

Introdução

O trabalho prático em questão se diferencia do anterior por se tratar de um servidor multithread, tendo que lidar com até 15 clientes simultaneamente. Para lidar com esse desafio inicial, a última videoaula do professor Ítalo Cunha foi a referência utilizada.

Dificuldades na Sincronização

A arquitetura do projeto é baseada em uma thread principal no servidor para o loop do jogo e uma thread dedicada para cada cliente conectado. O desafio mais árduo foi fazer essas threads trabalharem em conjunto da forma como era esperado.

Entre os primeiros problemas, uma questão crítica foi um deadlock que congelou o servidor. A thread do jogo bloqueava um mutex e posteriormente chamava a função de broadcast que tentava bloquear o mesmo mutex, o que gerava erro já que uma thread não pode adquirir um lock que já possui. A solução se resumiu em refazer a lógica de bloqueio, passando a liberar o mutex antes de chamar funções que tinham o próprio escopo. Posteriormente, na versão final do código, a função de broadcast passou a não ter controle sobre o mutex, passando a ser responsabilidade apenas da função principal.

```
28
29 void broadcast_message(struct aviator_msg *msg) {
30     for (int i = 0; i < MAX_CLIENTS; i++) {
31         if (clients[i]) {
32             send(clients[i]->csock, msg, sizeof(struct aviator_msg), 0);
33         }
34     }
35 }
36
```

Além do deadlock, inicialmente a parte de apostas era realizada apenas por um sleep na thread do

jogo. Esse acontecimento fazia com que o estado do jogo mudasse para IN_FLIGHT antes que as threads dos clientes tivessem tempo de processar as mensagens do tipo bet que chegavam pelo terminal. A partir disso, as apostas eram ignoradas e o sleep teve que ser substituído por um loop de usleep que durava 10 segundos, mantendo o estado em BETTING.

```
65 broadcast_message(&start_msg);
66 pthread_mutex_unlock(&state_mutex);
67
68 for (int i = 0; i < 100; i++) {
69     usleep(100000);
70 }
71
72 usleep(100000);
73
74 pthread_mutex_lock(&state_mutex);
75
```

Por fim, a configuração inicial de bloqueios de mutex fazia com que ele ficasse bloqueado durante todo o voo do avião. O acontecido prejudicava o funcionamento do cashout, já que nenhum cliente adquiria o lock para processar qualquer pedido de cashout. A solução foi fazer a thread game bloquear o mutex, enviar a atualização do multiplicador e liberar imediatamente a cada loop. Após essa alteração, as threads dos clientes conseguiram processar pedidos dessa ordem.

Dificuldades no Servidor

Por se tratar de um servidor multithread, foi um grande desafio fazer o mesmo continuar funcionando corretamente mesmo com todos os comportamentos inesperados que os clientes poderiam realizar.

Entre as dificuldades de estabelecer o servidor, em um dos testes foi observado que ele caía quando um cliente se desconectava. A

solução do problema foi fazer o sistema ignorar sinais SIGPIPE, que encerravam o processo, bem no início da main. Sinais desse tipo estavam sendo gerados quando existia uma chamada send para um socket fechado por um cliente. Mesmo assim, o servidor ainda não consegue lidar com alguns cenários de saída de clientes.

Problemas na Interface

Existiram alguns problemas na comunicação entre cliente e servidor através do terminal, o que era esperado em um código com muitos logs específicos para cada situação. Entre eles, o mais problemático se tratou do bug no nickname do cliente. Inicialmente, ele estava sendo enviado no campo value do aviator_msg, causando um erro em que o nome aparecia com nick no final, como por exemplo “Flipnick”. O erro foi resolvido simplesmente alterando malloc por calloc, limpando assim a memória.

```
struct client_data *cdata = calloc(1, sizeof(*cdata));
if (!cdata) {
    logexit("malloc");
}
cdata->sock = csock;
memcpy(&(cdata->storage), &cstorage, sizeof(cstorage));
```

Houveram falhas na implementação do cronômetro de contagem regressiva para os 10 segundos pelo terminal, que deixava a tela cheia de logs repetidos com o texto do mesmo se sobrepondo ao input do cliente. O ocorrido se tratava de uma tentativa da função de cronômetro, que agora não existe mais, e da thread principal de escrever no terminal ao mesmo tempo. Isso foi resolvido desabilitando essa função e infelizmente agora só é possível

visualizar a mensagem de 10 segundos restantes. Mesmo assim, quando um novo cliente chega e a aposta ainda está ativa, os segundos que restam são mostrados corretamente.

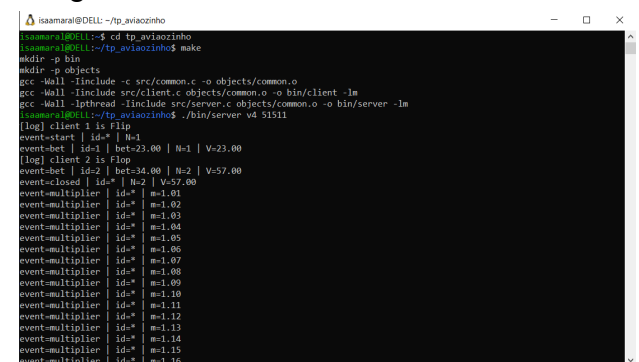
Melhorias

Considerando tudo o que foi feito do trabalho até aqui, algumas resoluções de problemas obrigatórios não foram atingidos. Entre os problemas menores, a implementação de um cronômetro que se atualiza constantemente não foi possível. Além dessa melhoria, o código ainda não lida muito bem com condições em que o cliente se retira do jogo abruptamente. Quando a tecla Q é pressionada durante um voo, por um exemplo, o servidor não é capaz de voltar para um estado normal de funcionamento mesmo com as tentativas de resolução implementadas.

Portanto, o código não implementa todos os controles de erro requeridos pelo trabalho para todos os cenários possíveis, mas possui resultados minimamente satisfatórios para uma primeira comunicação multithread.

Imagens do Terminal

Imagem do terminal do servidor:



```
isaacmora@DELL: ~/tp_aviao2020
isaacmora@DELL:~/tp_aviao2020$ cd tp_aviao2020
isaacmora@DELL:~/tp_aviao2020$ make
mkdir -p bin
mkdir -p objects
gcc -Wall -Iinclude -c src/common.c -o objects/common.o
gcc -Wall -Iinclude src/client.c objects/common.o -o bin/client -lm
gcc -Wall -pthread -Iinclude src/server.c objects/common.o -o bin/server -lm
isaacmora@DELL:~/tp_aviao2020$ ./bin/server v4 51511
[log] client 1 is Flip
event-start | id=1 | N=1
event-bet | id=1 | bet=23.00 | N=1 | V=23.00
[log] client 2 is Flip
event-bet | id=2 | bet=34.00 | N=2 | V=57.00
event-closed | id=1 | N=2 | V=57.00
event-multiplier | id=1 | m=1.01
event-multiplier | id=1 | m=1.02
event-multiplier | id=1 | m=1.03
event-multiplier | id=1 | m=1.04
event-multiplier | id=1 | m=1.05
event-multiplier | id=1 | m=1.06
event-multiplier | id=1 | m=1.07
event-multiplier | id=1 | m=1.08
event-multiplier | id=1 | m=1.09
event-multiplier | id=1 | m=1.10
event-multiplier | id=1 | m=1.11
event-multiplier | id=1 | m=1.12
event-multiplier | id=1 | m=1.13
event-multiplier | id=1 | m=1.14
event-multiplier | id=1 | m=1.15
event-multiplier | id=1 | m=1.16
```

Imagem do terminal do primeiro cliente:

```
isaamaral@DELL: ~/tp_aviaozinho
isaamaral@DELL:~$ cd tp_aviaozinho
isaamaral@DELL:~/tp_aviaozinho$ make
mkdir -p bin
mkdir -p objects
gcc -Wall -Iinclude -c src/common.c -o objects/common.o
gcc -Wall -Iinclude src/client.c objects/common.o -o bin/client -lm
gcc -Wall -lpthread -Iinclude src/server.c objects/common.o -o bin/server -lm
isaamaral@DELL:~/tp_aviaozinho$ ./bin/client 127.0.0.1 51511 -nick Flip

Rodada aberta! Digite o valor da aposta ou [Q] para sair (10 segundos restantes):
23
Aposta recebida: R$ 23.00

Apostas encerradas! Nao e mais possivel apostar nesta rodada.
Digite [C] para sacar.
Multiplicador atual: 1.89x
Aviãozinho explodiu em: 1.89x
Você perdeu R$ 23.00. Tente novamente na próxima rodada! Aviãozinho tá pagando :)
Profit atual: R$ -23.00
Profit da casa: R$ 9.40

Rodada aberta! Digite o valor da aposta ou [Q] para sair (10 segundos restantes):

Apostas encerradas! Nao e mais possivel apostar nesta rodada.

Aviãozinho explodiu em: 1.00x
Profit da casa: R$ 9.40
```

Imagem do terminal do segundo cliente:

```
isaamaral@DELL: ~/tp_aviaozinho
isaamaral@DELL:~$ cd tp_aviaozinho
isaamaral@DELL:~/tp_aviaozinho$ make
mkdir -p bin
mkdir -p objects
gcc -Wall -Iinclude -c src/common.c -o objects/common.o
gcc -Wall -Iinclude src/client.c objects/common.o -o bin/client -lm
gcc -Wall -lpthread -Iinclude src/server.c objects/common.o -o bin/server -lm
isaamaral@DELL:~/tp_aviaozinho$ ./bin/client 127.0.0.1 51511 -nick Flop

Rodada aberta! Digite o valor da aposta ou [Q] para sair (4 segundos restantes):
34
Aposta recebida: R$ 34.00

Apostas encerradas! Nao e mais possivel apostar nesta rodada.
Digite [C] para sacar.
Multiplicador atual: 1.40xc

Você sacou em 1.40x e ganhou R$ 47.60!
Profit atual: R$ 13.60
Multiplicador atual: 1.89x
Aviãozinho explodiu em: 1.89x
Profit da casa: R$ 9.40

Rodada aberta! Digite o valor da aposta ou [Q] para sair (10 segundos restantes)

Apostas encerradas! Nao e mais possivel apostar nesta rodada.

Aviãozinho explodiu em: 1.00x
Profit da casa: R$ 9.40
```

Código do projeto no GitHub:

https://github.com/Isaamaral/TP2_Aviaozinho