/Nanoc/C++/data.txt");

string mot;
monFlux >> mot;          //On lit un mot depuis le fichier

monFlux.ignore();        //On change de mode

string ligne;
getline(monFlux, ligne); //On lit une ligne complète
```

Mais je vous avouerai que ce n'est pas souvent que l'on change de mode de lecture en cours de route. ;)

Lire un fichier en entier

On veut très souvent lire un fichier en entier. Je vous ai montré comment lire, mais pas comment s'arrêter quand on arrive à la fin !

Pour savoir si l'on peut continuer à lire, il faut utiliser la valeur renvoyée par la fonction getline(). En effet, en plus de lire une ligne, cette fonction renvoie un bool indiquant si l'on peut continuer à lire. Si la fonction renvoie true, tout va bien, la lecture peut continuer. Si elle renvoie false, c'est qu'on est arrivé à la fin du fichier ou qu'il y a eu une erreur. Dans les deux cas, il faut s'arrêter de lire. Vous vous rappelez des boucles ? On cherche à lire le fichier tant qu'on n'a pas atteint la fin. La boucle while est donc le meilleur choix. Voici comment faire :

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
```

```
{
   ifstream fichier("C:/Nanoc/fichier.txt");

   if(fichier)
   {
      //L'ouverture s'est bien passée, on peut donc lire

      string ligne; //Une variable pour stocker les lignes
lues

      while(getline(fichier, ligne)) //Tant qu'on n'est pas
à la fin, on lit
      {
         cout << ligne << endl;
         //Et on l'affiche dans la console
         //Ou alors on fait quelque chose avec cette ligne
         //À vous de voir
      }
   }
   else
   {
      cout << "ERREUR: Impossible d'ouvrir le fichier en
lecture." << endl;
   }

   return 0;
}
```

Une fois que l'on a lu les lignes, on peut les manipuler facilement. Ici, je me contente d'afficher les lignes mais, dans un programme réel on les utiliserait autrement. La seule limite est votre imagination. C'est la méthode la plus utilisée pour lire un fichier. Une fois que l'on a récupéré les lignes dans une variable string, on peut facilement travailler dessus grâce aux fonctions utilisables sur les chaînes de caractères.

Quelques astuces

Il ne reste que quelques astuces à voir et vous saurez alors tout ce qu'il faut sur les fichiers.

Fermer prématurément un fichier

Je vous ai expliqué en tout début de chapitre comment ouvrir un fichier. Mais je ne vous ai pas montré comment le refermer. Ce n'est pas un oubli de ma part, il s'avère juste que ce n'est pas nécessaire. Les fichiers ouverts sont automatiquement refermés lorsque l'on sort du bloc où le flux est déclaré.

```
void f()
{
   ofstream flux("C:/Nanoc/data.txt");  //Le fichier est
ouvert

   //Utilisation du fichier

}  //Lorsque l'on sort du bloc, le fichier est
automatiquement refermé
```

Il n'y a donc rien à faire. Aucun risque d'oublier de refermer le fichier ouvert.

Il arrive par contre qu'on ait besoin de fermer le fichier avant sa fermeture automatique. Il faut alors utiliser la fonction close() des flux.

```
void f()
{
   ofstream flux("C:/Nanoc/data.txt");  //Le fichier est
ouvert

   //Utilisation du fichier

   flux.close();  //On referme le fichier
                  //On ne peut plus écrire dans le fichier à
partir d'ici
}
```

De la même manière, il est possible de retarder l'ouverture d'un fichier après la déclaration du flux en utilisant la fonction open().

```
void f()
{
   ofstream flux;  //Un flux sans fichier associé

   flux.open("C:/Nanoc/data.txt");  //On ouvre le fichier
C:/Nanoc/data.txt

   //Utilisation du fichier

   flux.close();  //On referme le fichier
                  //On ne peut plus écrire dans le fichier à
partir d'ici
}
```

Comme vous le voyez, c'est très simple. Toutefois, dans la majorité des cas, c'est inutile. Ouvrir directement le fichier et le laisser se fermer automatiquement suffit.

> Certaines personnes aiment utiliser open() et close(), alors que ce n'est pas nécessaire. On peut ainsi mieux voir où le fichier est ouvert et où il est refermé. C'est une question de goût, à vous de voir ce que vous préférez.

Le curseur dans le fichier

Plongeons un petit peu plus dans les détails techniques et voyons comment se déroule la lecture. Quand on ouvre un fichier dans le bloc-note, par exemple, il y a un curseur qui indique l'endroit où l'on va écrire. Dans la figure suivante, le curseur se situe après les deux « s » sur la quatrième ligne.

Position du curseur

Si l'on tape sur une touche du clavier, une lettre sera ajoutée à cet endroit du fichier. J'imagine que je ne vous apprends rien en disant cela. Ce qui est plus intéressant, c'est qu'en C++ il y a aussi, en quelque sorte,

un curseur.

Lorsque l'on écrit la ligne suivante :

```
ifstream fichier("C:/Nanoc/scores.txt")
```

le fichier C:/Nanoc/scores.txt est ouvert et le curseur est placé tout au début du fichier. Si on lit le premier mot du fichier, on obtient bien sûr la chaîne de caractères « Nanoc » puisque c'est le premier mot du fichier. Ce faisant, le « curseur C++ » se déplace jusqu'au début du mot suivant, comme à la figure suivante.

Le curseur a été déplacé

Le mot suivant qui peut être lu est donc « : », puis « 118218 », et ainsi de suite jusqu'à la fin. On est donc obligé de lire un fichier dans l'ordre. Ce n'est pas très pratique.

Heureusement, il existe des moyens de se déplacer dans un fichier. On peut par exemple dire « je veux placer le curseur 20 caractères après le début » ou « je veux avancer le curseur de 32 caractères ». On peut ainsi lire uniquement les parties qui nous intéressent réellement.

La première chose à faire est de savoir où se situe le curseur. Dans un deuxième temps, on pourra le déplacer.

Connaître sa position

Il existe une fonction permettant de savoir à quel octet du fichier on se trouve. Autrement dit, elle permet de savoir à quel caractère du fichier on se situe. Malheureusement, cette fonction n'a pas le même nom pour les flux entrant et sortant et, en plus, ce sont des noms bizarres. Je vous ai mis les noms des deux fonctions dans un petit tableau

Pour ifstream	Pour ofstream

tellg()	tellp()

En revanche, elles s'utilisent toutes les deux de la même manière.

```
ofstream fichier("C:/Nanoc/data.txt");

int position = fichier.tellp(); //On récupére la position

cout << "Nous nous situons au " << position << "eme
caractere du fichier." << endl;
```

Se déplacer

Là encore, il existe deux fonctions, une pour chaque type de flux.

Pour ifstream	Pour ofstream
seekg()	seekp()

Elles s'utilisent de la même manière, je ne vous présente donc qu'une des deux versions.

Ces fonctions reçoivent deux arguments : une position dans le fichier et un nombre de caractères à ajouter à cette position :

```
flux.seekp(nombreCaracteres, position);
```

Les trois positions possibles sont :

- le début du fichier : ios::beg ;
- la fin du fichier : ios::end ;
- la position actuelle : ios::cur.

Si, par exemple, je souhaite me placer 10 caractères après le début du fichier, j'utilise flux.seekp(10, ios::beg);. Si je souhaite aller 20 caractères plus loin que l'endroit où se situe le curseur, j'utilise flux.seekp(20, ios::cur);. Je pense que vous avez compris.

Voilà donc notre problème de lecture résolu.

Connaître la taille d'un fichier

oc/C++/Fichiers/sco Cette troisième astuce utilise en réalité les deux précédentes. Pour connaître la taille d'un fichier, on se déplace à la fin et on demande au flux de nous dire où il se trouve. Vous voyez comment faire ? Bon, je vous montre.

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream fichier("C:/Nanoc/meilleursScores.txt");  //On
ouvre le fichier
    fichier.seekg(0, ios::end);  //On se déplace à la fin du
fichier

    int taille;
    taille = fichier.tellg();
    //On récupère la position qui correspond donc a la
taille du fichier !

    cout << "Taille du fichier : " << taille << " octets."
<< endl;

    return 0;
}
```

Je suis sûr que vous le saviez !

Voilà, on a fait le tour des notions principales. Vous êtes prêts à vous lancer seuls dans le vaste monde des fichiers.

En résumé

- En C++, pour lire ou écrire dans un fichier, on doit inclure le fichier d'en-tête <fstream>.
- On doit créer un objet de type ofstream pour ouvrir un fichier en écriture et ifstream pour l'ouvrir en lecture.
- L'écriture se fait comme avec cout : monFlux « “Texte”; tandis que la lecture se fait comme avec cin : monFlux » variable;.
- On peut lire un fichier ligne par ligne avec getline().
- Le curseur indique à quelle position vous êtes au sein du fichier, pendant une opération de lecture ou d'écriture. Au besoin, il est possible de déplacer ce curseur.

Retourner au sommaire principal

Cours, C++