Gestion d'un jeu chronométré

***Mots-clés** : Communication interprocessus, mémoire partagée, signaux, synchronisation*

Spécifications du problème à résoudre

On souhaite implémenter une application de gestion d'un jeu à 2 joueurs où la saisie d'un coup est chronométrée (partie d'échecs, par exemple).

1. Chaque joueur dispose d'une horloge qui lui indique le temps restant pour le coup en cours. Si le temps alloué est épuisé, la partie est terminée et l'autre joueur est déclaré gagnant.

2. Quand un joueur a effectué sa saisie et que le coup est déclaré valide, son horloge est stoppée et l'horloge de l'autre joueur est activée.

3. Un menu permet de gérer le déroulement de la partie. Il offre les choix suivants :

```
1 : lancement d'une nouvelle partie
2 : connexion à une partie déjà lancée par un autre joueur
3 : reprendre une partie préalablement interrompue et sauvegardée
4 : interruption d'une partie avec sauvegarde
5 : mise en pause d'une partie
6 : reprise de la partie
7 : visualisation de l'historique de la partie en cours
8 : fin de l'application avec visualisation de l'historique
9 : fin de l'application
/ : indique que la suite de la saisie est un coup joué.
```

4. Lancement de l'application

   Un utilisateur qui souhaite jouer lance l'application dans un terminal. Les parties en cours sont affichées à l'écran ainsi que le menu. Il peut alors choisir de lancer une nouvelle partie (choix 1), de se connecter à une partie déjà lancée (choix 2), de reprendre une partie interrompue (choix 3) ou de quitter (choix 9).

5. Choix 1: lancement d'une nouvelle partie

   Le nom du joueur qui lance la partie, les temps de jeu (temps total de la partie et temps maximum alloué pour un coup) sont saisis.

   L'application attend qu'un autre joueur se connecte à la partie pendant un délai à fixer.

6. Choix 2 : connexion à une partie déjà lancée par l'autre joueur

   Le nom de la partie est saisi ainsi que le nom du joueur.

7. Choix 3 et 4 : Interruption / reprise d'une partie

   L'application offre la possibilité d'interrompre une partie pour la reprendre ultérieurement et d'en gérer l'historique. A cet effet, l'information concernant chaque coup joué est stockée dans un fichier. Le nom de ce fichier est saisi au début de la partie. On lui donnera l'extension .histo

8. Choix 5 et 6 : Mise en pause / reprise d'une partie

   Le décompte des horloges est interrompu.

9. Choix 7 : visualisation de l'historique

   L'historique des coups sauvegardés est affiché. Ce choix interrompt les horloges.

10. Choix / : coup de jeu

    Le contrôle de la validité d'un coup se fera par appel à une fonction jouer(). Celle-ci étant spécifique à chaque jeu, elle ne sera pas implémentée ici, il suffira qu'elle renvoie si le coup est valide, non valide ou si c'est un coup gagnant.

11. Gestion du temps

    L'application doit :

    - afficher le temps restant au joueur pour saisir son coup. L'affichage sera actualisé toutes les secondes,

    - signaler la fin du temps alloué dans le cas où le joueur n'a pas encore effectué sa saisie.

    Le décompte du temps se fera seconde par seconde. On ne le considérera pas comme critique.

12. Fin d'une partie

    La fin d'une partie peut, à votre choix, signifier la terminaison de l'application ou le retour au menu. Dans tous les cas, la partie terminée disparaîtra de la liste des parties et de l'historique ;

    Le nom du joueur gagnant sera affiché.

Implémentation

1. Architecture de l'application

L'application est multiprocessus.

2. Fichier historique

Le fichier historique aura la structure suivante :

1er enregistrement :

octet 1 : 'H'
octet 2 : 'X'
octets 3 à 10 : nom du joueur 1
octet 11 : espace
octets 12 à 19 : nom du joueur 2
octet 20 : espace
octets 21 à 24 : nombre de secondes allouées à la partie
octets 25 à 28 : nombre de secondes allouées à un coup
octet 29 : CR
octet 30 : LF

enregistrements suivants :

octets 1 à 4 : nombre de secondes restantes dans la partie
octet 5 à 12 : nom du joueur
octets suivants : informations sur le coup
2 derniers octets : CR LF

dernier enregistrement

octets 1 à 3 : « FIN »
octet 4 : CR
octet 5 : LF

Remarque : les caractères HX en début de fichier permettent de vérifier qu'il s'agit bien d'un fichier historique.

Un peu d'aide …

1. Exemple de récupération d'une chaîne de caractères contenant la date et l'heure

```
time_t t;
char *date;
t = time(NULL);
date = ctime(&t);
printf("%s\n", date);
```

Résultat obtenu: `Wed Oct 13 09:39:05 2004`

2. Accès au contenu d'un répertoire

Voir la fonction **`readdir()`**