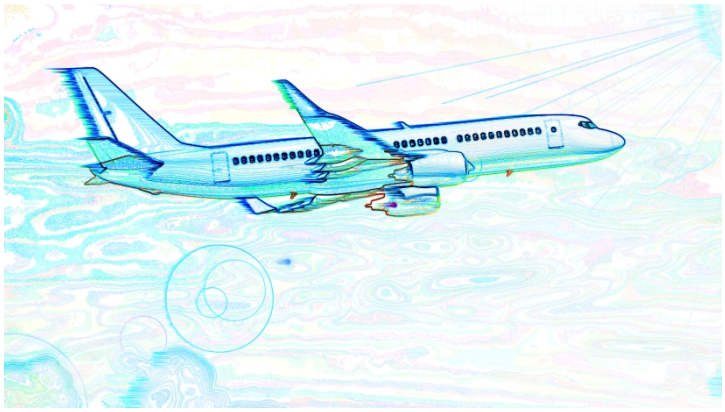
Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática
Sistemas Operativos 2



Leandro Adão Fidalgo | a2017017144
Pedro dos Santos Alves | a2019112789

Laboratório P5
Trabalho Prático 1
Meta 2

Coimbra, 13 de junho de 2021

Índice

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.

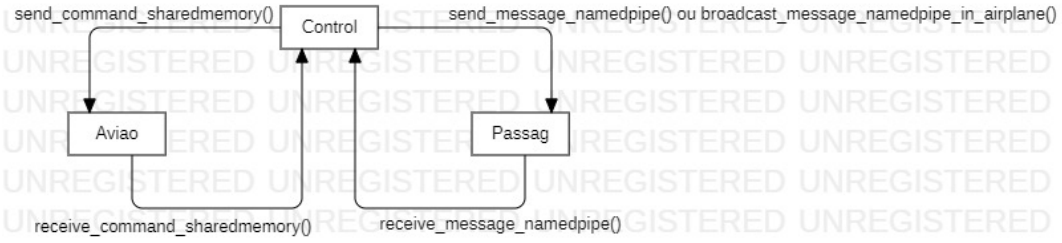
Na primeira meta do trabalho prático pretende-se que sejam feitas as seguintes funcionalidades: o Controlador aéreo (control), com uma interface do tipo consola, cria o(s) mecanismo(s) de comunicação e sincronização com os programas que representam os aviões. Atende até um máximo de aviões definido no 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.

Na segunda meta do trabalho prático pretende-se que seja desenvolvida uma interface gráfica para o Controlador e ainda sejam criados os Passageiros, os Passageiros podem esperar um tempo introduzido pelo utilizador e ainda pode esperar por um tempo infinito caso o utilizador não introduza nenhum tempo..

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.

2. Mecanismos de comunicação e sincronização



No diagrama acima apresentado é possível visualizar a maneira como o Controlador comunica entre o Aviao e o Passageiro. É possível constatar que o Avião não comunica com o Passageiro.

2. 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.

2. 2. 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.

2. 3. 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.

2. 4. 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.

2. 5. Eventos

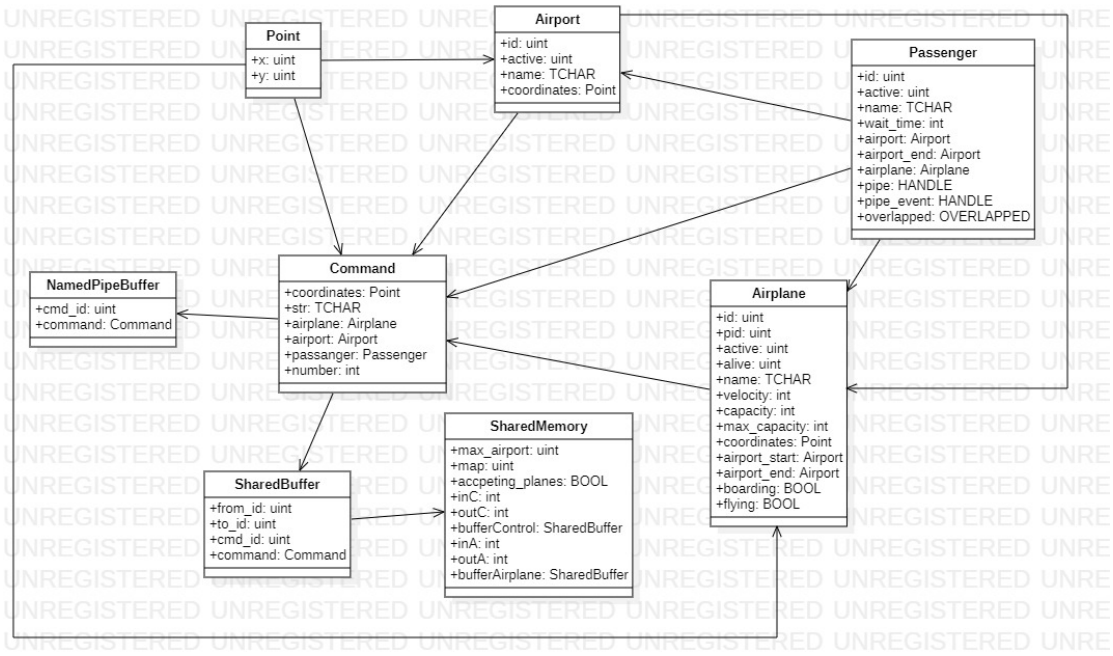
Tanto o Controlador como o Avião usam eventos para terminar as threads auxiliares e de seguida terminar o processo de forma ordenada.

2. 6. Pipes

Os pipes foram utilizados para a comunicação entre o Controlador e o Passageiro. O Controlador cria os pipes com a flag FILE_FLAG_OVERLAPPED.

3. 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.



3. 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.

3. 2. Estrutura 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.

3. 3. 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.

3. 4. 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.

3. 5. 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.

3. 6. Estrutura NamedPipeBuffer

Esta estrutura representa as mensagens que serão enviadas pelo named pipe. O NamedPipeBuffer é constituído por um ID de comando e por uma estrutura do tipo Command.

4. 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 e no Passageiro. Esta DLL foi implementada de forma implícita.

5. Decisões tomadas na implementação

5. 1. Triple Buffering

Para que a interface não cintile foi utilizado um buffer triplo. Não foram utilizados dois “back buffers” que alternam entre si para enviar para o “front buffer” que envia para o monitor. A técnica de triple buffering foi implementada de forma a que um “back buffer” (double_dc) envia para um “middle buffer” (triple_dc) e que por último envia para o “front buffer” (hdc).

6. Funcionalidades

ID	Descrição funcionalidade / requisito	Estado
1	Instância única (Controlador)	Implementado
2	Gestão de aeroportos (Controlador)	Implementado
3	Gestão do espaço aéreo (Controlador)	Implementado
4	Registry (Controlador)	Implementado
5	Interface gráfica Win32 (Controlador)	Implementado
6	Encerrar todo o sistema (Controlador)	Implementado
7	Suspender/ativar a aceitação de novos aviões (Controlador)	Implementado
8	Listar todos os aeroportos, aviões e passageiros (Controlador)	Implementado
9	Definir o próximo destino (Avião)	Implementado
10	Embarcar passageiros (Avião)	Implementado
11	Iniciar viagem (Avião)	Implementado
12	Terminar o programa a qualquer altura (Avião)	Implementado
13	Implementação DLL implícita (Utils, criada pelos alunos)	Implementado
14	Implementação DLL explícita (Fornecida pelos professores)	Implementado
15	Colisão entre aviões (Avião, estratégia: aguardar)	Implementado
16	Heartbeat 3 segundos (Avião)	Implementado
17	Memória partilhada (Controlador/Avião)	Implementado
18	Paradigma Produtor/Consumidor (Controlador/Avião)	Implementado
19	Aguardar que exista um avião disponível (Passageiro)	Implementado
20	Desistência automática do Passageiro (Passageiro)	Implementado
21	Named pipe (Controlador/Passageiro)	Implementado
22	Técnica de triple buffering (Controlador)	Implementado
23	Mouseover do rato em cima de um avião (Controlador)	Implementado
24	Click do rato em cima de um aeroporto (Controlador)	Implementado
25	Programação genérica de caracteres Char/Unicode	Implementado
26	Threads (Controlador: 5+, Avião: 3 ou 4, Passageiro: 2 ou 3)	Implementado
27	Mutexes (Controlador: 4, Avião: 3)	Implementado
28	Semáforos (Controlador: 5, Avião: 4, Passageiro: 1)	Implementado
29	Eventos (Controlador: 3+, Avião: 2, Passageiro: 4)	Implementado
30	Critical Section (Controlador: 3)	Implementado

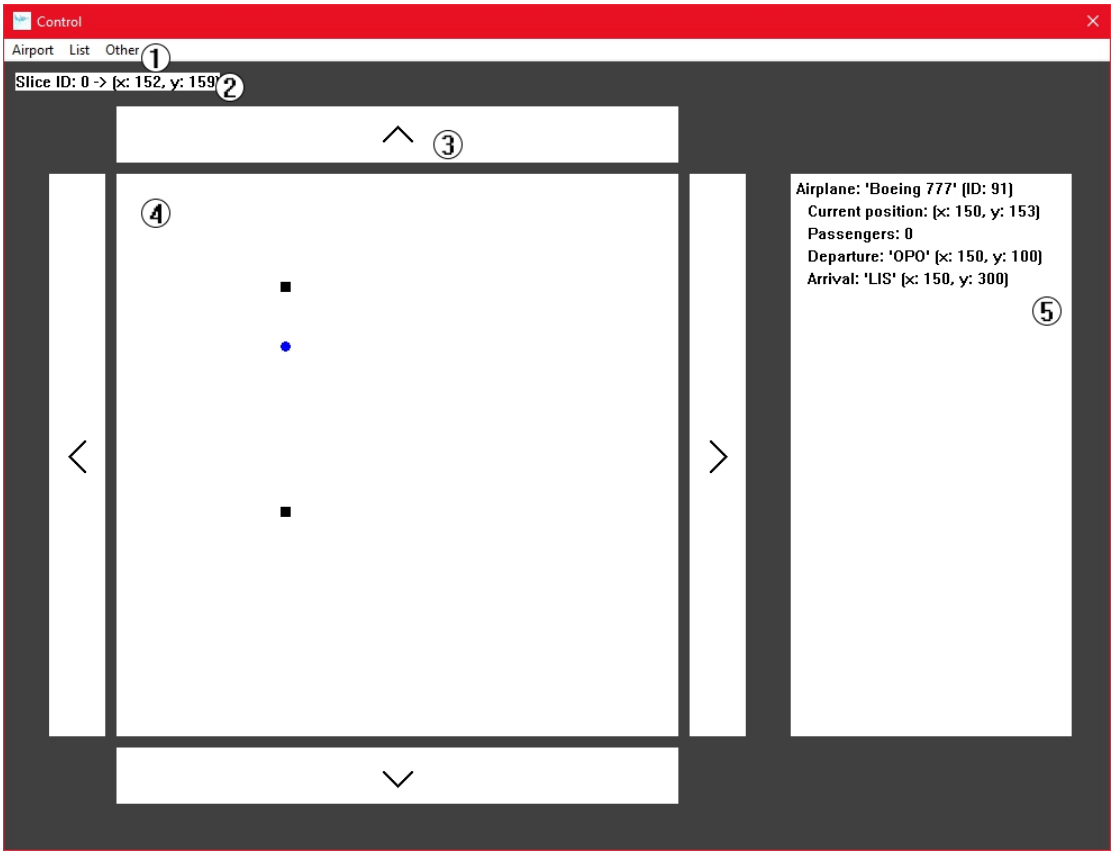
7. Conclusão

Com o desenvolvimento desta meta foi possível aprender muito sobre a API do Windows, nomeadamente a interação com semáforos, mutexes, threads, memória partilhada, eventos, secções criticas e named pipes, 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)



1 - Menu

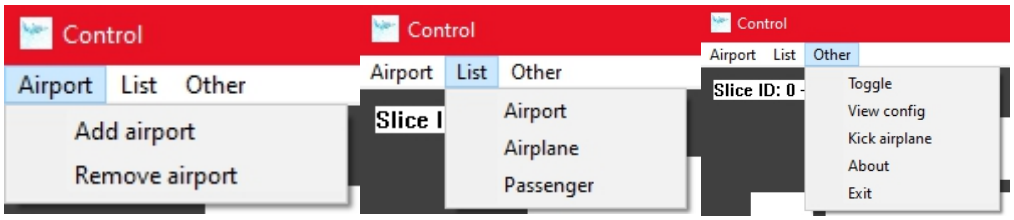
2 - Quadrante e coordenadas
- Indicação de que parte do mapa está a ser visto e as coordenadas do mapa onde o rato se encontra.

3 - Setas
- Setas para mudar a parte do mapa que está visível.

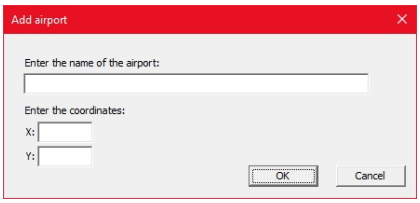
4 - Mapa
- Mapa do espaço aéreo.
- Legenda:
  - Quadrado preto: Aeroporto.
  - Circulo azul: Avião.

5 - Informações adicionais
- Informações adicionais acerca do avião, caso haja *mouseover* num circulo azul, ou aeroporto, caso haja um clique do rato num quadrado preto.

Menu

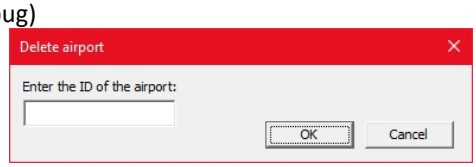


1 - Add airport



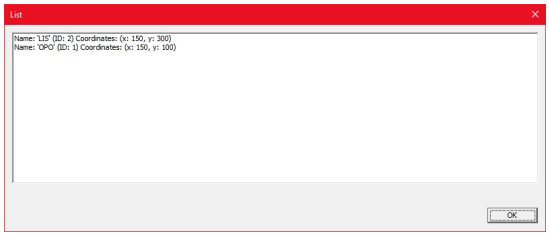
- Adicionar um aeroporto introduzindo um nome e as coordenadas.

2 - Remove airport (Debug)



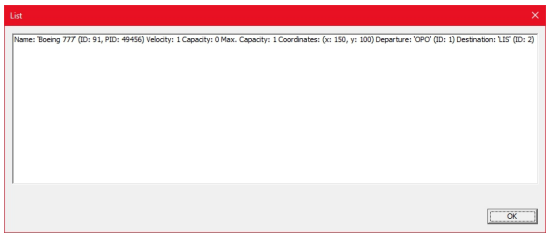
- Remover um aeroporto introduzindo o seu ID.

3 - List airport



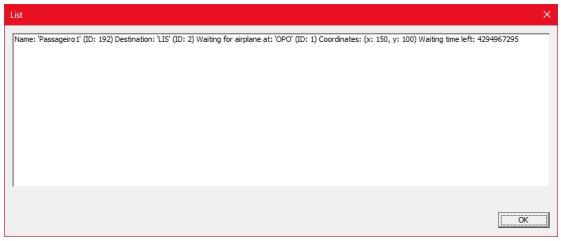
- Listagem de aeroportos registados no sistema.

4 - List airplane



- Listagem de aviões registados no sistema.

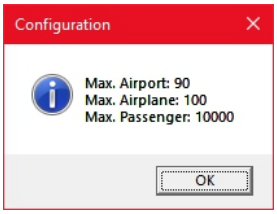
5 - List passenger



- Listagem de passageiros registados no sistema.

6 - Toggle
- Suspender/ativar a aceitação de novos aviões.

7 - View config (Debug)



- Listagem de alguns dados para fins de debugging.

8 - Kick airplane (Debug)

- Remover avião do sistema. (Útil caso haja dois aviões presos a aguardar para evitar colisões)

9 - About

- Informação sobre o grupo que realizou o projeto.

10 - Exit
- Termina a aplicação e o sistema todo.

Avião (aviao)

Iniciar

```
>Debug\Aviao.exe 50 2 1
Input airplane name:
 > Boeing 777
Airplane registered!
Name: 'Boeing 777' (ID: 91, PID: 1468)
Velocity: 2
Capacity: 0
Max. Capacity: 50
Coordinates: (x: 0, y: 0)
Departure: 'OPO' (ID: 1)
Destination: '' (ID: 0)

Input command:
 > _
```

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"

```
Input command:
 > help
help        -> Shows this
destination -> Define trip destination
board       -> Board passengers in the airplane
start       -> Start the trip
list        -> Lists information about the airplane.
exit        -> Stops the airplane
```

O comando "help" apresenta todos os comandos disponibilizados ao utilizador.

Comando "destination"

```
Input command:
 > destination
Input airport name:
 > LIS
Input command:
 > Destination airport has been set!
Coordinates: (999, 999)
```

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

```
destination
Input airport name:
 > OPO
Input command:
 > Error: 'Can not add departure as destination!'
destination
Input airport name:
 > doiwah
Input command:
 > Error: 'Airport does not exist!'
```

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"

```
Input command:
 > start
Input command:
 > board
Airplane is flying!
Input command:
 > destination
Airplane is flying!
Input command:
 > start
Airplane is flying!
```

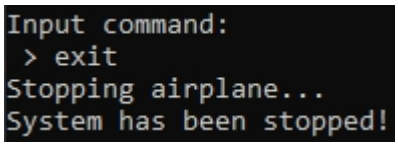
O comando "start" iniciará a viagem até ao destino. Enquanto o avião está a voar, não poderá aceder aos comandos: "board", "destination" ou "start".

Comando "list"

```
Input command:
 > list
Name: 'Boeing 777' (ID: 91, PID: 1468)
Velocity: 2
Capacity: 0
Max. Capacity: 50
Coordinates: (x: 216, y: 216)
Departure: 'OPO' (ID: 1) at (x: 0, y: 0)
Destination: 'LIS' (ID: 2) at (x: 999, y: 999)
```

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.

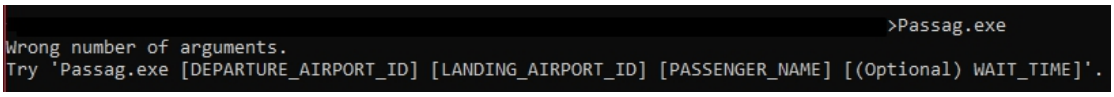
Comando “exit”



O comando “exit” termina o programa avião.

Passageiro (passag)

Iniciar



Para iniciar a aplicação “Passag” terá que executá-lo através da linha de comandos “pasta/Passag.exe [DEPARTURE_AIRPORT_ID] [LANDING_AIRPORT_ID] [PASSENGER_NAME] [(Optional) WAIT_TIME]”. A primeira opção indica o ID do aeroporto de saída, a segunda o ID do aeroporto de chegada, a terceira o nome do passageiro e a última o tempo de espera em segundos, esta última opção pode ser omitida deixando um tempo de espera infinito. Caso um tempo de espera seja colocado, passado o número de segundos introduzidos se não tiver embarcado num avião o programa terminará.

Comando “exit”

O comando “exit” termina o programa passageiro.