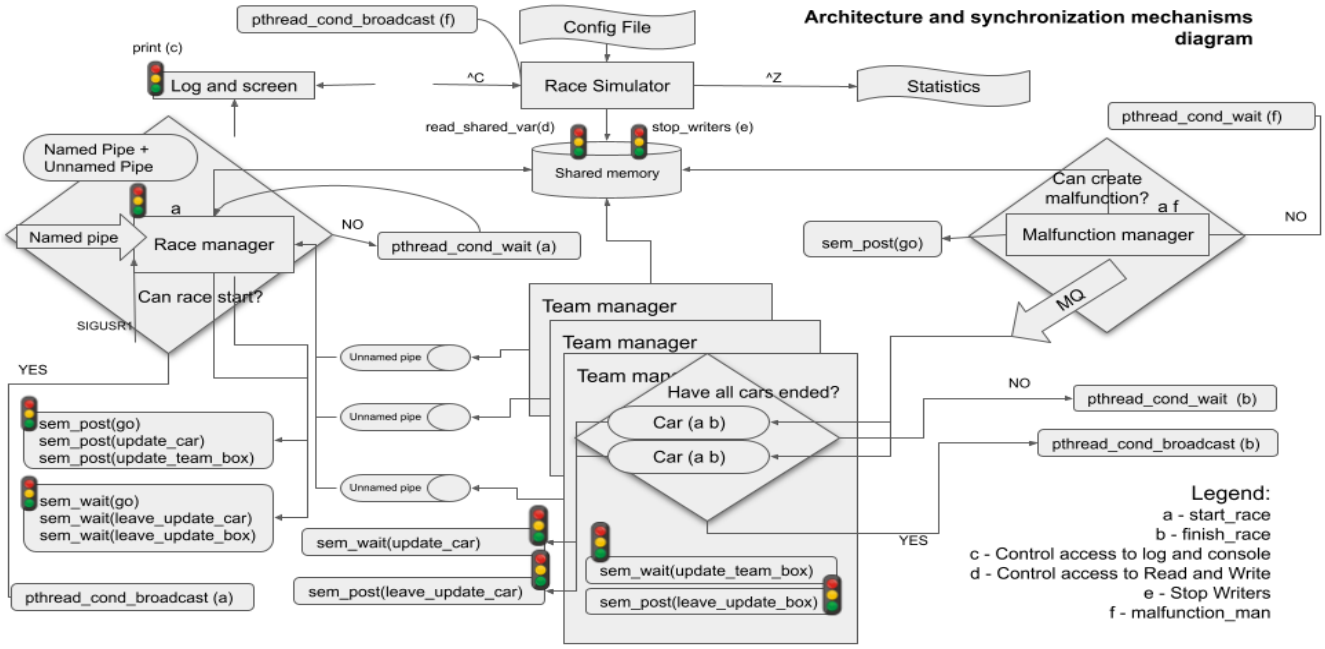
Arquitetura e diagrama com mecanismos de sincronização:



Mecanismos implementados:

- Variável de condição start_race (a) - Serve para notificar os processos Race manager e Malfunction manager do início da corrida
- Variável de condição finish_race (b) - Serve para notificar as threads carro que já podem sair da corrida
- Mutex para ficheiro log e ecrã (c) - Serve para impedir acessos ao mesmo tempo de escrita para o ficheiro log e ecrã
- Mutex para controlar escritas e leituras para a Shared Memory (d) - Serve para controlar os acessos às escritas e leituras da shared memory para não haver acessos concorrentes
- Mutex para controlar a escrita para a Shared Memory (e) - Serve para controlar as escritas à shared memory pois não podem haver escritas quando não houverem leituras
- Variável de condição malfunction_man (f) - Serve para notificar o Malfunction manager sempre que passou T_avaria de tempo

Opções Tomadas:

- Numa fase inicial, começa-se por bloquear todos os sinais antes de concluir a função ***initialize***. Após, desbloqueiam-se os sinais **SIGINT** e **SIGTSTP** para poderem ser tratados pelo processo Race Simulator como pedido. O tratamento dos sinais é feito através das funções ***termination_handler*** e ***end_race_signal***. Todos os processos filhos, nomeadamente, Race Manager e Malfunction Manager irão bloquear os sinais **SIGINT** e **SIGTSTP**.
- A função ***initialize*** lê o ficheiro o ficheiro de configurações e coloca os dados em memória, inicializa a shared memory, semáforos named e unnamed, mutexes, named e unnamed pipes.
- Para simular o tempo da corrida, foi decidido não utilizar a função *sleep*. São, portanto, usadas variáveis de condição e semáforos para sincronizar os diferentes processos e threads.
- O processo ***Race Manager*** é responsável por ler do named pipe, criar processos para as equipas e "simular o tempo". Antes da corrida começar, é criada uma thread para gerir os named e unnamed pipes (**pipe_manager**) e, depois, o processo **Race Manager** irá esperar na variável de condição **start_race** que indica o começo da corrida. Esta variável só irá ser notificada quando for enviado um "Start Race!" através do named pipe e, após todas as equipas estarem criadas. Quando isto acontecer, a função **check_named_pipe_data** que controla as mensagens do named pipe transmite aos processos Race Manager e Malfunction Manager o início da corrida.

  Quando a corrida é inicializada, cria-se todos os processos **Team Manager**, que, por sua vez, vão criar todas as **Car Threads**. Quando todas as threads carro terminarem de ser criadas - verificação esta que é feita com um **sem_wait(&cars[index].mutex_finish_car)** de todos os carros em corrida - começará então a corrida.

  A corrida é representada através de um ciclo infinito que acabará quando não existirem mais threads carros em corrida, cada ciclo representa um segundo que se passa na corrida.

  Em primeiro lugar, é verificado se é necessário notificar o Malfunction Manager para atualizar. Se

sim, então o Malfunction Manager é notificado através da variável de condição ***malfunction_man*** e, assim que terminar, irá libertar o semáforo ***go***. Se não, então o Race Manager liberta-o.

Em segundo lugar, antes de se atualizarem os carros, é aguardado este mesmo semáforo **go** e, de seguida, é libertado o semáforo ***mutex_start_car*** de cada thread carro para todos os carros atualizarem e, quando terminam, libertam o semáforo **mutex_finish_car**, semáforos estes que estão a ser aguardados pelo Race Manager.

Em terceiro lugar, antes de se atualizar as boxes das equipas, é verificado quantas boxes das equipas precisam de ser atualizadas dependendo se ainda têm carros em corrida dessa mesma equipa ou não (se não precisar de ser atualizada simplesmente espera no semáforo **mutex_start_team**). Depois, liberta-se o semáforo ***mutex_start_team*** de cada Team Manager que precisa de atualizar, e é aguardado o ***mutex_finish_team*** de cada uma dessas equipas.

No final, quando já não existirem mais equipas, notifica-se os processos **Team Manager,** o processo **Malfunction Manager** e a thread **pipe_manager,** estes irão depois verificar que já não há mais carros em corrida e, portanto, sairão dos seus ciclos e o Race Manager também. Depois espera-se então pela saída controlada da thread dos pipes e de todos os processos Team Manager.

- O processo ***Malfunction Manager*** começa por esperar ser notificado pela variável de condição ***start_race*** onde aguarda que a corrida comece. Após, espera ser notificado pelo processo Race Manager, processo este que após T_avaria de tempo envia um sinal para a variável de condição ***malfunction_man***. Quando existir o sinal, significa que está na hora de existir uma malfunction e então é gerada (ou não) uma avaria para todos os carros em corrida que não tenham já uma avaria, avaria esta que é enviada através da Message Queue ***mqid***. Quando este terminar de dar update às avarias, dá post ao semáforo ***go*** e fica novamente à espera da variável de condição ***malfunction_man***.
- Os vários processos ***Team Manager*** começam por criar as threads para os seus respectivos carros. Depois, aguarda ser notificado pelo semáforo ***mutex_start_team*** para começar a atualizar as boxes de cada equipa. Caso já não haja carros em corrida, o processo é terminado. Caso contrário, verifica se a sua box está ***OCCUPIED***, caso esteja, atualiza o tempo que já passou desde o início da reparação, se esse mesmo tempo terminar então o carro está reparado, com combustível e o estado voltar a estar ***FREE*** e é, novamente, verificado o estado da box. Caso a box esteja livre, vai ver se algum dos seus carros está em modo ***SECURITY***, se estiver vai alterar o estado da box para ***RESERVED***. Se a box estiver ***RESERVED*** verifica todos os carros em estado **SECURITY** se estão nos zero metros, caso estejam, o primeiro que chegar altera o estado da box para ***OCCUPIED*** e, no ciclo de tempo seguinte, procede à reparação e/ou colocação de combustível no carro. Caso a box esteja ***FREE*** e o carro esteja nos zero metros e precise de combustível, altera o estado da box para **OCCUPIED** e, no ciclo de tempo seguinte, coloca combustível. No fim de atualizar, dá post ao semáforo ***mutex_finish_team***. Caso não existam mais carros em corrida, aguarda a saída controlada das threads e sai.
- As **Car Threads** começam por libertar o semáforo ***mutex_finish_car*** para sinalizar ao Race Manager que já foram criadas. Em seguida, entram num ciclo que representa a atualização,em cada segundo de corrida, dos carros. Começam por verificar a sua Message Queue se for o tempo disso (T_avaria), pois é por aí que são notificados de avarias. Caso não tenha nenhuma avaria, faz os seus respectivos consumos, estando em estado ***NORMAL*** ou ***SECURITY***. Se estiver em ***SECURITY*** verifica se ficou sem combustível ou não, caso esteja em estado ***NORMAL*** vai fazer as verificações se passa ao estado de ***SECURITY*** ou não. No final libertam o semáforo ***mutex_finish_car*** indicando que concluíram a atualização.

  Se houver alteração do estado do carro envia-se uma notificação via unnamed pipe.

  Se o carro atingir o fim da corrida ou caso tenha ocorrido o sinal ***SIGINT***, o carro ficará à espera da variável de condição ***finish_race*** se não for o último em corrida. Se for, então notifica todas as threads carro (incluindo a ele mesmo) que podem sair.

Trabalho futuro:

Como trabalho futuro, como falta tratar do sinal **SIGUSR1**, começaríamos por bloquear o sinal no Race Manager até a corrida começar. Após a corrida começar, como não queremos receber o sinal em nenhum momento crítico do código, usaríamos a função **sigprocmask** para bloquear o sinal até terminar um segundo na corrida representado como um ciclo, após terminar um ciclo o sinal ia ser tratado com o **termination_handler**. Aí esperaríamos que os carros chegassem à meta e, logo após, reiniciariamos todas as variáveis necessárias à corrida a 0. Por fim, esperava-se pelo sinal do Named Pipe com “Start Race!” para correr de novo a corrida.

As **Car Threads** e os **Team Manager** não podiam terminar, portanto seria feita uma verificação de que se o sinal está ativo, então não é para terminarem.

Tempo dispendido:

Samuel Tiago Almeida Pires nº 2019225195: 70h

Sofia Santos Neves nº 2019220082: 70h