Relatório

TRABALHO PRÁTICO

DE

SISTEMAS OPERATIVOS

Etiandro António 2017290285 e João Pedro Dionísio 2019217030

Introdução ao problema

Neste projeto era-nos pedido que criássemos um simulador de corrida. Tal como qualquer simulador de corrida este teria uma lista de equipas, cada uma com uma lista dos carros que lhe pertencem. Para além disso o cérne do problema está na gestão de vários processos e de várias threads com recurso a semáforos, message queues e outros.

Solução

O simulador começa na função "main", onde chama o config() que serve para ler e guardar todos os dados referentes ao ficheiro de texto "config.txt". Chama também o project_output_log() e o log_file_write(), que estão direcionados á escrita no ficheiro do tipo log, antes de iniciar os processos ***Gestor De Corrida*** e ***Gestor de Avarias***.

No Simulador de Corrida temos o *signal handler* que lida com o sinal SIGTSTP onde sempre que o sinal é capturado chama a função *print_stats* que trata de imprimir para o ecrã toda a informação necessária para a estatística, o sinal SIGINT chama a função *signal_sigint* onde termina os processos e remove/liberta os recursos, não conseguimos fazer com que a função esperasse os carros terminarem a corrida devido ao tempo que dividimos para terminar o projecto.

No processo ***Gestor de Corrida***, antes de criarmos os processos ***Gestor de Equipa*** verificamos os comandos, caso o utilizador adicione um carro verificamos os valores introduzidos no comando se são válidos e depois definimos os valores do carro com os valores que recebemos do comando, para validação dos comandos usamos algumas flags, depois de recebermos o comando "*START RACE*" o processo ***Gestor de Corrida*** cria os processos ***Gestor de Equipa***.

No processo ***Gestor de Equipa*** criamos todas as *threads* carros, usamos uma flag ***isEmpty*** na estrutura das *threads* Carros que serve para verificar se a *thread* daquele especifico carro já foi criada, assim certificamos que criamos as *threads* ao mesmo tempo para que a corrida inicie no mesmo instante, para a funcionalidade de abastecer os carros e reparar os carros na box, o processo verifica a flag **isInBox** que temos na estrutura das threads carros que caso seja *true* significa que o carro está na box, então depois do carro levar o seu tempo na box, o processo põe o valor da box à 0 e faz um sinal com uso da variável de condição para a *thread* que se encontra na box a espera do sinal para poder continuar a corrida. Já no processo ***Gestor de Avarias*** ele simplesmente verifica os estados de todos os carros e caso estejam a correr e não no modo segurança, gera um número aleatório de 1 à 100 que posteriormente é comparado com o valor da *reliability* de cada carro e caso esse valor aleatório seja maior que o valor de *reliability* do carro então ele manda uma mensagem de avaria, ele manda um inteiro "avaria" com

o valor de 1, e na *thread* carro o que fazemos é logo que recebe uma mensagem(com a flag ***IPC_NOWAIT*** para que possa continuar a corrida caso não haja mensagens), ele entra no modo segurança mais propriamente no modo avaria e faz uma requisição para ir a box onde caso ela não esteja ainda reservada ele reserva, e logo que o carro da sua equipa sair da box, ele ocupa o lugar.

Nas *threads* carros eles todos percorrem um loop que vai até a última volta, caso eles estejam no modo segurança tentam entrar na box e caso ela esteja ocupada eles continuam a corrida e se ela estiver livre eles entram na box onde esperam um sinal do ***Gestor de Equipa***(variável de condição) que depois de reparar e abastecer os carros os carros são liberados da box e podem prosseguir a corrida, temos as verificações todas dos carros em funcionamento e caso ele desista o carro sai da pista.

DIAGRAMA DA ARQUITETURA DO PROGRAMA

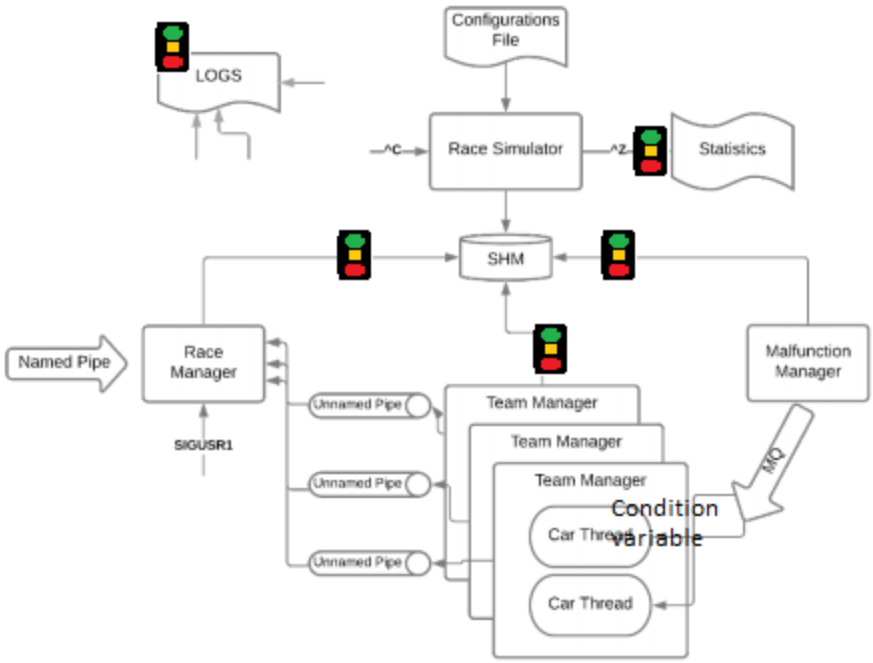

Figura 2: Visão geral do simulador a desenvolver