

Projet Unix Bataille Navale

Systeme et Réseaux Unix

Année 2002-2003

Projet UNIX de fin d'année BATAILLE NAVALE en Mode Client Serveur

Dans ce rapport nous allons présenter les bases du développement du projet de fin d'année en cours Unix. Ce projet consiste à développer un jeu bien connu du grand public : LA BATAILLE NAVALE. Ce jeu sera développé en mode Client-Serveur, c'est à dire qu'une machine sous Unix lancera une partie (lancée par l'administrateur du Jeu) et une (ou plusieurs) autre(s) machine(s) viendra(ont) se connecter sur cette partie afin de jouer.

Dans ce but nous gérerons un serveur et 2 clients (2 joueurs ou plus).
Le schéma suivant montre le principe de ce projet.

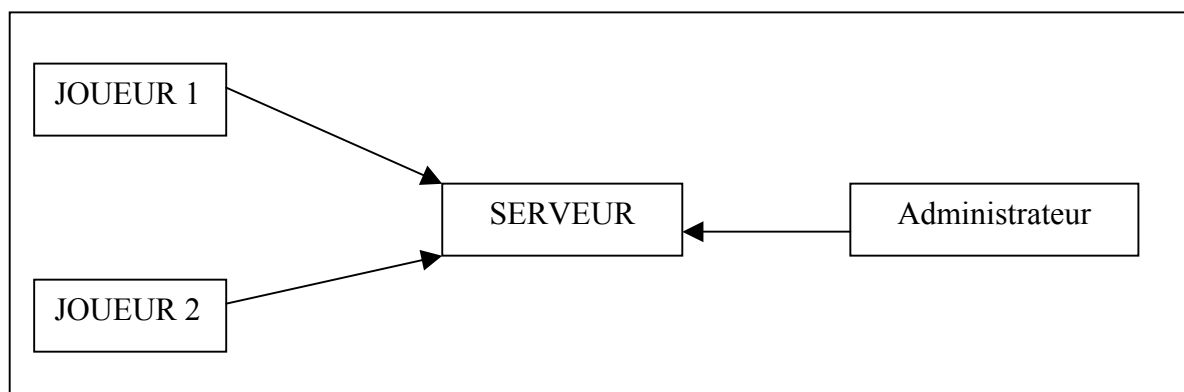


Schéma de Principe de l'objectif du Projet

Nous allons maintenant définir la liste des messages qui circulera dans l'ensemble du programme (client et serveur), afin de pouvoir développer plus rapidement à la suite.

Protocole d'échange à travers les Sockets

Les messages seront séparé suivant l'origine et la destination de celui-ci. Un numéro lui sera associé selon cette règle :

- **Les messages avec un N° pair** : Du client vers le serveur
- **Les messages avec un N° impair** : Du serveur vers le client

Chaque message comportera donc pour cette définition un identifiant sur 2 caractères (2 octets) (Nommé « Num Msg » dans le tableau).

La gestion de la réception d'un message se fera par traitement en deux étapes :

- 1) Récupération du Numéro du message (sur 2 caractères)
- 2) Récupération du reste des données en fonction de l'identifiant de celui-ci.

On va masquer ce système de reconnaissance directement par la couche de communication lors de l'utilisation de l'application.

Num Msg	Contenu du Message	Type	Taille (octet)	Description du Message
0	Login Mot de Passe	Char Char	10 10	Demande d'identification (le client attend ensuite un msg 1)
1	Status	Char	1	Réponse à l'identification du client par le serveur Status : 0 = OK 1 = refus (user inconnu) 2 = already logged (Déjà connecté) 3 = pb technique (tout autre problème)
2				Inscription à une partie Le client attend ensuite un msg 3
3	Status	Char	1	Réponse à la demande d'inscription à une partie Status : 0 = inscription acceptée (le client se met en attente d'un msg de début de partie msg 5) 1 = inscription refusée, car trop de user (paramètre nb max user sur serveur) 2 = inscription refusée par admin 3 = Game full (nombre max de joueur pour la partie atteinte, le client se met en attente d'un autre msg 3) 4 = pas de partie créée par le serveur 5 = pas d'admin connecté sur le serveur 6 = pb technique (autre)
5	Nb de joueurs	Int	1	Début de la partie Ce message permet d'initialiser l'affichage sur tous les clients (Liste des joueurs et carte vide) Après avoir lu le nombre de joueur, chaque client récupère autant de msg 15 qu'il y a de joueurs puis, ils se mettent en attente d'un msg de tour de jeu (7) ou d'un résultat de tour de jeu (9 ou 11) ou d'une fin de partie(13)
15	Login Score	Char int	10 1	Nom et score d'un joueur
7				Le serveur demande à un client de jouer

8	Position X Position Y	Int Int	1 1	Suite à un msg 7 , le client indique au serveur la case X, Y jouée puis il se met en attente d'un msg 9 ou 11.
9				Réponse coup dans l'eau
11	Valeur case Login Score Nb positions	Char Char Int Int	1 10 1 1	Le serveur publie le résultat d'un tour de jeu à tous les clients si le joueur a touché quelque chose. Valeur des cases (T ou C), login et score du joueur ayant touché et le nombre de cases concernées par ce changement. Les clients attendent ensuite autant de msg 17 qu'il y a de cases concernées.
17	Position X Position Y	Int Int	1 1	Position X,Y d'une case modifiée (suite du msg 11)
13				Fin de partie diffusée à tous les participants. Un msg 31 est ensuite envoyée pour affichée la map complète
14				Déconnexion d'un client
20	Login Password	Char Char	10 10	<u>Msg uniquement admin</u> Ajout d'un user dans la liste
21	Status	Char	1	Réponse à la création d'un user Status : 0 = OK 1 = déjà enregistré 2 = pb technique
22	Login	Char	10	<u>Message concernant uniquement l'administrateur</u> Suppression d'un user
23	Status	Char	1	Réponse à la suppression d'un user Status : 0 = OK 1 = user inconnu 2 = pb technique

24	Nb joueurs Nb bateaux	Int Int	1 1	<u>Message concernant uniquement l'administrateur</u> Création d'une partie Le nombre de joueurs maximum pour la partie et le nombre de bateaux. Il attend en réponse un msg 29. Puis, l'admin envoie ensuite autant de msg 26 qu'il y a de bateaux. La réponse est un msg 31 à chaque positionnement d'un bateaux.
29	Status	Char	1	Résultat de la demande de création d'une partie Status : 0 = OK, attente de la position des bateaux 1 = partie créée ou en cours 2 = nb joueurs pour la partie trop important 4 = nb bateaux trop importants
26	Position X Position Y Taille Direction	Int Int Int Char	1 1 1 2	Position de départ d'un bateau (X,Y), taille du bateau, nombre de cases et directions : N-S-E-O L'admin attend une réponse (msg 25)
25	Status	Char	1	Résultat du placement d'un bateau Status : 0 = OK 1 = hors limite 2 = chevauchement
27				<u>Message concernant uniquement l'administrateur</u> Inscription d'un joueur à la partie (l'admin doit accepter ou refuser l'inscription par msg 28)
28	Status	Char	1	<u>Message concernant uniquement l'administrateur</u> Acceptation d'un joueur Status : 0 = joueur accepté 1 = joueur refusé pour la partie
30				<u>Message concernant uniquement l'administrateur</u> Début de partie
31	Map	Char	Taille map	Permet d'avoir le plan global avec tous les bateaux et leurs états

Remarque sur le Tableau :

Pb Technique :

Exemple : file system full, shared memory, sémaphore non présente, etc...

1. Programme serveur

Le programme de serveur de jeu est de type multi-processus

Nous avons paramétré 1 processus sur une socket de RDV.

Il y a 1 processus par client donc N processus pour N clients.

Les données du jeu sont réparties dans plusieurs shared memory :

- map du jeu (définition de la grille de jeu),
- liste des bateaux (positions, état),
- liste des login/password (de l'administrateur et de clients),
- la liste des clients connectés :
login et pid de chaque processus de gestion de ces clients. Le processus maître, qui est celui de l'administrateur peut ainsi dialoguer avec chacun des processus des clients voulant se connecter ou jouer,
- la liste des joueurs sélectionnés pour la partie,
- une zone d'échange pour le dialogue entre les processus des joueurs et le processus d'admin (login, type de la demande).

Liste des sémaphores utilisées et leur fonction :

- un pour chaque processus des joueurs qui permet de gérer le tour de jeu entre chaque joueur,
- un sémaphore d'administration pour initier le dialogue avec l'admin,
- un sémaphore de verrouillage pour bloquer l'accès aux shared memory des données.

2. Programme client

Le programme client est de type mono-processus, il possède les paramètres suivant :

login,
password,
nom de la machine du serveur de bataille navale
le numéro de port (en pton).

Pendant toute opération et le déroulement du jeu, un CTRL-C permet d'interrompre une attente.

Algorithme principal :

Positionnement du handler de CTRL-C

Demande identification (msg 0)

Si réponse différente de OK : affichage message puis arrêt

Si login = admin

Mode Admin

Sinon

Mode Joueur

En mode Admin, le menu suivant est proposé :

c – création d’une partie

i – demande d’informations

n – déclaration d’un nouvel utilisateur

d – suppression d’un utilisateur

x - déconnection

La demande d’information est à détailler (liste des connectés, map avec bateaux,...), le détail sera effectué ultérieurement et pourra entraîner l’ajout de message.

En mode Joueur, les choix sont différents :

I – inscription à une partie

X - déconnection

Nous nous réservons la possibilité d’ajouter des choix lorsque l’analyse et les développements seront plus avancés.

3. Descriptif du jeu

Le lancement du serveur de jeu se fait par l'exécution de -SVRJEU n°socket-
Le lancement des clients s'exécute par la commande -CLIENTJEU localhost n°socket

Principe de connexion

Au lancement du serveur, le serveur effectue un fork afin de dialoguer avec le ou les clients.

Les clients se log sur le serveur à l'aide d'un login et d'un password (répertoriés dans listeusers.cfg modifiable celui-ci gère l'ensemble des noms des joueurs). Le serveur prend ainsi contact avec le client sur une socket de rendez vous. Après vérification du login, le client se connecte et le serveur crée un process avec une autre socket, une socket de connexion.

Le socket de rendez vous est le même pour toutes les demandes de connexion sur le serveur alors que le socket de connexion ainsi que le process engendré est unique pour chaque client.

Si le login et le password n'apparaît pas dans la liste, la connexion est interrompue dans le cas contraire, apparaît un menu (différent si l'on est logué en admin ou en joueur simple).

1^{er} cas : Log en admin.

C'est un user qui crée et gère la partie à l'aide du menu. Les différents choix que disposent l'admin permet de créer la map, les bateaux et le nombre de joueur. Un menu caché : Z permet d'afficher sur une console du serveur de jeu, l'ensemble des informations concernant la partie, c'est à dire la récapitulation sur les différents éléments du jeu mais également sur l'évolution des points affecté à chaque joueur. C'est aussi sur cette console que l'on pourra visualiser le gagnant de la partie.

2^{eme} cas : Log en user (joueur).

Le joueur doit attendre que l'administrateur lance le jeu après sa configuration ensuite, le joueur a la possibilité de lancer sa demande de jeu.

L'ordre de jeu des joueurs est défini par leur ordre d'arrivée sur le plateau.

L'ensemble des menus est géré par la partie IHM comme demandé dans le cahier des charges.

La configuration du jeu gérée par l'administrateur est paramétrable dans le fichier -gestpart.h- .

Le fichier -jeu.c- permet la gestion des bateaux sur la carte, il vérifie leur position et leur non superposition ou leur probable débordement de l'air de jeu.

4. Les fichiers du jeu

L'ensemble du jeu de la bataille demandé comprends 18 fichiers se reposant sur le cahier des charges. Il faut noter que le fichier protoclient.c demandé a été décomposé en 2 fichiers.

4.1 Répartition des fichiers

SERVEUR	CLIENT
gestpart.c gestpart.h comserv.c connectjoueurs.c connectjoueurs.h joueurs.c joueurs.h partie.c partie.h servjeu.c socket.c socket.h	comclient.c ihm.c clientjeu.c socket.c socket.h

4.2 Description des fichiers

Clientjeu.c	Initialisation du dialogue socket coté client
Servjeu.c	Corps serveur socket avec fork pour chaque connexion
Socket.c Socket.h	Créer une socket du domaine INIT
Comclient.c	Fonctions d'émission ou réception du client pour le serveur
Comserv.c	Fonctions d'émission ou réception du client pour le serveur
Connectjoueurs.c Connectjoueurs.h	Fonctions pour gérer les listes : Listes des connectés Listes des joueurs
Gestpart.c Gestpart.h	Gestion de la partie : Gestion des share memory Gestion des listes Description de tous les paramètres systèmes
Ihm.c	Menu des clients Saisie des données
Partie.c Partie.h	Corps du programme de la bataille navale

Listejoueurs.cfg	Liste des utilisateur du système
Protoserv.h	Définition des identifiant des messages échangés Définitions des structures échangées
joueurs.c joueurs.h	Fonction de gestion de la liste des users / fichiers

4.3 Les programmes

Voir annexes.

5. Répartition du travail

Fabrice ROMELARD a traité la partie serveur avec le traitement du dialogue Client/Serveur. Il a également traité la construction des sockets de rendez vous et de connexion entre tous les joueurs.

Nicolas RICHETON a traité les parties client et la communication entre poste (comclient.c, comserv.c...) ainsi que la fonction de gestion des connexion clients. Il a également traité la partie ihm du jeu.

Le corps principal du jeu (jeu., jeu.h) a été traité par les 2 membres du projet ainsi que la rédaction de ce présent rapport.