Insititut Universitaire de Lille 1
Ichinoseki kousen

Rapport de stage
Obokata Sensei

Projet: Ichiman Yen

Vandecappelle STEEVE

Ichinoseki, Année 2009 du 30 mars au 28 Juin

Table des matieres

Remerciement

Je souhaite tout d'abord remercier le directeur de l'Etablissement qui m'a si bien acceuillit, — Nom —-, ainsi que son équipe administrative.
En second lieux je souhaiterais remercié les personnes qui m'ont encadrés en ce qui concerne ce projet : Monsieur Obokata, Monsieur Chiba, Monsieur taka.... Ainsi que les élève de la classe 5S particulièrement Henhee, miyuki, et — .
Je souhaiterais enfin remercié, les personnes qui m'ont aidé durant avant et pendant mon voyage au niveau Linguistique et Administratif : Monsieur Rigal, Monsieur Lebegue, Tchai, Pakusan, Azulin, et Moyo.

Ce Stage est donc l'aboutissement de mes deux années d'étude à l'Institut Universitaire de technologie 'A' de lille 1, section DUT Informatique de gestion.J'espere que vous prendrez plaisir à suivre toute les étapes de mon projet. Et remercie donc d'avance le lecteur pour son esprit critique. Vous pourrez ainsi me contacter pour toutes remarques, ou suggestions concernant le rapport ou le projet à proprement parlé.

Résumé

Ce projet éffectué à Ichinoseki avait pour but, de fournir une application complète et facile d'utilisation sur le thème du jeu "Ichiman yens". J'ai donc centré mon travail sur la réalisation d'une interface graphique simple d'utilisation, dans le but éventuel de rendre l'application la plus claire aux yeux des non-programmeurs.Le projet avait deux contraintes : d'une part il devait être écrit en langage C, et d'autre part le lancement du jeu devait s'éxécuter sur le principe de programmes clients / serveur (tous écrit en C).

Nous expliquerons plus en détails le principe, et les règles du jeu, mais pour présenter nous dirons que le jeu oppose deux programmes qui possède chacun une strategie propre. Le programme qui possède la meilleure strategie d'attaque et de défense gagne le plus de manche, à la fin de la partie le nombre de manche gagnés est compté. Celui qui a gagné le plus souvent gagne la partie.
Mon travail à consisté dans un premier temps à réaliser l'interface graphique de l'éxécution du programme Serveur, c'est à dire le programme qui affichait le résultat de la mise en concurence des deux programmes clients. Cette partie à été réalisé uniquement sur l'utilisation du langage C standart et de la librairie "Gtk+-2.0" qui offrait les possibilités graphiques.
Dans un second temps je me suis penché sur la création en elle même des deux programmes concurants. Le but était de fournir une application permettant de créé les deux programmes concurants, et à les exécutés de façon dynamique. Dans ce but j'ai réalisé une interface permettant de faire le lien entre "programmation des programmes concurants" et "Execution du programme serveur". Pour celui-ci j'ai donc utilisé, dans un soucis de rapidité et de lisibilité, les librairies "Glade" et "Gtk+". Glade nous permettant de nous concentré sur le code "utile" du programme en laissant de coté les déclarations fastidueuse dû au grand nombre de composants graphiques. Ce programme integre aussi une section d'édition pour débutant ainsi, les non programmeurs pourront également participés en créant des programmes simples, et eventuellement en les retouchant par eux-mêmes.
L'application ainsi créé, permet d'écrire, compiler, et éxécuter "les programmes combattants" graphiquement.

Abstract

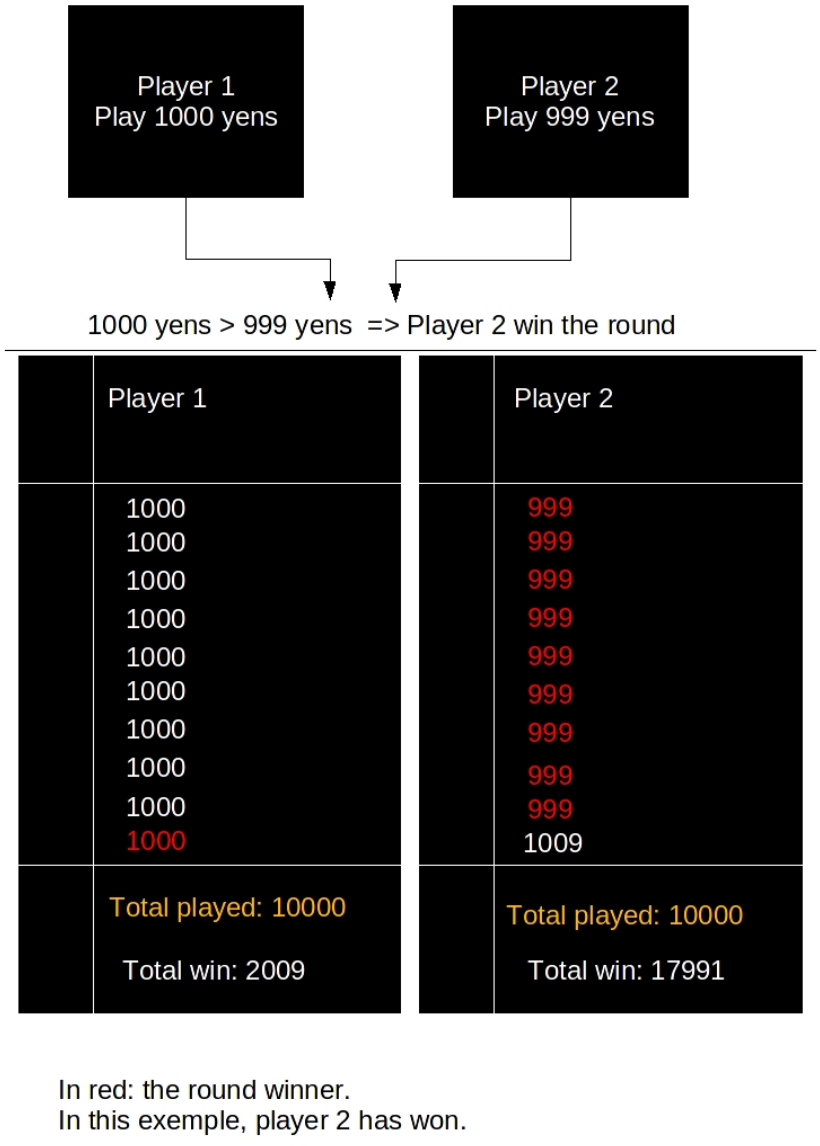
This project, wich I have done at Ichinoseki, had the aim of give a software, the most easier and the most empty it could it be on the "Ichiman yens" game. So, that's why I do my priority of the Interface graphical-mode, for more lisibility and facility for the users.This project had two axes mandatory : first it would be a C program-coded, and second it had to be a client/server application.

In the first hand my work consisted of to make a graphical-mode of the server program. The server display the result of each round of the game. For this part of the project, I'd just used the C standard Library and Gtk2.0 Library to obtains the graficals objects necessary.
On the other hand I wanted to create an other program of reference to do every things the game needs.For create C program, compile them, and execute them. -Devant la longeur du travail - I used the Glade Library -Gagner du temps- and only programing the important code, and don't code all the graphicals declarations, which it would took a long time. This program contains also a help section, for beginners and non-programer.
At the end, the window application was able to create C program easily, compile and execute them on the server's top-level window.

1 Présentation du jeu

1.1 Principe Général

"Ichiman yen" est un jeu strategique, où les deux concurrent ont chacun une somme en leur possession. Cette somme est identique pour les deux participants, et est égale à 10 000 (d'où le nom du jeu 10 000 yen). Le principe du jeu est qu'une manche est constitué de 10 parties. Chacun mise une somme sur la table de jeu, sans savoir ce que l'autre joueur à misé. Ensuite on regarde qui des deux joueurs, à misé le moins. Celui-ci remporte les deux mises placé sur le plateau. La manche ce termine après 10 tours, et la totalité des 10 000 yens de chacun des joueurs doit avoir été misé durant la partie. Voici un petit exemple en image, du déroulement d'une partie :



In red: the round winner.
In this exemple, player 2 has won.

1.2 Programmes Clients

Le programme client fonctionne simplement sur le fait que à chaque tours de jeux, il doit renvoyé une valeure sur la sortie standard : "stdout". C'est au serveur d'attendre la récéption des deux informations, mise du programme 1 et mise du programme 2, avant de lancer le tours de jeux suivant.(Mecanisme que nous développeront plus tard dans la partie consacré au Programme serveur)
Les programmes clients se décompose donc de facon simple, afin de connaitre le nombres de tours de jeux voulus, il sera passé à la chaine d'execution à l'aide des arguments de la ligne de commande. Et c'est le serveur, qui connaissant le nombre de tours à éxécuté qui lancera l'éxécution des deux programmes clients.
Pour mieux comprendre, voici un exemple très simplifié en langage LAP d'un programme client.

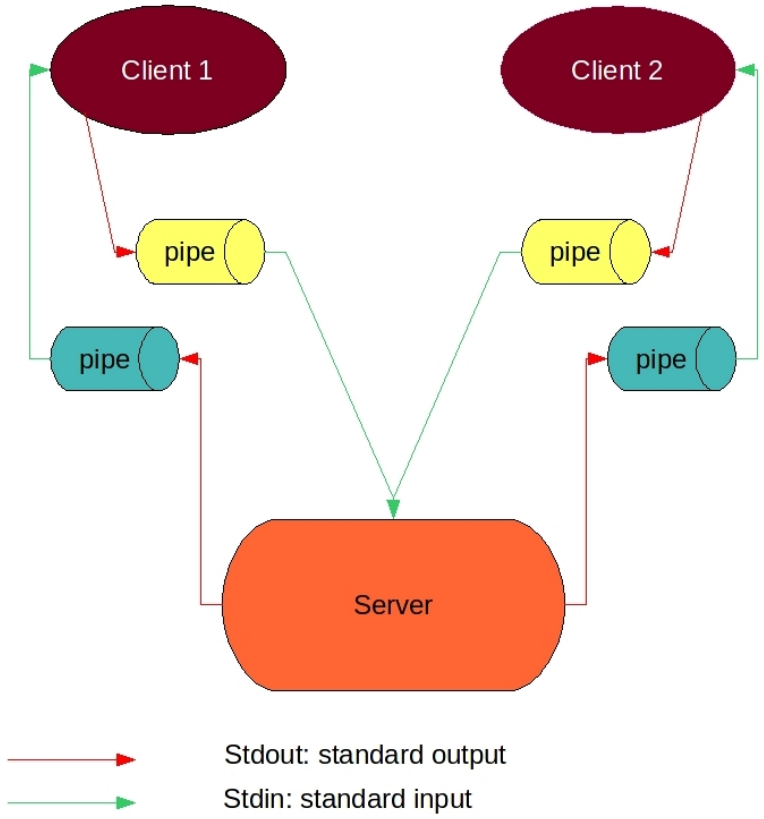
Algorithme 1 Programme Client : Mise toujours 1000 yens

ENTRÉES: entier : nombre_de_Tours
SORTIES: EXIT_SUCCESS

```
integer : i, j ;
Pour i allant de 0 jusqu'à nombre_de_Tours Faire
  Pour j allant de 0 jusqu'à 10 Faire
    /*Affichage sur sortie standard (1000)*/
    Print(1000) ;
    /*Pour forcer l'affichage sur la sortie standard*/
    Flush() ;
    /*Pour attendre la fin du tours en cours ... c'est la réponse du serveur*/
    Put() ;
  Fin pour
Fin pour
Retourner EXIT_SUCCESS ;
```

1.3 Programme Serveur

Le programme Serveur est lui chargé de faire la jonction entre les deux programmes Clients, et affiché le résultat de la confrontation.
Pour ce faire il éxécute deux programmes clients choisi, et les exécutent dans une nouveau processus. Il a donc fallut faire attention aux redirections des fichiers, pour les sorties standard et les entrées standards.Voici un schéma explicatif du fonctionnement de la redirection de fichiers :



2 Programme serveur et interface de développement en gtk+2.0

Le dévelloppement d'une interface graphique en C nécéssite l'utilisation d'une librairie d'objets graphiques. En C il n'en existe qu'une seule, utilisant la technologie GNOME[1]e UNIX.
Pour la création d'interfaces graphiques, l'utilisation d'un langage objet est nécéssaire, afin de profiter des systemes d'heritage et d'interfacages. Or, le langage C n'introduit pas le langage objet, c'est pourquoi Gtk fonctionne d'une manière particulière. Tout objet de la librairie Gtk est un GtkWidget, et l'on appelle les fonctions correspondantes à l'objet que l'on souhaite créer. Gtk fonctionne par l'utilisation exclusive des pointeurs, ainsi aucun type a proprement parlé n'est déclaré, il n'y a que les référence vers ces types qui sont déclarés. Ce sont les fonctions d'initialisation qui se charge de créer le type en mémoire, et de renvoyer l'adresse mémoire en question. Par exemple :

```
GtkWidget *window;
Gtkwidget *label;
window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
label = gtk_label_new("a label");
```

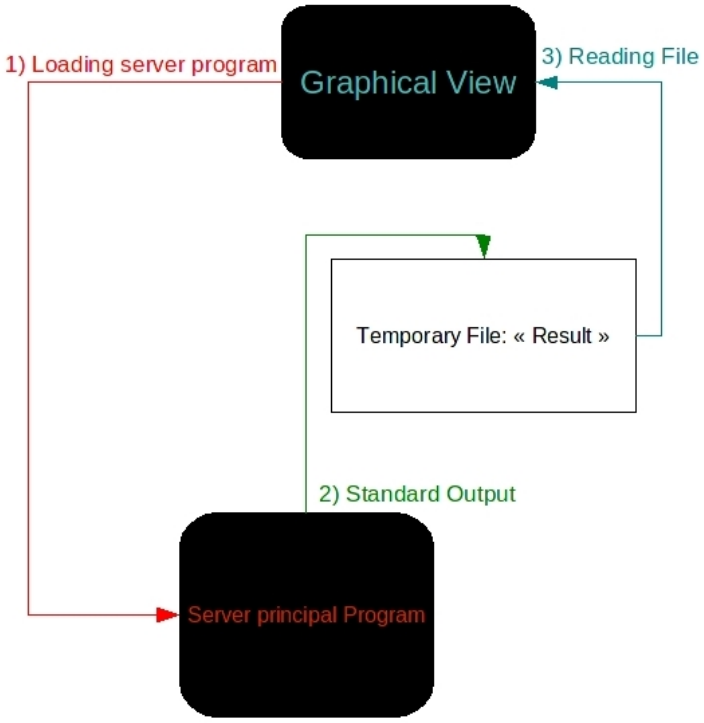
L'autre aspect important de Gtk sont les Macros[2] Lorsque l'objet ne peut pas être définit autrement que par son type particulier, on peut faire appelle à une macro de la librairie Gtk qui se chargera de faire comprendre à la fonction appellée que le type correspond ou non. Par exemple, la fonction gtk_label_set_text permet de modifier le texte d'un label, mais sa signature prend en paramêtre un GtkLabel* ainsi l'on a recourt à une macro pour convertir le type :

```
GtkWidget *label;
label = gtk_label_new("a label");
gtk_label_set_text(GTK_LABEL(label), "a new text label");
```

C'est à l'aide de ces deux principaux aspects que la librairie Gtk permet de créer des interfaces graphiques complexes. Grâce à toute une liste d'éléments graphique et à l'élaboration d'un jeu de calque, nous pouvons placer n'importe quel élément dans une fenêtre.
Gtk est cependant limité par le fait que le langage C n'est pas un langage objet, aussi on ne peut créer de nouveaux types graphiques, et on ne peut pas créer de nouvelles fonctions graphiques. Seul les fonctions et éléments graphiques de la librairie standard Gtk sont accessibles.

2.1 Programme d'affichage serveur

Ce programme est la base de ce projet, il consiste au remplacement de l'affichage du programme fournis par le tuteur.Or je souhaitais que ce programme ne soit qu'une mise à jours possible et donc que l'on puisse encore exploiter le programme en Mode "Terminal". Pour ce faire j'ai donc souhaiter garder le programme d'origine intact. Un probleme c'est donc evidement posé à moi, comment garder le programme intact et en même temps créer une ammélioration, sans pour autant devoir écrire deux fois le code ? Une solution qui n'est peut être pas la plus économique en ressources mémoire, mais qui évitait de devoir créer des spécifications qui aurait entrainé à la retouche du code fournis, m'est venu. Comme dans beaucoup d'applications j'ai donc utilisé un fichier temporaire, dans lequel le résultat de l'éxécution du programme serveur avait été écrit. Voici un petit schéma representant le mode de fonctionnement de l'interface graphique du Serveur.



2.2 Programme de dévloppement

2.2.1 Lien entre programme de dévloppement et affichage serveur : l'Execution

2.2.2 Systeme de sauvegarde

2.2.3 Editeur de texte

2.2.4 Outil de compilation

2.2.5 Onglet d'édition simplifié

3 Cross-compilation et aide dynamique

Une fois le projet global terminé, j'eu l'idée de pousser le projet un peu plus loin qu'une simple application universitaire. Dans cette optique, je me suis inspiré des applications bien connues que l'on pouvait trouver sur nos propres ordinateurs. Qu'est-ce qui fait qu'un programme puisse être une application complete. D'une part tout les projets commerciaux, afin d'être vendeurs, devrait pouvoir être installer sur n'importe qu'elle plateforme d'éxécution. D'où l'idée d'utiliser un outil de cross compilation. D'autre part, les applications les plus complètes bénéficient généralement d'une aide en mode Off-line, pour permettre à l'utilisateur de comprendre au mieux tout les aspects de fonctionnement du programme.D'où cette seconde idée de créer un contexte de d'aide dynamique concernant l'utilisation du programme principal Interface de dévellopement.

3.1 Cross-compilation via CMake

Toujours dans le but de fournir une application exploitable par tous, l'utilisation d'une plateforme de cross-compilation etait nécéssaire, afin de pouvoir éxécuter Ichiman-yen sur tout type de plateforme, Unix, Windows et Mac par exemple.

3.2 Aide dynamique

Annexes

Le programme Serveur, fournis par le tuteur Mr Obokata.

Algorithme 2 Programme Serveur

```
ENTRÉES: chemin_joueur_1 & chemin_joueur_2 & nombre_de_tours
  Si probleme d'arguments Alors
    Quitter programme EXIT_FAILLURE
  Fin Si
  /* Creer les pipes, et redirections des entrées sorties de programmes. 4 pipes à créer*/
  Si processus fils 1 Alors
     /*on ferme les entrée standards des programmes clients*/
     close(entrée_pipe1) ;
     close(entrée_pipe2) ;
     /*on redirige les sorties standards vers les programmes clients et depuis le programme serveur*/
     dup2(sortie_programme_serveur,0) ;
     dup2(_sortie_client1,1) ;
     execl(argv[2],argv[2],argv[3],NULL) ;
  Fin Si
  Si processus fils 2 Alors
    close(entrée_pipe2) ;
     close(entrée_pipe1) ;
     dup2(sortie_programme_serveur,0) ;
     dup2(_sortie_client2,1) ;
     execl(argv[2],argv[2],argv[3],NULL) ;
  Fin Si
  Pour j Allant de 1 à nb_de_round Faire
     Pour i Allant de 1 à 10 Faire
       Lire_les_octets_dans_le_pipe_1 ;
       Si Lecture pipe 1 se déroule mal Alors
          EXIT_ALL_PROGRAMS
       Fin Si
        Lire_les_octets_dans_le_pipe_2 ;
       Si Lecture pipe 2 se déroule mal Alors
          EXIT_ALL_PROGRAMS
       Fin Si
      ecrire_dans_pipe_1 ;
      ecrire_dans_pipe_2 ;
    Fin pour
  Fin pour
```

Glossaire

Pipe : De l'anglais signifiant "Tube" c'est un moyen de communication entre deux processus. Il fonctionne comme son nom l'indique à la manière d'un tube. On envoi l'information à travers le tube, et l'autre processus lit l'information.Lors de sa création on redirige donc les entrées et sorties des programmes afin de récupéré ou ecrire systématiquement dans le tube.

Liens / Documentation Externe

Liens d'aide en Japonais :

```
http://www.dictionnaire-japonais.com/
```

Liens à propos de la librairie Gtk

```
http://www.gtk-fr.org/
http://www.gtk.org/
```

Liens à propos de CMake et de la cross-compilation

```
http://live.gnome.org/gtkmm/UsingCMake
http://www.cmake.org/
```

Et bien sur les "man pages" des librairies standards C