HAUTOT Nolan - LEFRANC Robin | TP Gestion Logs

Itération 4 - Manipulation de Fichiers

Etape 1 : Fonction de lecture de fichier logs

Code de `menu.h` :

```cpp
/**
 * @file menu.h
 * @brief Module de gestion du menu principal
 * @author HAUTOT Nolan - LEFRANC Robin
 * @date 16/09/2025
 */

// On inclut les include guards
#ifndef MENU_H
#define MENU_H

// On ajoute les bibliothèques nécessaires
#include <iostream>
#include <fstream>

// On déclare les prototypes
/**
 * @brief Affiche le menu principal du programme
 * @details Affiche le titre et toutes les options disponibles
 */
void afficherMenu();

/**
 * @brief Traite le choix de l'utilisateur
 * @param choix Numéro de l'option sélectionnée (0-3)
 * @details Utilise un switch/case pour afficher les logs correspondants
 */
void choisirLog(int choix);

/**
 * @brief Lis le fichier log désigné
 * @param nomFichier, titre - Pour le nom du fichier - Le titre du choix
 * @details permet de lire un fichier log et renvois la valeur
 */
void lireFichierLogs(std::string nomFichier, std::string titre);

#endif // MENU_H

// ifndef, define et endif empêchent l'inclusion multiple du même fichier .h. Cela pourrait prov
```

Code de `menu.cpp` :

```cpp
/**
 * @file menu.cpp
 * @brief Implémentation du module de gestion du menu
 * @author HAUTOT Nolan - LEFRANC Robin
 * @date 16/09/2025
 */

#include "menu.h"
#include <cstdlib>

using namespace std;

void afficherMenu() // On créer la fonction et on colle le code pour afficher le menu.
{
    system("cls");
    cout << "" << endl;
    cout << "======================================================" << endl;
    cout << "HAUTOT Nolan - LEFRANC Robin | CIEL - Gestion de logs" << endl;
    cout << "======================================================" << endl;
    cout << "" << endl;
    cout << "Menu principal :" << endl;
    cout << "" << endl;
    cout << "1 - Voir les evenements systeme" << endl;
    cout << "2 - Voir les connexions reseau" << endl;
    cout << "3 - Voir les actions utilisateur" << endl;
    cout << "4 - Voir les connexions utilisateur" << endl;
    cout << "0 - Quitter" << endl;
    cout << "" << endl;
    cout << "Votre choix : ";
}

void choisirLog(int choix) // On créer la fonction et on colle le code pour gérer le choix de l
{
    switch (choix) // Ici on utilise switch à la place de if/else if pour une meilleure lisibil
    {
    case 1: // Les 'case' sont l'equivalent des 'if'.
        lireFichierLogs("logs_test/windows_events.txt", "EVENEMENTS SYSTEME");
        break; // Le 'break' est obligatoire afin d'eviter une execution continue.
    case 2:
        lireFichierLogs("logs_test/network_logs.txt", "CONNEXIONS RESEAU");
        break;
    case 3:
        lireFichierLogs("logs_test/user_actions.txt", "ACTIONS UTILISATEUR");
```

```
        break;
    case 4:
        cout << "Affichage des connexions utilisateur...";
        break;
    case 0:
        cout << "Au revoir !";
        break;
    default: // Le 'default' est l'equivalent du 'else' final.
        cout << "Erreur ! Le choix " << choix << " n'est pas un choix valide !";
    }
}

void lireFichierLogs(string nomFichier, string titre) // On créer la fonction avec comme paramèt
{
    cout << "\n=== " << titre << " ===" << endl; // On met le titre, par exemple "CONNEXIONS RES

    // Déclarer un objet ifstream
    // Pour GCC 5.1, ajouter .c_str()
    ifstream fichier(nomFichier);

    if(fichier.is_open()) // On regarde si le fichier est ouvert
    {
        string ligne; // On créer un string ligne
        int compteur = 0; // On initialise un compteur à 0

        while(getline(fichier,ligne)) // Ici on fait une lecture ligne par ligne du fichier
        {
            cout << ligne << endl; // On renvoie la valeur d'une ligne
            compteur++; // On ajoute 1 au compteur
        }

        fichier.close(); // On ferme le fichier

        cout << "\nTotal : " << compteur << " evenements lus" << endl; // On renvoie le nombre d
    } else
    {
        cout << "Erreur : Impossible d'ouvrir le fichier " << nomFichier << endl; // Sinon on re
    }
}
```

> *Question 1 :* Pourquoi utiliser `getline()` plutôt que >> ?

- On utilise `getline()` à la place de >> pour lire ligne par ligne le fichier log. Ca rend le code plus lisible et on l'effectue en une ligne.

> *Question 2 :* Que se passe-t-il si on oublie fichier.close() ?

- Il ne se passe rien visuellement mais le fichier reste ouvert en fond par notre programme. De ce fait, si quelqu'un veut vérifier le fichier il ne pourra pas car il est déjà ouvert pour notre programme.

> *Question 3 :* Comment adapter le chemin si on change la structure des dossiers ?

- Si l'on change la structure des dossiers, il faut bien penser à l'indiquer. Par exemple `"logs_test/windows_events.txt"` , si nous le mettons dans un dossier "windows" nous allons devoir écrire `"logs_test/windows/windows_events.txt"` .

> *Question 4 :* Pourquoi passer le titre en paramètre plutôt que l'écrire en dur ?

- Cela permet pour chaque choix d'afficher le titre sans qu'on ai besoin de faire 4 conditions pour vérifier si tel choix est sélectionné, on affiche son titre. En une ligne, on peut faire toute cette vérification et l'afficher sans problème. La fonction s'occupe de tout ça.

Etape 2 : Génération de rapports automatisés

Code de `menu.h` :

```
/**
 * @file menu.h
 * @brief Module de gestion du menu principal
 * @author HAUTOT Nolan - LEFRANC Robin
 * @date 16/09/2025
 */

// On inclut les include guards
#ifndef MENU_H
#define MENU_H

// On ajoute les bibliothèques nécessaires
#include <iostream>
#include <fstream>

// On déclare les prototypes
/**
 * @brief Affiche le menu principal du programme
 * @details Affiche le titre et toutes les options disponibles
 */
void afficherMenu();

/**
 * @brief Traite le choix de l'utilisateur
 * @param choix Numéro de l'option sélectionnée (0-3)
 * @details Utilise un switch/case pour afficher les logs correspondants
 */
void choisirLog(int choix);

/**
 * @brief Lis le fichier log désigné
 * @param nomFichier, titre - Pour le nom du fichier - Le titre du choix
 * @details permet de lire un fichier log et renvois la valeur
 */
void lireFichierLogs(std::string nomFichier, std::string titre);

/**
 * @brief Génère un rapport
 * @details Permet de générer un rapport détaillé.
 */
void genererRapport();

/**
```

```
 * @brief Analyse les fichiers
 * @param nomFichier, typeLog, rapport - Pour le nom du fichier - Le dossier log étudié - Un rap
 * @details Permet d'analyser les fichiers et d'en faire un rapport
 */
void analyserFichier(std::string nomFichier, std::string typeLog, std::ofstream& rapport);

#endif // MENU_H

// ifndef, define et endif empêchent l'inclusion multiple du même fichier .h. Cela pourrait prov
```

Code de `menu.cpp` :

```cpp
/**
 * @file menu.cpp
 * @brief Implémentation du module de gestion du menu
 * @author HAUTOT Nolan - LEFRANC Robin
 * @date 16/09/2025
 */

#include "menu.h"
#include <cstdlib>

using namespace std;

void afficherMenu() // On créer la fonction et on colle le code pour afficher le menu.
{
    system("cls");
    cout << "" << endl;
    cout << "======================================================" << endl;
    cout << "HAUTOT Nolan - LEFRANC Robin | CIEL - Gestion de logs" << endl;
    cout << "======================================================" << endl;
    cout << "" << endl;
    cout << "Menu principal :" << endl;
    cout << "" << endl;
    cout << "1 - Voir les evenements systeme" << endl;
    cout << "2 - Voir les connexions reseau" << endl;
    cout << "3 - Voir les actions utilisateur" << endl;
    cout << "4 - Generer rapport de synthese" << endl;
    cout << "0 - Quitter" << endl;
    cout << "" << endl;
    cout << "Votre choix : ";
}

void choisirLog(int choix) // On créer la fonction et on colle le code pour gérer le choix de l
{
    switch (choix) // Ici on utilise switch à la place de if/else if pour une meilleure lisibil
    {
    case 1: // Les 'case' sont l'equivalent des 'if'.
        lireFichierLogs("logs_test/windows_events.txt", "EVENEMENTS SYSTEME");
        break; // Le 'break' est obligatoire afin d'eviter une execution continue.
    case 2:
        lireFichierLogs("logs_test/network_logs.txt", "CONNEXIONS RESEAU");
        break;
    case 3:
        lireFichierLogs("logs_test/user_actions.txt", "ACTIONS UTILISATEUR");
```

```
        break;
    case 4:
        genererRapport();
        break;
    case 0:
        cout << "Au revoir !";
        break;
    default: // Le 'default' est l'equivalent du 'else' final.
        cout << "Erreur ! Le choix " << choix << " n'est pas un choix valide !";
    }
}

void lireFichierLogs(string nomFichier, string titre) // On créer la fonction avec comme paramèt
{
    cout << "\n=== " << titre << " ===" << endl; // On met le titre, par exemple "CONNEXIONS RES

    // Déclarer un objet ifstream
    // Pour GCC 5.1, ajouter .c_str()
    ifstream fichier(nomFichier);

    if(fichier.is_open()) // On regarde si le fichier est ouvert
    {
        string ligne; // On créer un string ligne
        int compteur = 0; // On initialise un compteur à 0

        while(getline(fichier,ligne)) // Ici on fait une lecture ligne par ligne du fichier
        {
            cout << ligne << endl; // On renvoie la valeur d'une ligne
            compteur++; // On ajoute 1 au compteur
        }

        fichier.close();

        cout << "\nTotal : " << compteur << " evenements lus" << endl; // On renvoie le nombre d
    } else
    {
        cout << "Erreur : Impossible d'ouvrir le fichier " << nomFichier << endl; // Sinon on re
    }
}

void genererRapport() // On créer la fonction sans paramètre
{
    cout << "\n=== GENERATION DU RAPPORT ===" << endl; // On affiche le titre
```

```
    ofstream rapport("logs_test/rapport_synthese.txt"); // On génère un txt ou sera écrit le ra

    if(rapport.is_open()) // Si rapport est ouvert
    {
        rapport << "========================================" << endl;
        rapport << "RAPPORT DE SYNTHESE DES LOGS - BTS CIEL" << endl;
        rapport << "Date de generation : " << __DATE__ << endl;
        rapport << "========================================" << endl << endl;

        analyserFichier("logs_test/windows_events.txt", "Evenements systeme", rapport);
        analyserFichier("logs_test/network_logs.txt", "Connexions reseau", rapport);
        analyserFichier("logs_test/user_actions.txt", "Actions utilisateur", rapport);

        rapport << endl << "--- FIN DU RAPPORT ---" << endl;

        rapport.close();
        cout << "Rapport genere avec succes : logs_test/rapport_synthese.txt" << endl;
    } else
    {
        cout << "Erreur : Impossible de creer le fichier rapport" << endl; // Sinon on renvoie u
    }
}

void analyserFichier(string nomFichier, string typeLog, ofstream& rapport) // On créer la fonct
{
    ifstream fichier(nomFichier); // On déclare un objet fichier

    if(fichier.is_open()) // Si le fichier est ouvert
    {
        string ligne; // On créer un string ligne
        int compteur = 0; // On initialise un compteur à 0

        rapport << "--- " << typeLog << " ---" << endl; // Ici on renvoie le titre du log qu'on

        while(getline(fichier,ligne))
        {
            compteur++; // On rajoute 1 au compteur à chaque ligne
        }

        rapport << "Nombre total d'evenements : " << compteur << endl; // On affiche le nombre t
        rapport << endl;
```

```
        fichier.close(); // On ferme le fichier
    } else
    {
        rapport << "ERREUR : Fichier " << nomFichier << " non trouvé" << endl << endl;
    }
}
```

Code de `main.cpp` :

```cpp
/**
 * @file main.cpp
 * @brief Programme de gestion de logs - Iteration 4
 * @author HAUTOT Nolan - LEFRANC Robin
 * @date 16/09/2025
 * @version 4.0
 */

/**
 * @brief Fonction principale du programme
 * @details Affiche un menu en boucle jusqu'à ce que
 * l'utilisateur choisisse de quitter (choix=0)
 * @return 0 si le programme se termine normalement
 */

#include <iostream>
#include <string>

// On inclut le module Menu qu'on a créer
#include "Menu/menu.h"

using namespace std;

int main()
{
    // Variable pour stocker le choix
    int choix = 0;

    do // On effectue une boucle do-while
    {
        // Affichage du Menu "Gestion de logs"
        afficherMenu(); // On appelle la fonction afficherMenu()

        // Demander le choix de l'utilisateur
        cin >> choix;

        choisirLog(choix); // On appelle la fonction choisirLog(choix)

        if (choix != 0)
        {
            cout << "\nAppuyer sur Entree...";
            cin.ignore(); // Vide le buffer
            cin.get();    // Attend Entrée
```

```
        }

    } while (choix != 0); // Ici tant que choix est différent de 0, on continue la boucle.

    return 0;
}
```

Fichier rapport_synthese.txt

Voici l'intérieur du fichier rapport_synthese.txt :

```
==========================================
RAPPORT DE SYNTHESE DES LOGS - BTS CIEL
Date de generation : Sep 16 2025
==========================================

--- Evenements systeme ---
Nombre total d'evenements : 7

--- Connexions reseau ---
Nombre total d'evenements : 7

--- Actions utilisateur ---
Nombre total d'evenements : 7


--- FIN DU RAPPORT ---
```