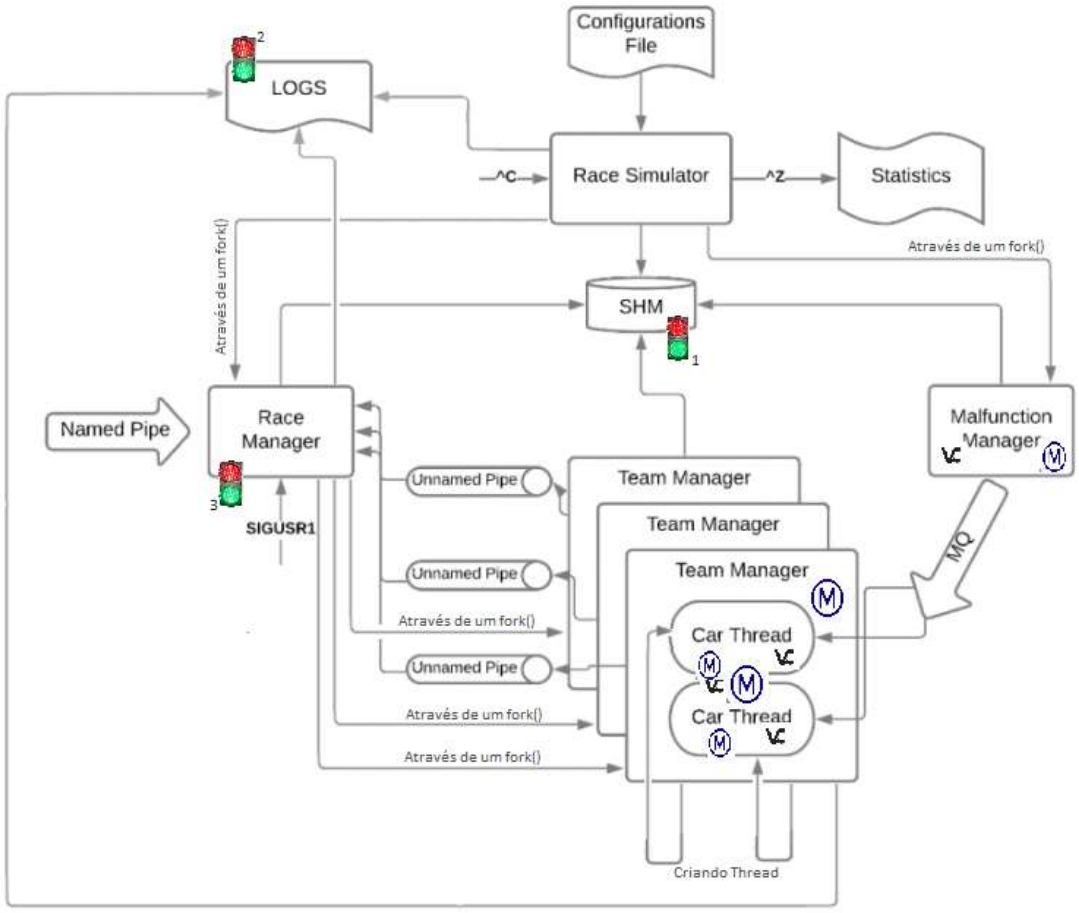
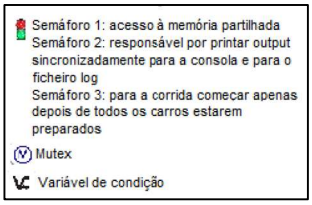
Projeto SO

Duarte Emanuel Ramos Meneses, 2019216949 – 120 horas
Patrícia Beatriz Silva Costa, 2019213995 – 96 horas





O programa inicia-se com a abertura do ficheiro de *log* onde serão registadas todas as informações respeitantes à aplicação. Posteriormente, cria-se um semáforo que será utilizado para enviar de forma síncrona o *output* para o ficheiro de *log* e para o ecrã. Caso este semáforo não existisse, corríamos o risco de os prints não serem executados na ordem correta e com a informação pretendida. É criada memória partilhada (*SHM*) para uma estrutura que conterá as informações dos carros e das boxes/equipas. Todos os processos que acedem a esta memória partilhada irão fazê-lo através de um semáforo (semáforo 1 no diagrama) para que apenas 1 processo possa acedê-la de cada vez. Se isto não acontecesse, por exemplo, se 2 processos acedessem a essa memória ao mesmo tempo, 1 para ir buscar o valor de uma variável e o outro para o atualizar, caso a atualização se efetuasse posteriormente quando devia ser antes, o processo que foi buscar o valor teria acesso a um valor desatualizado.

De seguida, as informações relativas às configurações da corrida são lidas do ficheiro correspondente e guardadas numa estrutura criada para o efeito. Note-se que todas estas operações são realizadas no processo respeitante ao simulador de corrida (*Race Simulator*). A seguir é criado o processo gestor de corrida (*Race Manager*) que irá, entre outras coisas, criar os processos gestor de equipa (*Team Manager*) - 1 por

equipa - e lê os comandos vindos de um *named pipe*. O programa valida esses comandos e após receber a informação para começar a corrida, se estiverem reunidas todas condições para começar (número de equipas definido pelo ficheiro de configurações, por exemplo), recusa qualquer novo comando dizendo que a corrida já começou.

Em cada gestor de equipa serão criadas *x threads* (*Car Thread*) sendo *x* o número de carros em competição dessa equipa. Esse trabalho só acontecerá quando se der ordem para começar a corrida (para não criar recursos desnecessários antes do tempo). Para isso, colocamos uma variável de condição que só irá desbloquear quando for dada ordem de início de corrida. A corrida só começa depois de todos os carros estarem prontos (threads todas criadas). Para isso, utilizamos semáforos para sincronizar. Na função de gestão de cada carro, existe um *mutex* uma vez que apenas 1 carro pode estar na box de cada vez (abre a box para a entrada de um carro se necessitar). Como apenas um carro pode estar na box, o carro só tenta entrar se o estado da box não indicar que está ocupada. A cada unidade de tempo passada, o carro avança X metros e gasta Y combustível, valores indicados no comando passado pelo *named pipe* e de acordo com o estado do carro.

No processo do simulador de corrida é ainda criado outro processo que dirá respeito ao gestor de avarias (*Malfunction Manager*), responsável por gerar avarias nos carros. Esta função só começa a gerar avarias quando a corrida começar. Para isso utilizamos uma variável de condição associada a um mutex que verifica o estado da corrida. O gestor de avarias comunica com cada carro (*thread*) através de *message queue*. Decidimos que a melhor maneira de o fazer seria utilizar o índice do carro no *array* de carros + 1 (para não utilizar o 0) como *message type* porque este é um identificador único de cada carro. Recebendo uma mensagem de avaria, o carro entra em modo de segurança (se ainda não estiver) e reduz a sua velocidade e o consumo de combustível.

Quando um carro entra em estado de segurança, a box passa ao estado reservado. Isto faz com que apenas possam entrar carros em estado de segurança e não aqueles que, em estado de corrida, precisam de abastecer. O carro estando em estado de segurança, passa a tentar entrar na box (sempre que passar pela linha da meta) e caso a box esteja livre nessa altura, entra para ser arranjado. Aproveitando que está na box, é também abastecido. O abastecimento demora 2 segundos, enquanto o arranjo da avaria demora um tempo aleatório definido entre 2 valores presentes no ficheiro de configurações. Para o carro parar a sua corrida na função que diz respeito a controlar o funcionamento do carro (que gere a *thread*), colocamos uma variável de condição que só irá desbloquear quando o carro sair da box. Ao sair da box, já virá em estado de corrida.

A corrida acaba quando todos os carros cruzarem a meta tantas vezes quanto o número de voltas especificado no ficheiro de configurações ou quando estejam em modo de desistência (ficar sem combustível). Quando isto acontece, é impresso o vencedor da corrida.

Caso o processo simulador de corrida (*main*) receba o sinal SIG_INT, espera que todos os carros terminem a corrida, imprime as estatísticas da mesma, liberta os recursos utilizados e fecha o simulador. Já se receber o sinal SIG_TSTP, imprime apenas as estatísticas. Já o gestor de corrida recebendo o sinal SIG_URS1, tal como quando acontece com SIG_INT, vai esperar que todos os carros terminem a corrida e imprime as estatísticas da mesma.