HE-IG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Signaux et sockets réseau (IPC)

lab08 (Semaine du 13.11.2023)

Objectifs de laboratoire

Ce laboratoire permet d'exercer l'utilisation des *sockets* permettant la communication via le réseau. Il exerce aussi l'utilisation de signaux et de leur handlers associés.

Récupération et rendu du laboratoire

Pour récupérer la branche de ce laboratoire, utilisez les commandes:

1. "*git fetch upstream*"
2. "*git checkout labXX"*

Ce laboratoire sera à rendre selon la méthode vue lors du précédent laboratoire, à savoir

1. Ajouter les fichiers modifiés avec "*git add <vos_fichiers>"*
2. Commiter les changements avec "*git commit –m "<message de commit>"*
3. Envoyer les changements sur votre repo gitlab avec "*git push –u origin"* ou "*git push —set-upstream origin"*

Etape 1 – Bataille navale

L'objectif de ce laboratoire sera de créer un programme permettant de jouer une partie de bataille navale simplifiée via deux terminaux liés par sockets. Complétez pour cela le fichier "**usr/host/battleship.c**"

La partie se déroule sur une grille de 4x4 case numérotées de 0 à F en hexadécimal :

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

Chaque joueur dispose de 3 bateaux occupant une seule case. Au début de la partie, le programme doit demander au joueur 3 emplacements pour placer ses bateaux. Le joueur ne peut pas placer plusieurs bateaux sur une même case

Une fois les bateaux placés, le programme "serveur" commence:

1. Le "serveur" demande une case à attaquer au joueur et envoie le numéro de la case au "client"
2. Le "client" met à jour l'état de la grille de son joueur (touché ou raté), informe son joueur de la case attaquée et de son nouvel état, affiche la grille de son joueur puis notifie le "serveur" du résultat de son attaque.

3. Le "serveur" indique le résultat à son joueur et affiche la grille adverse.
4. C'est ensuite au tour du "client" d'effectuer les mêmes opérations.

La partie se déroule ainsi jusqu'à ce que les 3 bateaux d'un joueur aient été coulés. Les entrées des joueurs doivent être assainies afin d'interdire les valeurs plus petites que 0, plus grandes que 15 ou ayant déjà été saisies

Les règles ci-dessus doivent être respectées. Vous êtes libres de décider:

- La représentation de la grille et des cases en fonction de leur état (vide, occupée, manqué, coulé)
- Comment représenter les valeurs et informations transmises entre le programme serveur et client (On ne demande pas que votre programme soit compatible avec celui d'un autre étudiant mais il doit être compatible avec lui-même). Cela dit, il est judicieux de faire au plus simple

Afin de simplifier singulièrement la validation de vos rendus par vos estimés assistants, merci de respecter l'utilisation stricte des suggestions de texte situées dans le code (+0.2 points à la note de ce labo pour le premier étudiant de chaque classe qui enverra le nom de cette figure de style à Anthony I. Jaccard). Vous pouvez les compléter à l'aide de *sprintf* mais ne modifiez pas les chaines de base se trouvant dans le tableau au début du programme.

Exemple de partie:
Serveur

```
~/sye23_student/usr/host$ ./battleship.elf -s
Please enter 3 grid cells to place your boats into
0
G
Invalid ! Please enter a grid cell index between 0-F
-1
Invalid ! Please enter a grid cell index between 0-F
A
A
Invalid ! You cannot place 2 boats on the same grid cell
B
User grid
0 ~ ~ ~
~ ~ ~ ~
~ ~ 0 0
~ ~ ~ ~
Server listening on port 5000...
Client connected
Please enter a grid cell to attack your opponent
A
You attacked A: MISS
Opponent grid
~ ~ ~ ~
~ ~ ~ ~
~ ~ o ~
~ ~ ~ ~
Waiting for opponent...
Your opponent attacked 0: HIT
```

```
User grid
X ~ ~ ~
~ ~ ~ ~
~ ~ 0 0
~ ~ ~ ~
Please enter a grid cell to attack your opponent
A
Invalid ! You already attacked this cell
B
You attacked B: HIT
Opponent grid
~ ~ ~ ~
~ ~ ~ ~
~ ~ o X
~ ~ ~ ~
Waiting for opponent...
Your opponent attacked A: HIT
User grid
X ~ ~ ~
~ ~ ~ ~
~ ~ X 0
~ ~ ~ ~
Please enter a grid cell to attack your opponent
3
You attacked 3: MISS
Opponent grid
~ ~ ~ o
~ ~ ~ ~
~ ~ o X
~ ~ ~ ~
Waiting for opponent...
Your opponent attacked B: HIT
User grid
X ~ ~ ~
~ ~ ~ ~
~ ~ X X
~ ~ ~ ~
YOU LOST T_T
```

Client

```
~/sye23_student/usr/host$ ./battleship.elf
Please enter 3 grid cells to place your boats into
0
2
B
User grid
0 ~ 0 ~
~ ~ ~ ~
~ ~ ~ 0
```

```
~ ~ ~ ~
Connecting to server 127.0.0.1:5000...
Connected to server successfully
Waiting for opponent...
Your opponent attacked A: MISS
User grid
0 ~ 0 ~
~ ~ ~ ~
~ ~ o 0
~ ~ ~ ~
Please enter a grid cell to attack your opponent
0
You attacked 0: HIT
Opponent grid
X ~ ~ ~
~ ~ ~ ~
~ ~ ~ ~
~ ~ ~ ~
Waiting for opponent...
Your opponent attacked B: HIT
User grid
0 ~ 0 ~
~ ~ ~ ~
~ ~ o X
~ ~ ~ ~
Please enter a grid cell to attack your opponent
A
You attacked A: HIT
Opponent grid
X ~ ~ ~
~ ~ ~ ~
~ ~ X ~
~ ~ ~ ~
Waiting for opponent...
Your opponent attacked 3: MISS
User grid
0 ~ 0 o
~ ~ ~ ~
~ ~ o X
~ ~ ~ ~
Please enter a grid cell to attack your opponent
B
You attacked B: HIT
Opponent grid
X ~ ~ ~
~ ~ ~ ~
~ ~ X X
~ ~ ~ ~
YOU WON ^_^
```

Etape 2 – Bataille navale - Abandon

A tout moment, un joueur doit pouvoir abandonner la partie en appuyant sur Ctrl+Z (Envoi du signal SIGSTP au processus en cours d'exécution). Remplacez le handler par défaut de ce signal par un handler de votre conception qui permettra d'envoyer un message au programme adverse signalant l'abandon de la partie avant de quitter le programme

Serveur

```
...
Waiting for opponent...
<hit Ctrl + Z>
^ZYou forfeited the game: YOU LOST T_T
```

Client

```
...
Please enter a grid cell to attack your opponent
1
Your opponent forfeited the game: YOU WON ^_^
```