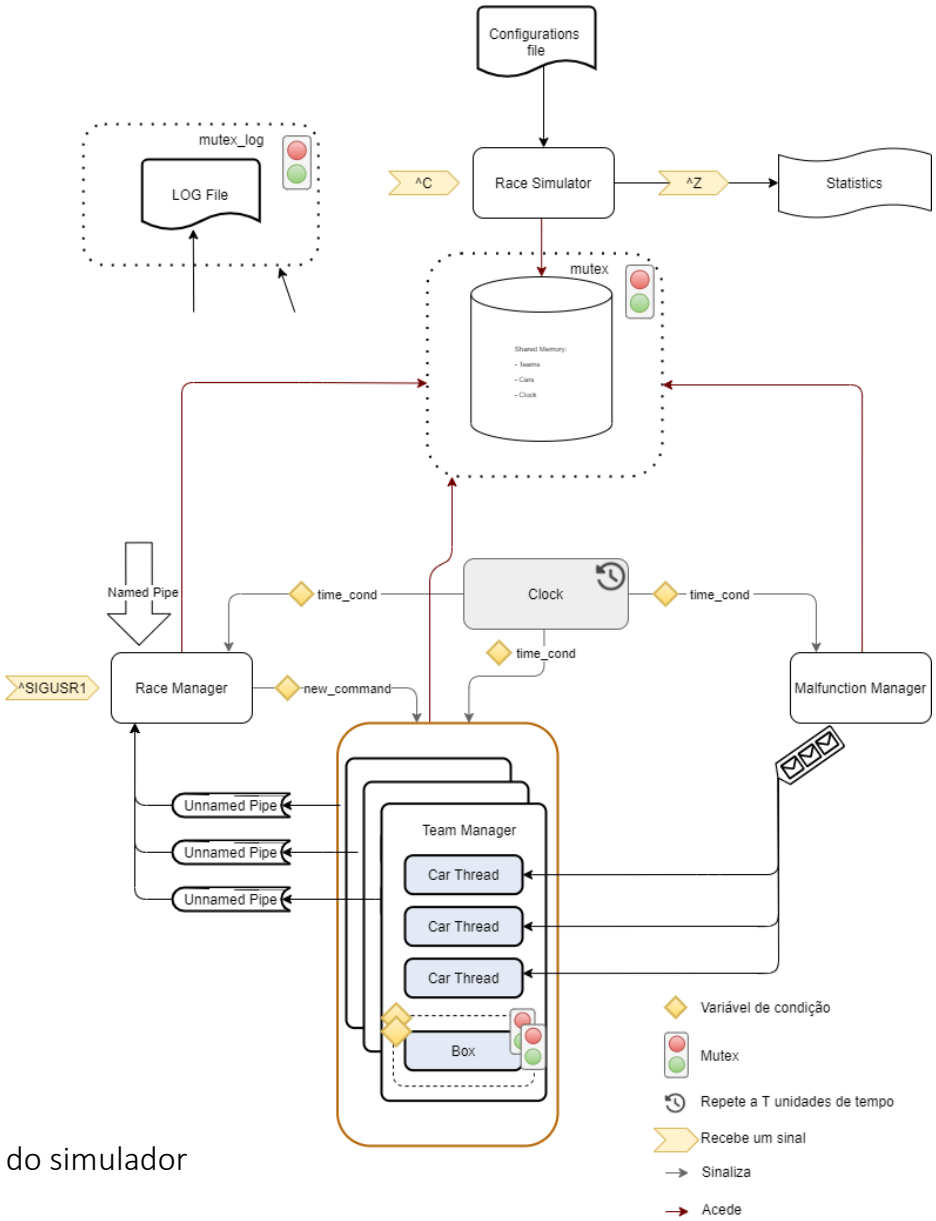
Universidade de Coimbra
Licenciatura em Engenharia Informática
Sistemas Operativos 2020 / 2021

Simulador de Corridas

Joana Maria Silva Simões, Nº2019217013

Samuel dos Santos Carinhas, Nº2019217199



Funcionamento do simulador

Processos

Race Simulator

Ao iniciar o programa, o Race Simulator começa por ler e validar o ficheiro de configurações, e de seguida, cria e inicializa a memória partilhada bem como todos os mecanismos de sincronização (mutexes, variáveis de condição (VC) e semáforos). Após isto, são criados os processos Malfunction Manager, Race Manager e Clock.

Este processo trata dois sinais (SIGINT e SIGTSTP): quando recebe o sinal SIGINT, coloca a flag *end_race* com o valor lógico 1 de forma a sinalizar os restantes processos de que devem terminar, ficando à espera deles; quando recebe

o sinal SIGTSTP chama a função *show_statistics*. Nesta função, é dado um *lock* do *mutex* relativo a cada carro (*car_mutex),* garantindo assim, que a informação atual da corrida não é alterada no mesmo instante em que está a ser apresentada.

Malfunction Manager

Quando este processo é iniciado, começa por esperar um sinal da VC *new_command* que altere o estado da variável *race_started* para 1, de forma a começar a gerar avarias. Para poder reinicializar a corrida, o ciclo de produção avarias está embutido noutro ciclo de forma a poder continuar a gerar avarias após esta interrupção.

Clock

Este processo simula um relógio global que a cada unidade de tempo notifica todos os outros processos através da VC *time_cond*, de forma a estes poderem executar de forma síncrona.

Race Manager

Recebe e trata os comandos provenientes do *Named Pipe*, colocando a *flag race_started* a 1 ao receber o comando "START RACE!" para iniciar a corrida. Após este comando, começa a receber mensagens dos carros através de *Unnamed Pipes,* identificando também quando o último carro termina. Estas duas leituras são realizadas recorrendo à função *select* para evitar esperas ativas. Este processo fica também encarregue de tratar o sinal SIGUSR1, colocando as *flags restarting_race* e *end_race* com valor lógico 1 de modo a alertar os restantes processos que a corrida vai reiniciar, ficando, assim, de seguida à espera de que todos processem essa *flag* e o sinalizem de volta usando a VC *reset_race.*

Team manager

É o processo responsável pelas *threads* dos carros e da *box.* Inicialmente fica à espera de ser sinalizado pela VC *new_command* para criar as *threads* de novos carros, e após a corrida começar, cria também a *thread* da *box.* Quando a corrida é reinicializada, espera que todas as threads terminem e notifica o *Race Manager*.

Car thread

Simula o comportamento do carro durante a corrida, incrementa a distância e decrementa o combustível a cada unidade de tempo. Quando necessita de ir à *box*, faz-lhe um pedido através da VC *request*, e, se puder entrar, vai aguardar pela VC *car_leave* significando que o carro abandonou a *box e que pode continuar.* De cada vez que o carro completa uma volta verifica o estado das flag *end_race,* terminando a corrida se esta estiver a 1.

Box

Espera por pedidos (reabastecimentos ou reparação de carros e atualização do seu estado) e simula o tempo necessário ao seu processamento. Na atualização do estado da box, se todos os carros já tiverem terminado a corrida, a *thread box* termina.

Ficheiro Log

Toda a informação relativa ao simulador, é escrita simultaneamente no *stdout* e no ficheiro "log.txt". Para permitir a sincronização da escrita em ambos, uma vez que vários os processos podem escrever neles ao mesmo tempo, é usado um *mutex mutex_log*. De forma a evitar sucessivas reaberturas do ficheiro "log.txt", é inicialmente criado um *file descriptor* no processo *Race Simulator* relativo ao *log*.