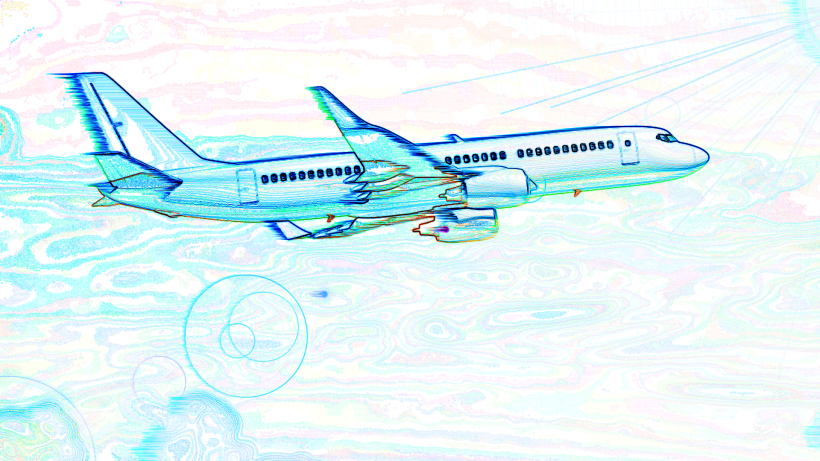


**Instituto Superior de Engenharia de Coimbra**

**Licenciatura em Engenharia Informática**

**Sistemas Operativos 2**



**Leandro Adão Fidalgo | a2017017144**

**Pedro dos Santos Alves | a2019112789**

**Laboratório P5**

**Trabalho Prático 1**

**Meta 1**

**Coimbra, 16 de maio de 2021**

**Índice**

[1. Introdução 1](#_Toc11026)

[2. Mecanismos de comunicação e sincronização 2](#_Toc2406)

[2.1. Memória partilhada 2](#_Toc1116)

[2.2. Mutexes e Semáforos 2](#_Toc13912)

[2.3. Heartbeat 2](#_Toc18880)

[2.4. Critical Section 2](#_Toc7585)

[2.5. Eventos 2](#_Toc6433)

[3. Estruturas de dados 3](#_Toc7401)

[3.1. Estrutura Airport 3](#_Toc23469)

[3.2. Estrutua Airplane 3](#_Toc31289)

[3.3. Estrutura Command 3](#_Toc8823)

[3.4. Estrutura SharedBuffer 4](#_Toc8395)

[3.5. Estrutura SharedMemory 4](#_Toc30870)

[4. Bibliotecas dinâmicas 5](#_Toc4366)

[5. Conclusão 6](#_Toc24465)

[Manual de utilização I](#_Toc1527)

[Controlador (Control) I](#_Toc4069)

[Iniciar I](#_Toc8681)

[Comando “help” I](#_Toc9950)

[Comando “add” I](#_Toc18352)

[Comando “remove” II](#_Toc29726)

[Comando “toggle” II](#_Toc10328)

[Comando “list” III](#_Toc27787)

[Comando “kick” III](#_Toc5373)

[Comando “exit” III](#_Toc13689)

[Avião (aviao) IV](#_Toc23482)

[Iniciar IV](#_Toc4681)

[Comando “help” IV](#_Toc29020)

[Comando “destination” IV](#_Toc4049)

[Comando “board” V](#_Toc30102)

[Comando “start” V](#_Toc25680)

[Comando “list” V](#_Toc19893)

[Comando “exit” VI](#_Toc28604)

1. Introdução

O presente relatório descreve o projeto desenvolvido pelos alunos: Leandro Fidalgo e Pedro Alves, no âmbito da disciplina de Sistemas Operativos 2 da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

A primeira meta do trabalho prático pretende-se que sejam feitas as seguintes funcionalidades: o Controlador aéreo (control), com uma interface do tipo consola, cria o(s) mecanismo(s) de comunicação e sincronização com os programas que representam os aviões. Atende até um máximo de aviões definido no Registry. Comunica com os aviões em ambos os sentidos. Cria e gere as estruturas de dados a usar pelo sistema. O Avião (aviao) – Desloca-se, evitando colisões em voo com outros aviões, usa a biblioteca DLL fornecida pelos docentes para saber qual será a próxima posição a ocupar na sua trajetória.

O objetivo deste trabalho consiste num sistema de gestão do espaço aéreo.

O objetivo do presente trabalho é consolidar todos os conhecimentos adquiridos nas aulas teóricas e práticas ao longo de todo o semestre.

1. Mecanismos de comunicação e sincronização
   1. Memória partilhada

Como mecanismo de comunicação entre o Controlador e o Avião foi usada a memória partilhada. A memória partilhada dispõe de um mapa no qual o avião consegue verificar as posições, ainda na memória partilhada existe uma indicação para os Aviões, que ainda não se conectaram, saberem se o controlador está a aceitar aviões ou não.

Para as restantes mensagens foi utilizado o paradigma produtor/consumidor através do uso de buffers circulares. Estes buffers circulares contêm a informação necessária para que a comunicação entre o Controlador e o Avião seja efetuada com sucesso. A informação contida nos buffers circulares diz respeito a quem enviou a mensagem (produtor), de quem receberá a mensagem (consumidor), o código do comando associado á mensagem e os dados do comando.

* 1. Mutexes e Semáforos

Para manter a atomicidade e garantir a consistência dos dados partilhados na memória partilhada foram utilizados mutexes. Os mutexes permitirão que a memória apenas irá ser acedida por um processo ou thread de cada vez. Para os buffers circulares foram utilizados semáforos e mutexes. Os semáforos deixarão aceder aos buffers circulares, até um número máximo de processos ou threads, dependendo do número de espaços vazios ou número de itens existentes nesse buffer.

* 1. Heartbeat

Foi utilizado um protocolo heartbeat entre o Avião e o Controlador para determinar se o processo do Avião terminou de forma abrupta. O sinal de heartbeat é enviado do Avião para o Controlador de 3 em 3 segundos.

* 1. Critical Section

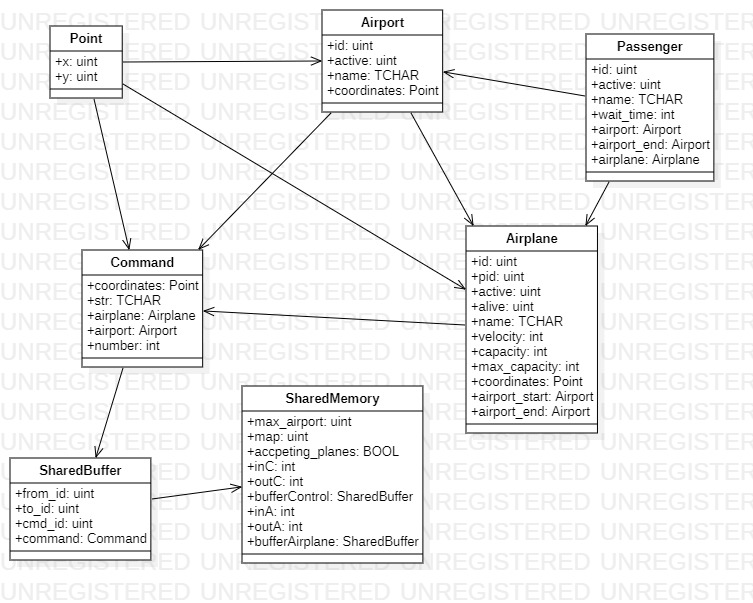
No lado do Controlador são usadas Critical sections para salvaguardar os dados relacionados com os Aviões, os Aeroportos e os Passageiros de serem corrompidos.

* 1. Eventos

Tanto o Controlador como o Avião usam eventos para terminar as threads auxiliares e de seguida terminar o processo de forma ordenada.

1. Estruturas de dados

Abaixo é possível ver o diagrama das estruturas de dados utilizadas no sistema e com as diversas ligações entre elas, de seguida irão ser descritas as várias estruturas..



* 1. Estrutura Airport

Esta estrutura representa um “Aeroporto”. Um “Aeroporto” é constituído por um id, um nome, coordenadas e ainda um identificador que identifica se está ativo ou não.

* 1. Estrutua Airplane

Esta estrutura representa um “Avião”. Um “Avião” é constituído por um id, um pid (process id), um nome, a velocidade(numero de posições por segundo), a capacidade, a capacidade total, as coordenadas, o aeroporto inicial, o aeroporto final, um identificador que identifica se está ativo ou não e ainda um outro identificador que identifica se está vivo ou não.

* 1. Estrutura Command

Esta estrutura representa uma union “Comando”. Um “Comando” é constituído por um dos seguintes elementos: coordenadas, cadeia de caracteres (string), um Avião, um Aeroporto ou um número. Esta union contém os dados que serão transferidos entre o controlador e o avião e vice-versa.

* 1. Estrutura SharedBuffer

Esta estrutura representa um item do buffer circular. Um item do buffer circular é constituído pelo id do recetor, pelo id do emissor, o código do comando e os dados do “Comando”. Esta estrutura irá ser transferida através da memória partilhada.

* 1. Estrutura SharedMemory

Esta estrutura representa a memória partilhada. A memória partilhada é constituída pelo número máximo de aeroportos, pelo mapa, por um índice de entrada e um de saída para o buffer circular do Controlador, por um índice de entrada e um de saída para o buffer circular dos Aviões.

1. Bibliotecas dinâmicas

A DLL fornecida pelos docentes da disciplina foi implementada de forma explicita. Foi ainda criada uma DLL com as várias estruturas e algumas macros que foram utilizadas tanto no Controlador como no Avião. Esta DLL foi implementada de forma implícita.

1. Conclusão

Com o desenvolvimento desta meta foi possível aprender muito sobre a API do Windows, nomeadamente a interação com semáforos, mutexes, threads, memória partilhada, eventos e secções criticas, também foi possível a aprendizagem do paradigma produtor/consumidor.

Durante o desenvolvimento desta meta foram surgindo problemas e desafios que foram superados com a ajuda dos professores da disciplina, os apontamentos por eles disponibilizados e da Internet.

Manual de utilização

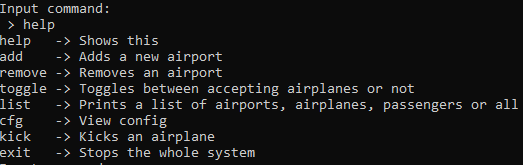
Controlador (Control)

Iniciar



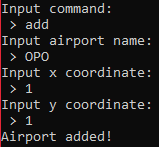
Para iniciar a aplicação “Controlador” basta clicar duas vezes no ficheiro “exe” ou executá-lo através da linha de comandos “pasta/Control.exe”.

Comando “help”

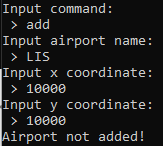


O comando “help” apresenta todos os comandos disponibilizados ao utilizador.

Comando “add”

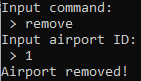


O comando “add” permite adicionar um novo aeroporto. A aplicação pedirá o nome do aeroporto, e as suas coordenadas. As coordenadas serão decrementadas por 1, ou seja, coordenadas (x, y): (1, 1) serão convertidas para (x, y): (0, 0).



A adição de um novo aeroporto falhará caso o nome do aeroporto já exista ou as coordenadas sejam inválidas.

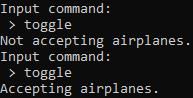
Comando “remove”



O comando “remove” permite remover um aeroporto. Para remover um aeroporto, o utilizador terá que introduzir o ID do aeroporto.

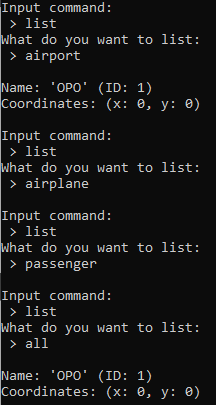
Este comando serve apenas para debug e deve ser usado com cuidado.

Comando “toggle”



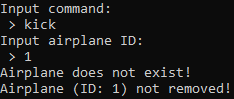
O comando “toggle” serve para começar ou parar de aceitar aviões.

Comando “list”



O comando “list” mostrará todos os aeroportos, aviões ou passageiros existentes no sistema. Poderá também ser listados todos ao mesmo tempo com a opção “all”.

Comando “kick”



O comando “kick” é apenas um comando de debug. Este comando enviará um sinal ao avião associado ao ID e terminará o avião.

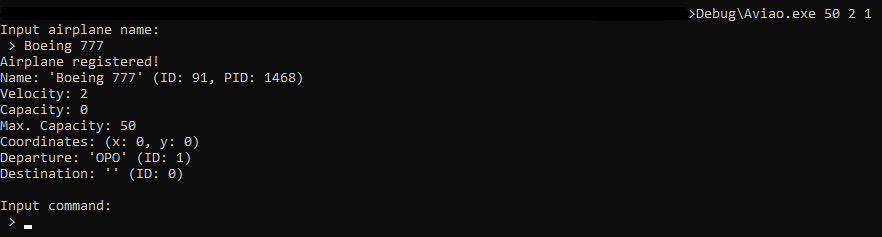
Comando “exit”



O comando “exit” terminará o sistema e todos os aviões e passageiros associados.

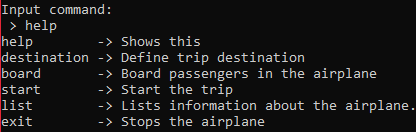
Avião (aviao)

Iniciar



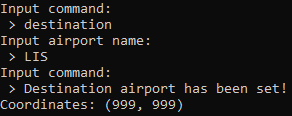
Para iniciar a aplicação “Aviao” terá que executá-lo através da linha de comandos “pasta/Aviao.exe [MAX\_CAPACITY] [VELOCITY] [AIRPORT\_ID]”. A primeira opção indica a capacidade máxima do avião, a segunda a velocidade (posições) por segundo e a terceira o ID do aeroporto inicial. O programa irá depois perguntar para introduzir um nome para que seja identificado.

Comando “help”

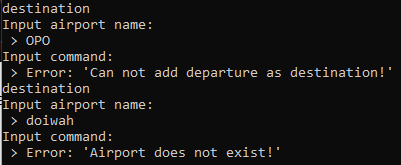


O comando “help” apresenta todos os comandos disponibilizados ao utilizador.

Comando “destination”



O comando “destination” permite ao utilizador introduzir um destino para o qual se deslocará.

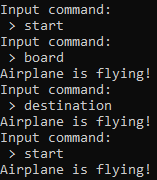


O comando falhará quando o utilizador introduzir o aeroporto de saída ou um aeroporto inexistente.

Comando “board”

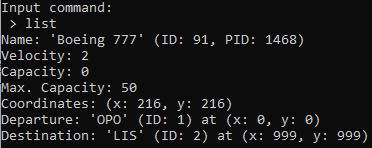
O comando “board” enviará sinal ao Controlador para avisar que está a aceitar passageiros neste momento.

Comando “start”



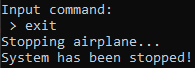
O comando “start” iniciará a viagem até ao destino. Enquanto o avião está a voar, não poderá aceder aos comandos: “board”, “destination” ou “start”.

Comando “list”



O comando “list” lista as definições do avião incluindo o nome, velocidade, capacidade atual, capacidade máxima, coordenadas atuais, aeroporto de saída e aeroporto de chegada.

Comando “exit”



O comando “exit” termina o programa avião.