Relatório Trabalho 1

Emanuelle Foscarini (19200415) e Ricardo Jensen Eyng (19203166)

Florianópolis - 22/08/2021

Neste trabalho, optamos por fazer a implementação da seguinte forma:

1. Inicialização chieftain

Implementação da função **chieftain_init,** colocamos o nosso array de *chairlist*, assim como os semáforos e mutexes necessários para o controle das threads. O contador em questão refere-se ao número de vikings que saíram da mesa (terminaram de comer), utilizado para saber quando todos terminaram e estão prontos para rezar.

O *for* em questão é onde inicialmente, todas as cadeiras são lugares vazios com um prato, representado pelo 3. Cadeiras vazias sem prato podem ser elementos com o número 4 ou 5 (dependendo em qual sentido seu pratos foram pegos pelos vikings).

2. Banquete

Para implementar o banquete, optamos por criar os seguintes parâmetros na função **chieftain_acquire_seat_plates(...)**:

- vikingChair, o -1 fará o programa parar de executar caso a variável mantenha esse valor após passar pela lógica aplicada na sequência, o que não irá ocorrer;
- vikingType, 2 para berserker e 1 para um viking comum;
- otherVikingType, para representar o tipo do viking que não é o mesmo que o meu: se eu for berserker, o outro tipo é 1, caso eu não seja, então o outro tipo é berserker (2);
- table_max_index, tamanho da minha mesa, menos 1.

A lógica da disposição de pratos e cadeiras é a seguinte:

Setando a primeira posição como vazia e contendo um prato, feito isso então posso me sentar na primeira posição e pegar mais um prato na segunda posição. A segunda posição (da qual eu peguei meu segundo prato) é colocada como uma cadeira vazia sem prato, com um número para indicar quem pegou o prato dela (sendo o sentido horário = 5).

Caso a próxima cadeira não esteja vazia, mas com um prato, a última cadeira (separada de mim pelo vão) deve estar vazia e com um prato. E a próxima cadeira (depois de mim) deve ter um viking do meu tipo, ou estar vazia (sabidamente sem prato). A primeira posição fica ocupada "comigo" a última posição é colocada como uma cadeira vazia sem prato [com um número para indicar quem pegou o prato dela (sentido anti-horário = 4)].

Por outro lado, caso a primeira cadeira não esteja vazia E COM UM PRATO ou não ser possível respeitar alguma das outras condições para que eu pudesse sentar nela (ter um outro prato por perto, não ter vikings de outro tipo na segunda posição), é feito da seguinte forma: é executado para cada cadeira na mesa, tirando a primeira e a última (as quais são casos especiais), a cadeira depois da que eu estou tentando sentar está vazia e com um prato e a cadeira antes da que eu estou tentando sentar ou está vazia (podendo ou não conter um prato) ou está ocupada por algum viking do meu tipo. Porém, caso a cadeira i+1 não esteja vazia E COM UM PRATO, a cadeira ANTES da que eu estou tentando sentar está vazia e com um prato e a cadeira DEPOIS da que eu estou tentando sentar ou está vazia (sabidamente sem prato) ou está ocupada por algum viking do meu tipo.

Por fim, caso não tenha sido possível pegar qualquer cadeira até a penúltima, a última posição está vazia e com um prato e a cadeira anterior (penúltima) está vazia e com um prato, então posso me sentar na última posição e pegar mais um prato na penúltima posição. Por outro lado, caso a penúltima cadeira não esteja vazia E COM UM PRATO, a primeira cadeira (separada de mim pelo vão) deve estar vazia e com um prato e a penúltima cadeira deve ter um viking do meu tipo, ou estar vazia (sabidamente sem prato).

Agora, na função **chieftain_release_seat_plates(...),** é verificado se o prato foi pego no sentido horário ou anti-horário, e após isso é colocado novamente o prato na cadeira. Liberando também os vikings que já comeram.

3. Preces

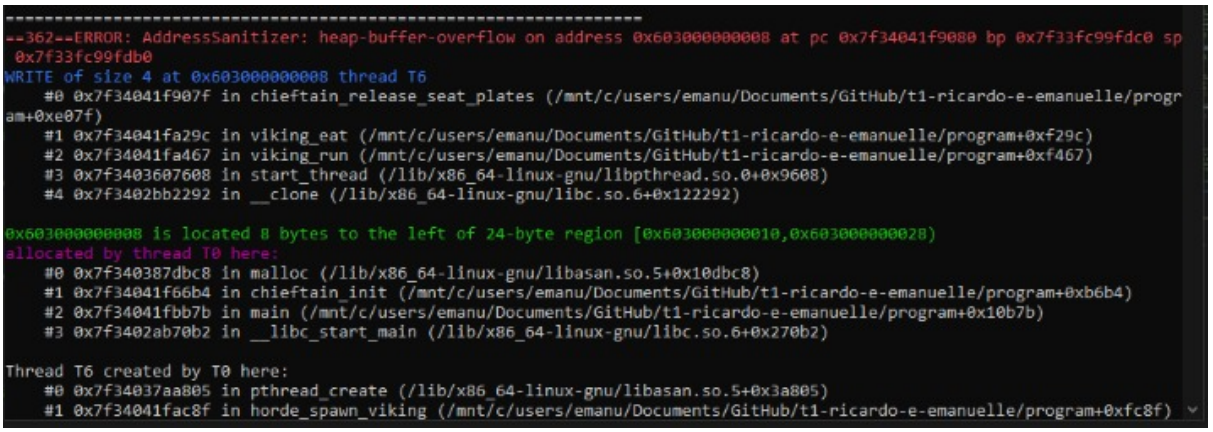
No código das preces, foi implementado mutexes que verificam se o contador é igual ao número de Vikings, esperando os Vikings terminarem de comer. Após isso, é escolhido um Deus aleatoriamente, tendo um tratamento para cada caso:

- Caso 0: reza para BALDR se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no BALDR ou LOKI.
- Caso 1: reza para LOKI se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no LOKI ou BALDR.
- Caso 2: reza para VALI se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no VALI ou HODER.
- Caso 3: reza para HODER se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no HODER ou VALI.
- Caso 4: reza para FRIGG se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no FRIGG ou JORD.
- Caso 5: reza para JORD se a regra dos deuses com rixa permitir ou se não tiver tido nenhum voto no JORD ou FRIGG.
- Caso 6: reza para ODIN se a regra dos deuses sem rixa permitir.
- Caso 7: reza para THOR se a regra dos deuses sem rixa permitir.

Por último, é finalizado o código, pela função **`chieftain_finalize(...)`**, a qual possui a destruição dos mutexes implementados, dos semáforos e a liberação de memória.

4. Dificuldades

Enfrentamos alguns problemas para implementar e entender a lógica dos semáforos quando aplicadas no código, além de que em diversos momentos por erro de implementação, acaba por resultar em alguns erros na execução, como:



O qual foi corrigido pensando melhor na forma que estava sendo implementado alguns index posições da *chairlist*.

Soma-se a isso, a dificuldade em repassar para o código de uma forma coesa a lógica que havíamos pensado para os pratos e cadeiras, e a falta de experiência com a linguagem C.