

Projet de fin d'année C & Web - CIR1

Filmothèque

Leandro Montero & Sylvain Lefebvre

leandro.montero@isen-ouest.yncrea.fr

sylvain.lefebvre@isen-ouest.yncrea.fr



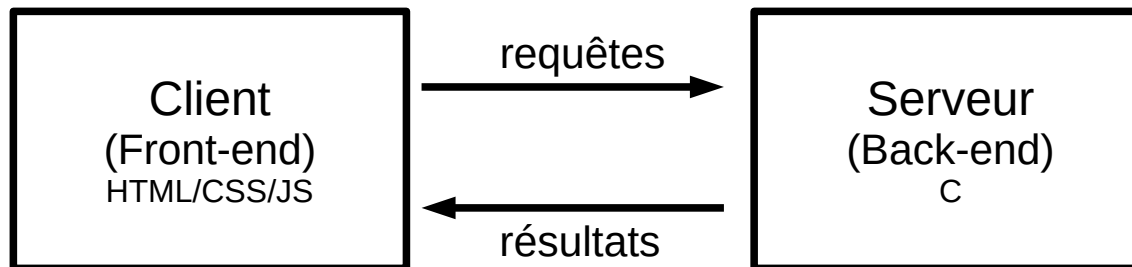
Présentation

Sujet : Filmothèque



Sujet : Filmothèque

- Vous allez développer un système permettant d'obtenir rapidement informations sur des films.
- La architecture du système sera la suivante



Client (Front-end):

- Vous allez créer un site web qui va permettre à l'utilisateur de faire des requêtes au Serveur pour obtenir des informations sur les films puis pouvoir visualiser les résultats envoyés par le Serveur.
- À développer en HTML, CSS et JS.

Serveur (Back-end):

- Vous allez créer un programme en C qui jouera le rôle du Serveur dans le système.
- Quand vous exécutez votre programme, il lira la BD fournie puis la stockera dans des structures de données nécessaires.
- Après il se mettra à l'écoute des requêtes du Client.
- Dès qu'une requête arrive, le Serveur va l'exécuter puis envoyer la réponse au Client.

Communication Client-Serveur :

- La communication entre le Client et le Serveur se fera avec des fichiers de texte.
- Une requête envoyée depuis le Client créera un fichier «request.txt» avec les informations de la requête (la fonction JS qu'écrit un fichier vous sera donnée).
- Le Serveur trouvera ce fichier, l'analysera, exécutera la requête demandée, créera un fichier «results.txt» avec le résultat et finalement effacera le fichier «request.txt» pour se remettre à l'écoute.
- Le Client va lire le fichier «results.txt» puis affichera les résultats (la fonction JS qui lit un fichier vous sera donnée).

Note : En pratique, ce genre de fonctionnement de création d'un fichier doit être entièrement sous contrôle pour éviter tout problème de sécurité !

Présentation

Exemple de flux du système :

- 1) Serveur : Exécution « ./filmothèque BD.txt »
- 2) Serveur : Lecture du fichier BD plus remplissage de structures de données.
- 3) Serveur : Essayer de lire le fichier «request.txt» tant qu'il n'existe pas.
- 4) Client : Ouverture du site web «index.html».
- 5) Client : Choisir sur un menu, la requête à faire, ex chercher par réalisateur (director en anglais).
- 6) Client : Remplir un formulaire avec les paramétrés de recherche, ex director = Tarantino puis valider.
- 7) Client : Le fichier «request.txt» sera créé avec les infos de la requête, ex «findByDirector Tarantino».
- 8) Serveur : Le fichier «request.txt» sera trouvé et lu.

- 9) Serveur : La fonction permettant d'exécuter la requête demandée sera appelée, ex `findByDirector(«Tarantino»)`.
- 10) Serveur : Le résultat sera enregistré dans le fichier «results.txt». Ex de fichier :

```
0.15  
Django Unchained;165;Western  
Kill Bill: Vol. 1;111;Action  
Reservoir Dogs;99;Crime
```

où 0.15 est le temps d'exécution de la requête en secondes, puis on a une ligne par film avec `title;durée en minutes;genre`.
- 11) Serveur : Le fichier «request.txt» est effacé puis 3)
- 12) Client : Le fichier «results.txt» sera trouvé puis les résultats seront lus et affichés d'une façon jolie.
- 13) Client : Il aura l'option de revenir sur la page d'accueil puis 4)
- 14) Client : On pourra envoyer une requête pour terminer le Serveur.
- 15) Serveur : Une fois reçu l'ordre pour terminer l'exécution, il devra supprimer la filmothèque puis s'arrêter.

Fonctionnalités minimales Client:

- Site web comprenant plusieurs pages (fichiers .html) avec un design «joli» grâce aux technologies HTML et CSS.
- Chercher films par réalisateur.
- Chercher films par durée.
- Chercher le réalisateur qui a fait le plus grand nombre de films et connaître ce nombre.
- Valider tous les formulaires avec JS.
- Visualiser les résultats des requêtes avec leur temps d'exécution.
- Envoyer une requête pour terminer le Serveur.
- D'autres fonctionnalités que vous voulez ajouter !

Fonctionnalités minimales Serveur:

- Séparation de fichiers de code source (main.c pour le programme principal puis plusieurs .c .h selon besoin).
- Pouvoir mesurer le temps d'exécution de requêtes (compter seulement le temps d'exécution de la fonction qui calcule le résultat sans tenir en compte la lecture/écriture de fichiers, etc).
- Obtenir tous les films d'un réalisateur doit se faire en $O(m)$ où m est le nombre de caractères du nom du réalisateur.
- Obtenir le réalisateur qui a fait le plus grand nombre de films doit se faire en $O(1)$. Obtenir ce nombre de films doit être $O(1)$ aussi.
- Obtenir tous les films d'une durée donnée doit être $O(1)$.
- Supprimer toute la filmothèque.
- D'autres fonctionnalités que vous voulez ajouter !

Fonctionnalités additionnelles Client/Serveur, quelques idées :

- Visualiser tout le contenu de la filmothèque, où on doit pouvoir visualiser par réalisateur tous ses films par exemple.
- Visualiser les résultats triés alphabétiquement (réalisateurs et films).
- Insérer un nouveau film.
- Chercher films par catégorie.
- Chercher un film par son nom.
- Créer des requêtes mixtes, par exemple, chercher tous les films d'un réalisateur d'une catégorie donnée, chercher tous les films d'une catégorie avec une durée entre X et Y minutes.
- Adapter vos structures de données pour faire les requêtes précédentes d'une façon efficace !
- Pagination des résultats (par exemple 20 par page).
- ...

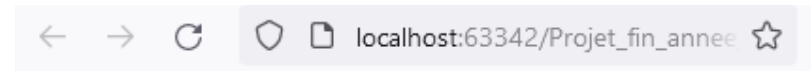
Exemple d'utilisation écriture/lecture de fichiers en JS :

Vous avez la fonction JS :

```
function writeFile(id_form, func)
```

Puis dans votre html vous avez :

```
<form id="form-findByDirector" action="result.html">  
Director's name: <input type="text" name="name"><br>  
<input type="submit" id="go" value="GO!">  
</form>
```



Puis dans votre navigateur vous avez :

Find By Director

Director's name:

Vous devriez ajouter un événement «click» au bouton d'id="go" tel que quand on clique sur «GO!», la fonction writeFile de JS sera appelé avec l'id du formulaire et le nom de la fonction C qui exécutera la requête, donc, le fichier «request.txt» sera créé et finalement on sera redirigé sur la page «result.html».

Présentation

La fonction `writeFile` créera le fichier «request.txt» toujours de la même façon :

```
functionName;formField1;formField2;.....;formFieldN
```

Le contenu dans notre exemple sera :



Finalement, dans le Serveur, on saura au moment de lecture du fichier qu'on doit appeler la fonction `findByDirector` avec les paramètres respectifs. Après exécution, le fichier «results.txt» sera créé par votre programme en C.

Pour lire les fichiers, vous avez la fonction JS : `function readFile()`

Donc, pour lire les résultats de l'exécution de la requête, dans votre fichier `result.html`, vous allez appeler la fonction `readFile` qui va vous renvoyer un string avec le contenu du fichier «results.txt». Par exemple :

```
let myResults = readFile();  
console.log(myResults);
```

Affichera dans la console tout le contenu du fichier «results.txt».

ATTENTION ! Pour ne pas avoir des problèmes de synchronisation, dans votre programme C, vous allez écrire le fichier «results.txt» et une fois fini, vous allez créer un fichier vide appelé «ready.txt».

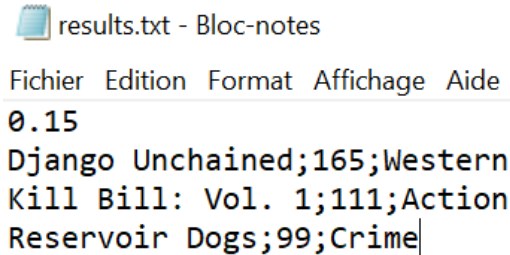
La fonction `readFile` va essayer de trouver ce dernier fichier et donc, une fois qu'il a été trouvé, le contenu du fichier «results.txt» sera lu et renvoie.

Cette manipulation est faite seulement pour ne pas commencer à lire le fichier «results.txt» AVANT qu'il soit fini d'être écrit !

N'oubliez pas d'effacer le fichier «ready.txt» après !

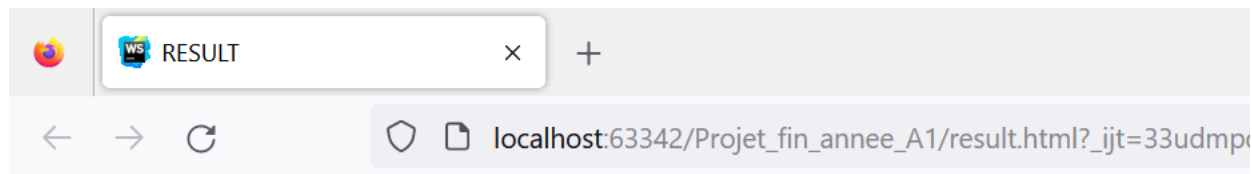
Présentation

Finalement, si votre fichier «results.txt» contient par exemple :



```
results.txt - Bloc-notes  
Fichier Edition Format Affichage Aide  
0.15  
Django Unchained;165;Western  
Kill Bill: Vol. 1;111;Action  
Reservoir Dogs;99;Crime
```

Dans la page result.html de votre navigateur vous allez avoir (bien plus joli :D) :



Films by Tarantino:

Elapsed time: 0.15ms

Name	Time	Category
Django Unchained	165 minutes	Western
Kill Bill: Vol. 1	111 minutes	Action
Reservoir Dogs	99 minutes	Crime

Phases du projet

Phase 1 : préparation

- Présentation du projet.
- Lecture et compréhension du sujet.
- Démarrage du projet : choix de structures de données et implémentations de base.
- Écriture du cahier des charges / planification.

Rendu sur moodle !

Phase 2 : mise en œuvre

- Implémentation des fonctionnalités principales.
- Tests et validation.
- Implémentation des fonctionnalités additionnelles (et tests, etc).
- Préparation de la soutenance (présentation,démo,rendu code).

Rendu sur moodle !

■ Cahier des charges / planification

- Lecture et compréhension du cahier des charges.
- Identification des tâches.
- Gestion des dépendances.
- Répartition des tâches dans le temps et par personne.
- Identification des points difficiles et/ou bloquants dans l'implémentation.

	J1	J1	J2	J2	J3	J3	J7
Eleve1		A ?	B ?						
Eleve2		A ?	B ?						

• Identification des tâches

- Tâche A : structure Film
- Tâche B : lire fichier BD et remplir structures

• Gestion des dépendances

- B dépend de A

• Répartition des tâches dans le temps

- Traiter A avant B, temps nécessaire...

• Répartition des tâches par personnes

- Elève 1 fait A, Elève 2 fait B (attente ...)
- Elève 1 fait A puis B (seul)
- Elève 1 et 2 travaillent en binôme sur A puis B (en échangeant)

Phase 1 : étude préliminaire

- **Cahier des charges / planification**
 - Prévoir les ajouts au projet.
 - Décrire et décomposer en sous-tâches : spécifiques, mesurables, atteignables, réalistes, temporelles.
 - Ajouter ces tâches au planning.
 - Ne pas oublier les tâches suivantes :
 - Écriture du cahier des charges.
 - Préparation de la soutenance.
 - Relecture du code.
 - Réunions de mise au point.
 - Création et dépôt de l'archive (éviter de s'y prendre à la dernière minute...).
 - Etc...

Phase 2 : implémentation

- Implémenter le code (et le tester!).
- Travail à 3 : relire et commenter le code de l'autre.
- Communiquer et échanger le plus possible avec son équipe : attention aux incompréhensions qui peuvent engendrer des catastrophes (phase "d'intégration").
- Sauvegarder régulièrement son code pour éviter la réflexion "ce matin, ça marchait, mais là, plus rien ne fonctionne" !
- Préparer la soutenance et rendu final du code.



Organisation du projet (30h)

Jeudi 08/06 au Mercredi 14/06

Jeudi	Vendredi	WE	WE
Présentation	Étude (Rendu 13h30)		
Étude	Implémentation		

Lundi	Mardi	Mercredi
Implémentation	Implémentation	Finalisation du code (Rendu 12h)
Implémentation	Implémentation / Relecture	Soutenances

- Horaire : 9h-12h et 13h30-16h30. PAS DE TÉLÉTRAVAIL !
- Salles : regardez webAurion !

Rendus

- Étude préliminaire → vendredi 09/06, 13h30 sur Moodle
Doit inclure (fichier .pdf) :
 - Un cahier des charges fonctionnel.
 - Des spécifications techniques.
 - Un planning prévisionnel détaillé.
- Rendu du code + support de la présentation → mercredi 14/06, 12h00 sur Moodle
Doit inclure (fichier .zip) :
 - Rendu du code source : code organisé et commenté + 1 fichier `readme.md` contenant les procédures de compilation et d'exécution de votre programme.
 - Rendu du support de présentation (pdf ou ppt).
- Soutenances : Mercredi 14/06 après midi.
- Bilan du projet : Mercredi 14/06 après la dernière soutenance.

Attention

Archives sur moodle : nommage : `projetCIR1_groupe_X.(pdf ou zip)`
X représente votre numéro de groupe. Tout retard sera sanctionné (l'heure du réseau fait foi). Les fichiers au mauvais format ou avec mauvais nommage seront pénalisés.

- **Cahier des charges fonctionnel**
 - Reformulation des attendus du projet.
 - Engagement sur la liste des fonctionnalités de l'application.
 - Prototypes du site web.
- **Spécifications techniques**
 - Technologies utilisées.
 - Choix de structures de données et justification.
- **Planning prévisionnel (le planning doit évoluer avec le temps)**
 - Identification des tâches.
 - Planification dans le temps.
 - Affectation de personnes.

La soutenance

- ♣ Timing :
 - 10 minutes de présentation.
 - 7 minutes de démonstration
 - 5 minutes de questions.
- ♣ Doit être répétée.
- ♣ Doit utiliser des supports.
- ♣ Tous les membres du groupe doivent prendre la parole.

Présentation (10 min)

■ Contenu attendu :

- Rappel du contexte (besoin, organisation, etc.).
- Vos grands choix de conception/structures de données, avec les arguments clés.
- Ce qui fonctionne et ne fonctionne pas.
- Captures d'écran/prototypes.
- Bilan et perspectives (principales difficultés rencontrées, notions apprises, ce qui est améliorable, etc.).

■ Conseils :

- Pas de diapo vide.
- Évitez de mettre du code.
- Privilégiez les schémas, en ne gardant que l'information pertinente.
- Tout ce qui est présent dans une diapo doit être utile (et présenté). Sinon, épurez !

La démonstration

- ♣ Doit être rapide et efficace (7 minutes maximum).
- ♣ Doit être préparée à l'avance.
- ♣ Doit montrer les fonctionnalités les plus importantes de votre projet.
- ♣ "Vendre votre projet à des techniciens".
- ♣ Attention : votre évaluateur est susceptible de :
 - poser des questions.
 - demander des manipulations non prévues à l'avance.

- Barème indicatif :
 - Phase 1 : étude préliminaire : 25%
 - Rendu du code final : 25%
 - Présentation et démonstration : 50%
- Remarques :
 - La présentation sera évaluée **individuellement**.
 - Malus possible sur des membres du groupe si l'investissement est jugé trop faible.
 - Possibilité d'être interrogé durant le projet de façon individuelle.
 - Plagiat sévèrement sanctionné pour TOUS les membres du/des groupe(s). ATTENTION à chatGPT !!
 - Le code sera évalué uniquement d'un point de vue fonctionnel. Toutefois, des mauvaises pratiques significatives seront sanctionnées par des malus.

Mises en garde

- Le ou les encadrants ne font pas le code à votre place. **Ils vous aident** éventuellement à **corriger vos bugs** et à trouver vos erreurs.
- **Votre code est unique.** Copier/coller le code source d'un autre groupe vous expose, ainsi que vos camarades, à une sanction dans la note du projet.
Pareil si vous copiez du code chez chatGPT !
- **Sauvegardez régulièrement** votre travail. Dès que vous avez une version opérationnelle, mettez là de côté au cas où...

Mises en garde

- **Ne comptez pas sur le mercredi** pour terminer le travail (dernier jour réservé aux finitions et commentaires).
- **Validez étape par étape**, découpez votre code en fonctions, testez le plus souvent possible. Réfléchissez à des solutions partielles pour pouvoir avancer.
- Ce projet pour être mené à bien nécessite **un investissement personnel et de groupe**, et de nombreux tests.

Salle moodle

- INSCRIVEZ VOUS ! Tout est là (et tout se passera là aussi!)

<https://web.isen-ouest.fr/moodle/course/view.php?id=413>

Des questions ?



Bon projet !

