Relatório Do Projeto Sistemas Operativos

Diogo Malheiro Boinas(2016238042)
Roman Zhydyk(2016231789)

Horas despendidas no projeto: 30 horas

Arquitetura

Simulation Manager

- Inicia todos os recursos utilizados no projeto;
- Lê as estatísticas da memória partilhada através do sinal SIGUSR1;
- Criação da torre de controlo;
- Leitura do Named Pipe;
- Termina o programa de forma controlada via ctrl+c(SIGINT);

Control Tower

- Faz gestão do vôos;
- Comunica com as threads do vôo pela message queue e pela shared memory;
- Atualizar shared memory relativa aos vôos, sendo que vai existir um array com as slots para cada vôo e, aqui, fazemos a atribuição de um slot a um vôo;
- Criamos threads para verificações de tempo no caso das threads vôo;
- Caso um vôo seja prioritário, este não vai fazer holding e comunicamos pela shared memory com a thread do respetivo vôo;

Flight

- Temos duas rotinas para cada tipo de vôo(arrivals e departures);
- No rotina do arrival, inicializamos a thread depois do tempo(init) via uma função extra chamada msleep(), notificamos a torre de controlo via message queue do seu eta e do fuel, depois ficamos à espera da resposta do slot de memória partilhada atribuído a este vôo e usamos a memória partilhada para a gestão dos vôos;
- No rotina do arrival, inicializamos a thread depois do tempo(init) via uma função extra chamada msleep(), notificamos a torre de controlo via message queue do seu tempo de takeoff, depois ficamos à espera da resposta do slot de memória partilhada atribuído a este vôo e usamos a memória partilhada para a gestão dos vôos;

Logs

- Criamos um ficheiro “logs.txt” e um named semaphore para sincronização;
- Vamos buscar o tempo atual por via da struct time_t;

- Damos print e escrevemos no ficheiro a string passada como parâmetro ao mesmo tempo usando o semáforo;

Shared

- Neste ficheiro temos várias funções auxiliares ao projeto e algumas inicializações;

Mecanismos De Sincronização e Estratégias Tomadas Na Resolução de Problemas

- Os mecanismos de sincronização usados foram mutexes para controlar threads e named semaphores para controlar acesso à memória partilhada e para os logs, para os logs escrevem no ficheiro e darem print ao mesmo tempo e no caso de o vôo estar a holding usamos o pthread_cond_wait();
- Usamos duas listas ligadas para receber os vôos do pipe, depois criamos a thread fazendo pop do primeiro elemento de uma destas listas. Assim estamos a fazer um escalonamento FIFO;
- Dentro da torre de controlo, criamos uma thread para cada vôo com o intuito de fazer gestão de cada vôo. São estas as threads que controlam o tempo e outras caraterísticas específicas de um vôo;