Barbier Frédéric GI02
Lourd Rodolphe GI02

EDITION COLLABORATIVE :

RAPPORT

SOMMAIRE

1. Cahier des charges

On souhaite réaliser un éditeur classique (pas d'interface graphique) qui permette l'édition collaborative, c'est-à-dire que plusieurs personnes peuvent accéder à un document simultanément. Cet éditeur ne prendra pas en charge le formatage de texte (gras, italique…) ; quatre fonctions sont attendues dans cet éditeur : créer un document, ouvrir un document, modifier/enregistrer un document et fermer un document.

Afin de pouvoir choisir une fonction à n'importe quel moment de l'édition d'un document, il est nécessaire de réserver une console pour un menu répertoriant les fonctions, et ce pour chaque utilisateur. Le menu se présentera sous la forme d'une liste de choix numérotés de un à quatre. Au lancement de l'éditeur, la console affiche le menu ; l'utilisateur a le choix entre la création et l'ouverture d'un document. Ensuite, une fois un document ouvert ou créé, l'utilisateur a la possibilité de le modifier ou de le fermer.

a. Création

Lorsque l'option création est lancée, on demande un nom de fichier à l'utilisateur. Le programme vérifie notamment que le nom de fichier ne comporte de caractères interdits ; il vérifie aussi la longueur de ce nom. Si le nom est incorrect, on demande à l'utilisateur de saisir un autre nom. On note que le fichier est créé dans un répertoire fixe accessible à tous, par exemple //gi/lo41, c'est-à-dire le même répertoire utilisé pour l'emplacement de l'éditeur.

b. Ouverture

Lorsque l'utilisateur appelle la fonction ouverture, l'éditeur va demander un nom de fichier à cet utilisateur ; on vérifie que ce nom de fichier existe dans le répertoire du programme. Si ce n'est pas le cas, le programme demande à l'utilisateur de saisir un nouveau nom de fichier valide. Quand le nom de fichier existe, l'éditeur ouvre ce document dans une nouvelle console réservée à l'affichage et au rafraîchissement du document ; si ce document est déjà ouvert, le programme le notifie à l'utilisateur sans le rouvrir.

c. Modification

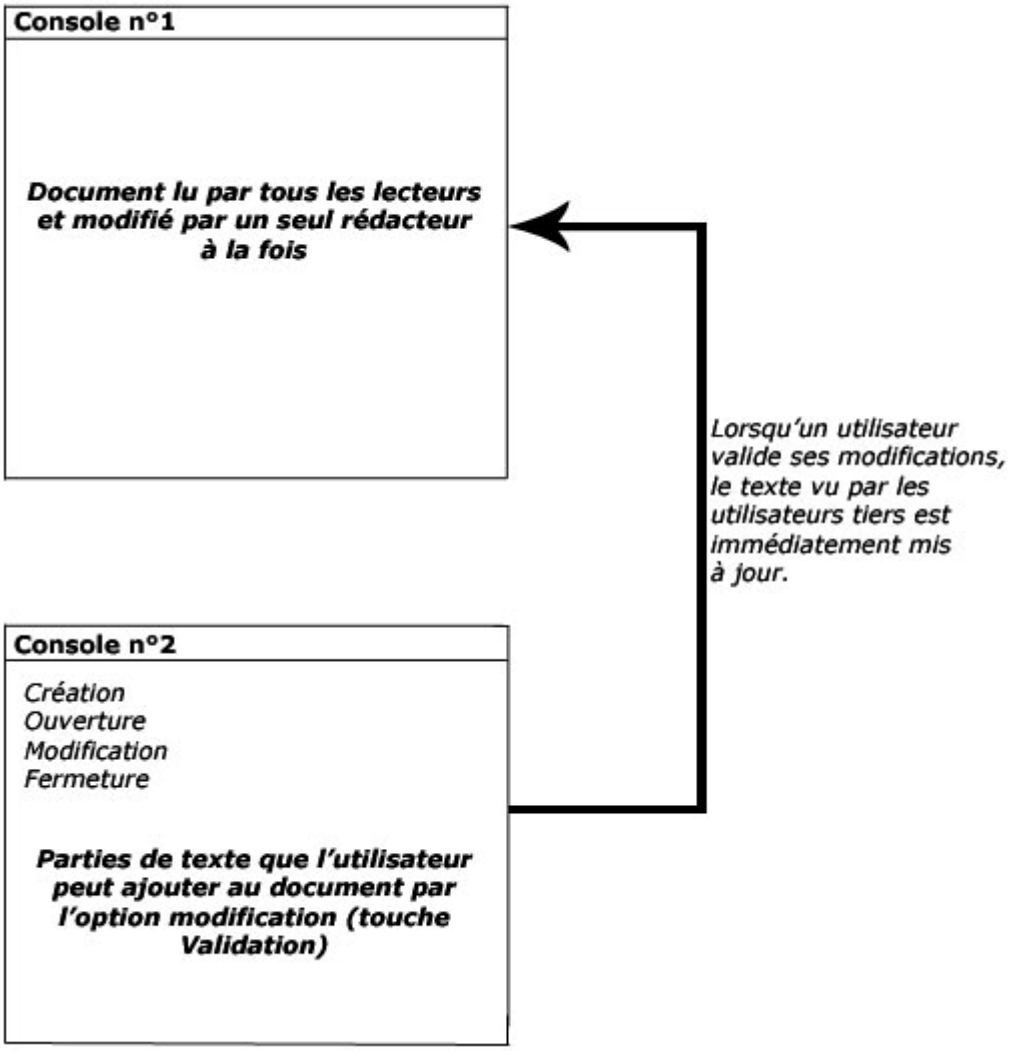
La fonction modification est assurée par la touche *validation* (*entrée*). Quand un utilisateur appuie sur cette touche, la ligne qu'il vient de taper est ajoutée au document ouvert dans sa console d'affichage ainsi que celle de chaque utilisateur. Un enregistrement est également effectué à cet instant.

d. Fermeture

Lorsque l'utilisateur choisit l'option fermeture, seule la fenêtre d'affichage du document est fermée, l'autre console affichant le menu et la saisie restant ouverte, pour une autre ouverture de fichier éventuelle.

e. Schéma de l'interface

L'interface très simple se présente de la manière suivante :



Chaque utilisateur ouvre deux consoles lorsqu'il lance le programme. La *console n°1* montre le document à l'ouverture ; la *console n°2* offre les commandes nécessaires telles que *création*, *ouverture*, *modification* et *fermeture*. Les modifications potentielles apportées au document sont d'abord rédigées dans la *console n°2* ; lorsque l'utilisateur décide des les valider, il choisit la commande *Modification* (touche *Entrée*). A ce moment-là s'applique le *principe d'exclusion mutuelle* qui empêche un autre utilisateur de valider aussi ses modifications pendant le très court laps de temps dont la première modification a besoin.

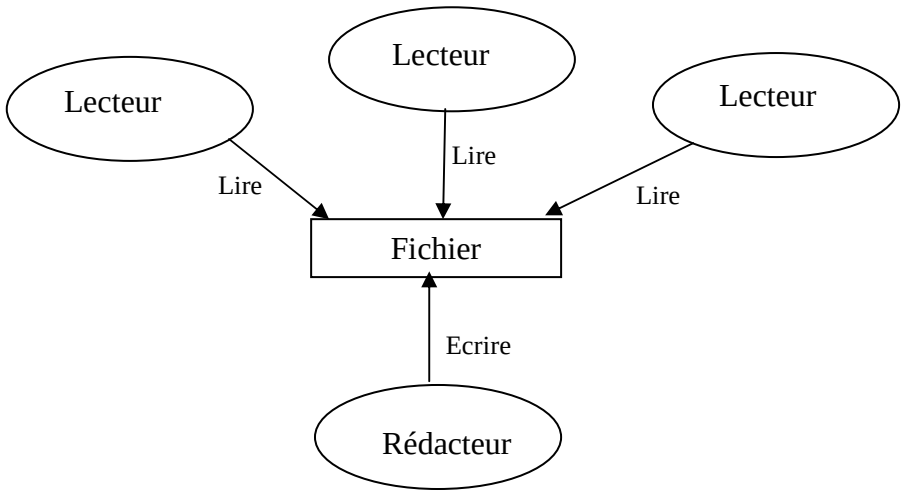
Une fois la modification effectuée, les autres utilisateurs voient leur document mis à jour en temps réel.

2. Spécifications

a. Synchronisation entre les processus lecteurs et rédacteurs

Le fichier créé est une ressource commune à tous les processus utilisateurs, certains de ces processus étant considérés comme *lecteurs* (ceux voulant lire dans le fichier) et d'autres comme *rédacteurs* (ceux voulant écrire dans le fichier).

On obtient donc le modèle des Lecteurs-Rédacteurs :



Principes

- La lecture est possible, simultanément par plusieurs lecteurs, à condition qu'il n'y ait aucune écriture.
- L'écriture n'est possible que s'il n'y a ni lecture ni autre écriture en cours.

Pour synchroniser ces processus, nous devons appliquer une méthode d'exclusion mutuelle. Nous pensons que la synchronisation *Lecteurs-Rédacteurs* par sémaphores est la plus adaptée. L'algorithme d'une modification dans notre cas serait le suivant :

```
procedure modification (verrouillage, texte, document)
début
        tant que (verrouillage = vrai) faire
            …
        fintantque
        sinon verrouillage := vrai
                document := document + texte
                verrouillage := faux
        finsi
fin
```

Dans cette procédure, *verrouillage* est une variable booléenne globale qui indique si le document est bloqué en écriture ou non. Ce type de procédure est lancé chaque fois qu'un utilisateur valide sa modification ; ainsi on évite les modifications simultanées

préjudiciables au bon fonctionnement de l'éditeur. La boucle *tant que* met simplement en attente l'utilisateur d'un déverrouillage du document en écriture.

b. Envoi de signaux pour la communication entre terminaux :

Pour rafraîchir les fenêtre d'affichage utilisateurs ou encore pour fermer sa fenêtre d'affichage à la fermeture du fichier, nous utiliserons l'envoi de signaux aux terminaux correspondants et donc, il nous faudra connaître le *pid* de chaque console d'affichage. Nous récupéreront donc ce *pid* à l'ouverture du fichier et, pour que chaque processus utilisateur puisse accéder à ces numéros, nous les stockerons dans un fichier. Nous devrons bien sûr mettre à jour ce fichier, c'est-à-dire qu'à chaque nouvelle ouverture de fichier, nous rajouterons un *pid* et à chaque fermeture de fichier, nous effacerons le *pid* correspondant.

Comme nous allons de nouveau utiliser un fichier avec des opérations de lecture écriture, nous devons bien sûr synchroniser les processus à l'aide de l'exclusion mutuelle décrite précédemment. Ainsi, nous utiliserons deux systèmes d'exclusion mutuelle similaires.

Pour conclure, nous devons préciser que ces idées de conception sont celles qui nous semble les plus adaptées au vu de nos connaissances actuelles. Il est par conséquent possible que durant la phase de codage, des difficultés imprévues nous incitent à rechercher des solutions différentes qui seraient plus facilement réalisables.

Réalisation