Relatório do Simulador de Corrida

O processo Race Simulator cria a memória partilhada e inicializa-a. Desta fazem parte os mutex que posteriormente irão ser usados na sincronização dos restantes processos e inicializados com PTHREAD_PROCESS_SHARED. Por outro lado, este também permite criar a Named Pipe, a message queue e ainda os processos Race Manager e o Malfunction Manager.

Optamos por dividir a memória partilhada em três variáveis, um ponteiro para os carros, um outro ponteiro para as equipas e uma variável onde estão guardadas aas informações gerais da corrida.

O processo Race Manager começa por criar processos Team Manager ficando estes a aguardar o início da corrida através de uma variável de condição. São também criadas as unnamed pipes, uma por equipa. É através de um select que o race manager aguarda a receção de comandos, posteriormente verificados, pela named pipe e as alterações do estado dos carros pelas unnamed pipes. Decidimos utilizar um select uma vez que é um comando que não provoca uma espera ativa.

Os processos Team Manager começam por aguardar a notificação do início de corrida do unnamed pipe através de uma variável de condição. Recebida a notificação, as threads dos carros são criadas e a corrida inicia-se. Após criar as threads, o Team Manager gere a box da sua respetiva equipa, aguardando ser notificado por uma das threads de forma a evitar esperas ativas. Durante o acesso à box é verificado o estado em que se encontra o carro, caso se encontre no estado de segurança (1) procede-se apenas à reserva da box caso contrário, verifica-se se ocorreu uma avaria e se tal se verificar gera-se um valor de tempo entre os limites estabelecidos nos ficheiros de configurações para a reparação. A reparação é feita com a box a aguardar durante o período de reparação a notificação da thread do controlo do tempo e por sua vez a thread carro aguarda a notificação do Team Manager.

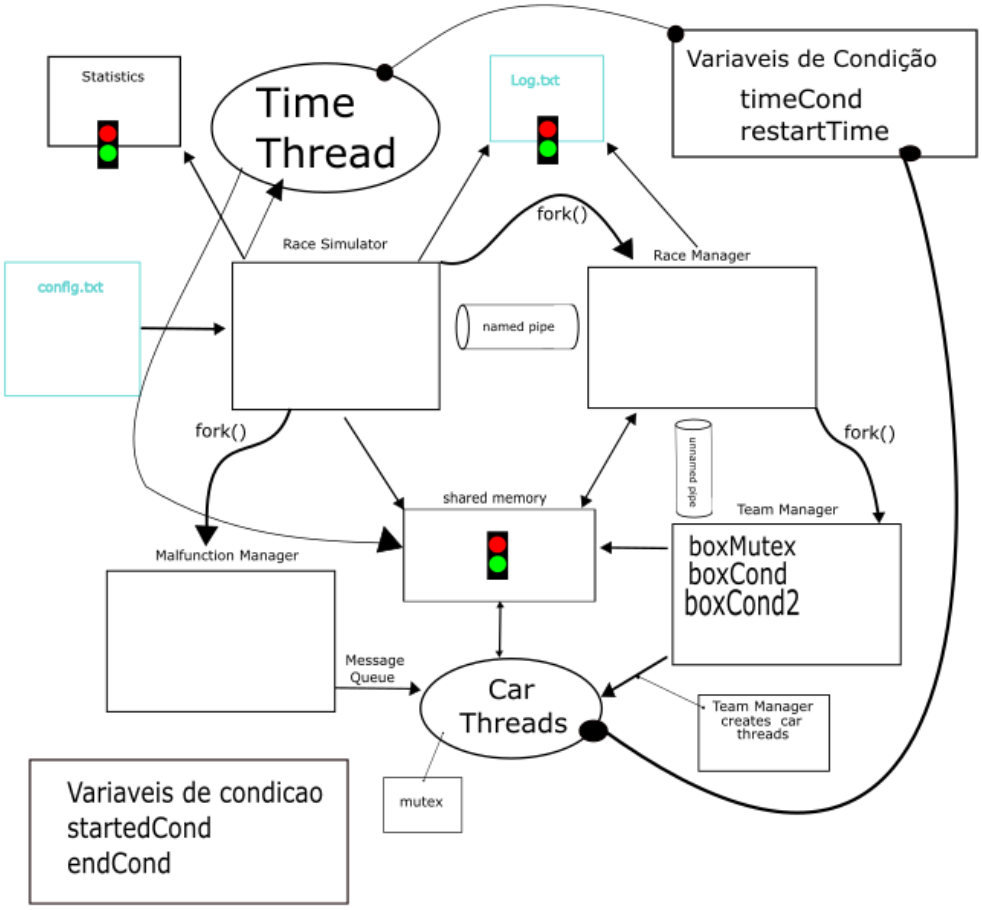
As threads carro estão sincronizadas com a passagem do tempo através de uma variável de condição de modo que nenhuma das threads fique numa situação de starvation. As avarias, geradas pelo Malfunction Manager (explicado mais à frente), chegam às threads pela message queue que é verificada no início da simulação de cada unidade de tempo em modo IPC_NOWAIT. O acesso à box é implementado através de dois mutex: o primeiro é utilizado para garantir a exclusão mútua de modo que apenas o carro que se encontra na box capture, tal é conseguido recorrendo à função pthread_mutex_trylock de forma a fazer uma verificação não bloqueante do estado do mutex; o segundo mutex é utilizado na coordenação box-thread.

O Malfunction Manager permanece à espera numa variável de condição e quando notificado pela thread responsável pelo controlo do tempo, percorre os carros gerando avarias nos mesmos que se encontrem em corrida e fora da box, e que não se encontrem já com uma avaria. O malfunction gera um número aleatório utilizando o seu pid como seed pois é algo que varia de corrida para corrida e permite introduzir aleatoriedade ao processo, caso este número seja superior à fiabilidade do carro em questão é então gerada uma avaria.

A thread responsável pelo controlo do tempo é criada pelo Race Simulator e permanece inativa através de uma variável de controlo até ao início da corrida. Após incrementar a variável de tempo, espera que todos os carros terminem a simulação daquela unidade de tempo antes de incrementar outra vez de modo que nenhum carro fique numa situação de starvation. Caso a variável de tempo seja um múltiplo do tdamage o Malfunction Manager é notificado. Esta thread

recorre a um usleep de forma a respeitar a duração de cada unidade de tempo definida no ficheiro de configurações.

Para o ordenamento e posterior impressão das estatísticas recorre-se a um bubble sort e coordena-se a aquisição de dados com o momento de impressão através de um named semáforo Posix.



Inês Martins Marçal 2019215917 36h

Noémia Quintano Mora Gonçalves 2019219433 36h