

TRABALHO PRÁTICO Nº 2**Simulação de um sistema de reserva de lugares**
Implementação de uma arquitetura cliente/servidor baseada em *FIFOs***Sumário**

Pretende-se desenvolver uma aplicação cliente/servidor que permita efetuar reservas de lugares para um evento. A simulação será feita através de processos a correr num único computador, sendo a comunicação entre clientes e servidor feita através de *pipes* com nome (*FIFOs*).

Objetivos

Familiarizar os estudantes com a programação de sistema, em ambiente Unix/Linux, envolvendo a gestão de processos/*threads* e a utilização de mecanismos de comunicação e de sincronização entre processos/*threads*.

Especificação

A aplicação a desenvolver será constituída por um programa servidor (**server**) que processa pedidos provenientes dos clientes e por um programa cliente (**client**) que simula o pedido de um utilizador. Cada pedido contém os seguintes dados: a quantidade de lugares pretendidos e uma lista de identificadores de lugares preferidos; esta lista poderá ter um número de elementos igual ou superior à quantidade de lugares pretendidos, mas limitado. O servidor tentará reservar os lugares, tendo em conta as preferências do cliente, e dará uma resposta adequada ao cliente, indicando os lugares reservados ou, caso não tenha sido possível fazer a reserva, o motivo da impossibilidade.

O servidor será constituído por um conjunto de "bilheteiras", simuladas por *threads*. Cada pedido de um cliente será encaminhado para uma bilheteira livre que se encarregará de tentar fazer a reserva e dará uma resposta ao cliente. Recorrendo a mecanismos de sincronização, devem ser tomadas as providências necessárias para que um lugar não seja atribuído a mais do que um cliente, evitando a contenção na execução das operações de reserva.

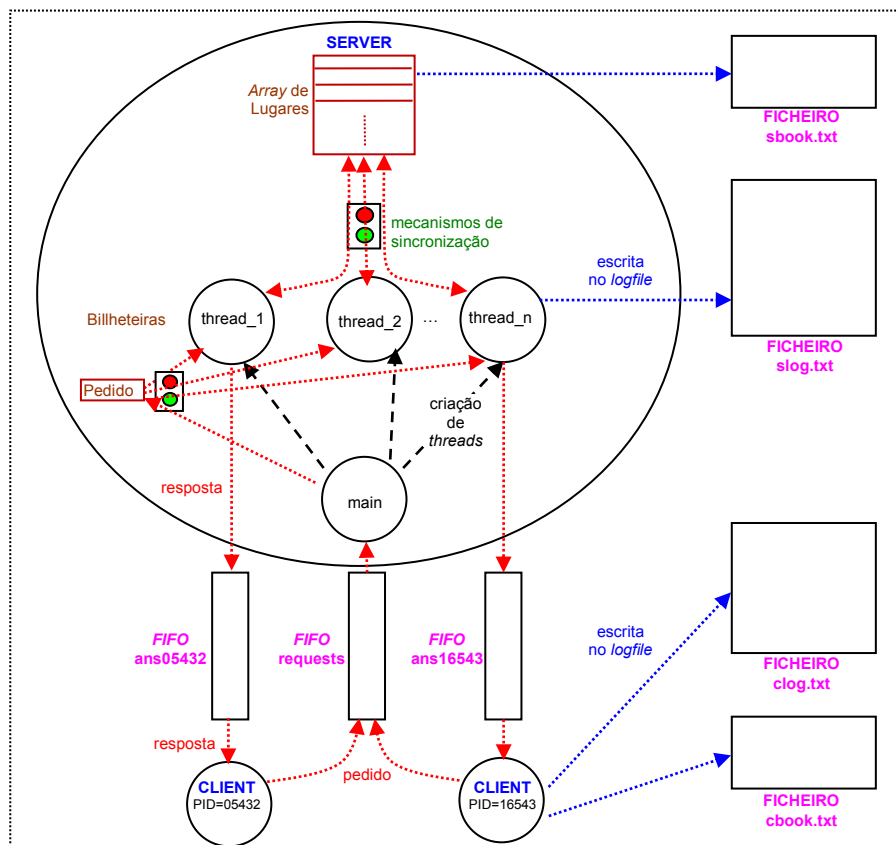
A comunicação entre clientes e servidor será feita através de *FIFOs*. O servidor receberá os pedidos dos clientes através de um *FIFO* comum, de nome **requests**, e cada cliente receberá a resposta do servidor através de um *FIFO* dedicado, de nome **ansXXXXXX**, em que **XXXXXX** representa o *PID* do cliente (ver figura).

Todas as operações realizadas pelo servidor e pelos clientes devem ser registadas em dois ficheiros de texto, **slog.txt** e **clog.txt**, respetivamente. O formato destes ficheiros será especificado adiante. Além disso, o servidor e os clientes devem registar noutros dois ficheiros de texto, **sbook.txt** e **cbook.txt** (este partilhado por todos os clientes), os números dos lugares que foram reservados (apenas os números).

Descrevem-se a seguir mais detalhadamente os programas a desenvolver e alguns aspetos do seu funcionamento e implementação.

• Programa server

- Aceita e executa os pedidos de reserva de bilhetes que lhe são enviados pelos clientes (instâncias de **client**), dando uma resposta adequada a cada cliente.
- Recebe como argumentos da linha de comandos o número de lugares disponíveis para o evento (**num_room_seats**), o número bilheteiras para atender clientes (**num_ticket_offices**) e o tempo de funcionamento das bilheteiras (**open_time**, expresso em segundos):
 - **\$ server <num_room_seats> <num_ticket_offices> <open_time>**
 - Exemplo: **server 1000 3 200**



Arquitetura geral da aplicação a desenvolver.

- Cria o *FIFO*, **requests**, de uso comum, através do qual recebe os pedidos de reserva de lugares; as respostas aos pedidos são dadas através de *FIFOs* dedicados, criados pelos clientes (ver adiante).
- Cria **num_ticket_offices threads** auxiliares – *threads* bilheteira – que tentam fazer uma reserva do seguinte modo: a reserva deve ser feita incrementalmente (um lugar de cada vez), tendo em conta a ordem das preferências, até se concluir se é possível ou não reservar o número de lugares pretendidos; se não for possível reservar todos os lugares pretendidos, os lugares reservados tentativamente devem ser libertados. Enquanto um *thread* está a tentar reservar um lugar, outros devem poder estar a tentar reservar outros lugares.
- O *main thread* deve estar permanentemente à escuta dos pedidos enviados pelos clientes, no *FIFO requests*, colocando-os num *buffer* de tamanho unitário, onde são recolhidos pelos *threads* bilheteira (ver figura).
- Para simplificação, considera-se que os identificadores dos lugares são números inteiros, tomando valores na gama [1..num_room_seats]. Cada pedido será constituído por uma sequência de números inteiros, em que o primeiro número é o identificador do cliente (para simplificar, poderá ser o *PID* do processo cliente), o segundo número é a quantidade de lugares pretendidos e os restantes números são os identificadores desses lugares (em número igual ou superior ao número de lugares pretendidos). Em cada pedido, o número máximo de lugares que é possível reservar é **MAX_CLI_SEATS** (uma constante global da aplicação, definida numa *header file*) e o número máximo de lugares pretendidos é também **MAX_CLI_SEATS**. Exemplos de pedidos válidos, num caso em que **MAX_CLI_SEATS** toma o valor 5:
 - 12345 1 112 13 1313 (o cliente 12345 pretende 1 lugar, entre os lugares com identificadores 112, 13 ou 1313)
 - 13579 3 11 12 13 14 15 (o cliente 13579 pretende 3 lugares, entre os lugares com identificadores entre 11 e 15)
 - 24680 5 101 102 103 104 105 (o cliente 24680 pretende 5 lugares, com identificadores entre 101 e 105)
- Cada *thread* bilheteira deve validar o pedido antes de o executar. A validação consiste em verificar se a quantidade de lugares a reservar, **num_wanted_seats**, está na gama [1..MAX_CLI_SEATS], se

o número de lugares preferidos está na gama `[num_wanted_seats..MAX_CLI_SEATS]`, e se o identificador dos lugares preferidos está na gama `[1..num_room_seats]`. Se o pedido for válido, o *thread* executá-lo-á.

- Será sempre enviada uma resposta adequada ao cliente, com exceção da situação identificada na descrição do funcionamento do programa `client` (ver adiante).

Se os parâmetros do pedido forem inválidos ou se não tiver sido possível satisfazer o pedido, a resposta será um número inteiro negativo, com um dos seguintes valores e significados:

- -1 – a quantidade de lugares pretendidos é superior ao máximo permitido (**MAX_CLI_SEATS**)
- -2 – o número de identificadores dos lugares pretendidos não é válido
- -3 – os identificadores dos lugares pretendidos não são válidos
- -4 – outros erros nos parâmetros
- -5 – pelo menos um dos lugares pretendidos não está disponível
- -6 – sala cheia

Se tiver sido possível satisfazer o pedido, a resposta será constituída por uma sequência de números positivos em que o primeiro indica o número de lugares reservados e os seguintes são os identificadores desses lugares. Por exemplo, a sequência `3 12 13 14`, significa que foram reservados 3 lugares com os identificadores 12, 13 e 14.

- Os *threads* bilheteira devem invocar 3 funções com nomes bem determinados, cujos protótipos se indicam a seguir, onde o parâmetro `seats` é um apontador para o início do *array* de lugares, partilhado por todos os *threads*, onde são registados os lugares livres/ocupados:

- `int isSeatFree(Seat *seats, int seatNum)` – testa se o lugar `seatNum` está livre
- `void bookSeat(Seat *seats, int seatNum, int clientId)` – reserva o lugar `seatNum` para o cliente cujo identificador é `clientId`.
- `void freeSeat(Seat *seats, int seatNum)` – liberta o lugar `seatNum`, após uma tentativa de reserva sem sucesso, em que foram pré-reservados alguns lugares mas não foi possível reservar todos os lugares pretendidos.

Cada uma destas funções, após a operação de leitura/escrita do conteúdo de `seats`, deve invocar, dentro na [secção crítica respetiva](#), uma *function-like macro*, `DELAY()`, definida usando a diretiva: `#define DELAY()`. Esta *macro* poderá ser usada para simular a existência de alguma demora na execução das operações.

- Quando expirar o tempo de funcionamento das bilheteiras, o *main thread* deve fechar o *FIFO requests*, informar aos restantes *threads* que devem terminar, e aguardar que eles terminem.
- Cada um dos *threads*, regista no ficheiro `slog.txt` (*server logging*), a abertura e fecho das bilheteiras, e os pedidos e as respostas dadas. Cada linha do ficheiro deverá ter um dos seguintes formatos:

- `TO-OPEN` – assinala a abertura da bilheteira com o número `TO`
- `TO-CLOSED` – assinala o fecho da bilheteira com o número `TO`
- `TO-CLIID-NT: aaaa bbbb cccc dddd ... - AAAA BBBB CCCC DDDD ...` – contém informação sobre um pedido que foi atendido com sucesso: qual a bilheteira onde foi atendido (`TO`), o identificador do cliente (`CLIID`), o número de lugares a reservar (`NT`), os identificadores dos lugares preferidos (`aaaa bbbb ...`) e os identificadores dos lugares efetivamente reservados (`AAAA BBBB ...`). A largura total dos campos `aaaa bbbb ...` e `AAAA BBBB ...` dependerá do valor de **MAX_CLI_SEATS**.

- `TO-CLIID-NT: aaaa bbbb cccc dddd ... - XXX` – contém informação sobre uma reserva que não foi atendida; todos os campos têm o significado indicado anteriormente e `XXX` é um código com 3 letras que indica o motivo pelo qual a reserva não foi efetuada.

Os valores de `XXX` e respetivos significados têm a seguinte correspondência com o código anteriormente referido (inteiro negativo) enviado aos clientes, indicativo de que não foi possível atender o pedido: -1→ **MAX**, -2→ **NST**, -3→ **IID**, -4→ **ERR**, -5→ **NAV** e -6→ **FUL**.

Os campos têm uma largura fixa e estão separados entre si por 1 carácter espaço (ver exemplo). Todos os valores numéricos que não ocupem a totalidade da largura do campo, devem ser preenchidos com zeros no início.

- Apresenta-se a seguir um exemplo do conteúdo de `slog.txt`:

Conteúdo de <code>slog.txt</code> 123456789012345678901234567890123456789012345678901234567890	Comentários das linhas a negrito (não incluídos no ficheiro)
01-OPEN	→ bilheteira 1 aberta
03-OPEN	...
02-OPEN	...
02-12345-01: 0010 0011 0012 0013 - 0010	→ bilheteira 02 atende
01-23456-02: 0010 0011 - IID	cliente 12345 que pretende
02-45678-02: 0020 0021 0022 0023 - 0021 0023	1 lugar entre 0010 e 0013;
03-45678-04: 0001 0002 0003 0004 - 0001 0002 0003 0004	foi atribuído o lugar 0010
01-45678-01: 0030 0031 0032 0033 0034 - 0033	...
02-45678-02: 0025 0026 - 0025 0026	...
03-45678-03: 0040 0041 0042 0043 0044 - 0040 0043 0044	...
...	...
01-23456-02: 0010 0011 - FUL	→ bilheteira 01 atende
01-CLOSED	cliente 23456 que pretende
02-CLOSED	2 lugares ...; sala cheia
03-CLOSED	...
SERVER CLOSED	

- Antes de terminar, deve guardar no ficheiro `sbook.txt` (*server bookings*) os números inteiros identificadores dos lugares que foram reservados (apenas estes), um por cada linha de texto e com zeros no início, sempre que o número não ocupar `WIDTH_SEAT` (= 4) carateres (ex: 0013).

• Programa `client`

- Recebe como argumentos da linha de comandos o tempo máximo que o cliente pode esperar por uma resposta (`time_out`), o número de lugares que o cliente pretende reservar (`num_wanted_seats`) e os identificadores dos lugares preferidos (`pref_seat_list`):
 - `$ client <time_out> <num_wanted_seats> <pref_seat_list>`
 - Exemplo: `client 120 3 "11 12 13 14 15"`
- Começa por criar um *FIFO* dedicado através do qual receberá as respostas aos pedidos de reserva que enviar ao servidor, através do *FIFO requests*. Esse *FIFO* deve ter o nome `ansXXXXXX`, em que `XXXXXX` representa o *PID* do cliente.
- Envia ao servidor, através do *FIFO requests*, um pedido de reserva com os seguintes dados: *PID* do cliente, número de lugares a reservar e lista de lugares preferidos.
- Aguarda, no *FIFO ansXXXXXX*, pela resposta do servidor que deve indicar a concretização (ou não) da reserva, de acordo com o que foi especificado anteriormente. Se a resposta não for recebida dentro do tempo `time_out` o cliente terminará a execução.
- O resultado dos pedidos de reserva efetuados deve ser registado, pelos clientes, num ficheiro partilhado por todos eles, `clog.txt`, no formato que se indica a seguir:
 - Cada linha do ficheiro é constituída por 2 ou 3 campos, dependendo da resposta recebida do servidor:
 - O primeiro é o *PID* do cliente.
 - O segundo campo tem um conteúdo variável consoante a resposta.
 - Se a reserva tiver sido bem sucedida, esse campo terá o formato `XX.NN` em que `NN` indica o número de lugares pretendidos (e reservados) e `XX` é um número que varia entre 01 e `NN`, por cada lugar reservado (exemplo: 01.03, 02.03 e 03.03, para enumerar cada um dos lugares de uma reserva bem sucedida de 3 lugares).
 - Se a reserva não tiver sido bem sucedida, este campo terá o formato `XXX`, em que `XXX` tem a correspondência que foi indicada anteriormente com os códigos negativos retornados pelo servidor, quando não é possível satisfazer o pedido: -1→ `MAX`, -2→ `NST`, -3→ `IID`, -4→ `ERR`, -5→ `NAV` e -6→ `FUL`. Se foi excedido o tempo máximo de espera do cliente, este campo deve ter o conteúdo `OUT`.
 - O terceiro campo só será preenchido nos casos em que a reserva teve sucesso, indicando o identificador do lugar correspondente a cada item `XX.NN`.
 - Os campos têm uma largura fixa (1º campo = `WIDTH_PID` = 5, 2º campo = `WIDTH_XXNN` = 5, 3º campo = `WIDTH_SEAT` = 4) e estão separados entre si por 1 carácter espaço. Os valores neles

escritos devem ser ajustados à esquerda do campo. Todos os valores numéricos que não ocupem toda a largura do campo, devem ser preenchidos com zeros no início.

- Apresenta-se a seguir um exemplo do conteúdo do ficheiro `clog.txt`:

Conteúdo de <code>clog.txt</code> 12345678901234567	Comentários (não incluídos no ficheiro)
05432 01.02 0317	o lugar nº 1 de 2 reservados para cliente 5432 é o lugar 317
05432 02.02 0318	o lugar nº 2 de 2 reservados para cliente 5432 é o lugar 318
...	...
09753 MAX	o número de lugares pretendidos é superior ao máximo/cliente
98765 NUM	o número de identificadores dos lugares pretendidos é inválido
10586 IID	os identificadores dos lugares pretendidos não são válidos
...	...
15792 01.01 1023	o lugar nº 1 de 1 reservado para o cliente 15792 é o lugar 1023
01234 OUT	esgotou o tempo máximo de espera do cliente 1234
16543 FUL	sala cheia

- Devem também ser guardados no ficheiro `cbook.txt` (*clients bookings*) os números inteiros identificadores dos lugares (apenas estes) que foram reservados para todos os clientes, um por cada linha de texto e com zeros no início, sempre que o número não ocupar `WIDTH_SEAT` (= 4) caracteres (ex: 0317).

Notas sobre o desenvolvimento

- Os nomes dos programas, a ordem dos argumentos da linha de comando, os nomes indicados para algumas funções e constantes, os nomes e a estrutura dos ficheiros contendo resultados devem ser respeitados escrupulosamente. **O incumprimento pode conduzir à não avaliação do trabalho**, com as inerentes consequências.
- Todos os parâmetros internos devem ser facilmente modificáveis (com recurso a diretivas `#define` ou constantes com nome).
- Devem ser definidas pelo menos as seguintes constantes (=valor), referidas ao longo do texto: `MAX_ROOM_SEATS` (=9999), `MAX_CLI_SEATS` (=99), `WIDTH_PID` (=5), `WIDTH_XXNN` (=5), `WIDTH_SEAT` (=4).
- É necessário estabelecer um protocolo e formato de mensagens a trocar entre clientes e servidor, tendo em conta as especificações fornecidas anteriormente.
- Devem ser previstos mecanismos de sincronização entre os *threads* do servidor de modo a evitar erros nas operações que lhe são solicitadas (por exemplo, reserva de lugares).
- Todas as estruturas de comunicação e sincronização devem ser destruídas no final da execução dos programas. Tudo o que não estiver especificado e que não venha a ser especificado numa nova versão do trabalho (a publicar, se necessário) poderá ser especificado pelos elementos do grupo de trabalho, devendo as especificações adicionais, contidas num ficheiro com o nome `addspecs.pdf`, ser enviadas juntamente com o código.

Validação da solução

- Para validar a solução desenvolvida serão realizados testes automáticos com o auxílio de um programa validador e ficheiros de testes.

Realização e entrega do trabalho

- Deverá ser elaborado um pequeno relatório (2 páginas) onde se descreva a estrutura das mensagens trocadas entre clientes e servidor (e vice-versa), os mecanismos de sincronização utilizados e a forma como é feito o encerramento do servidor (focar apenas estes aspetos). Sempre que necessário, a descrição deve ser acompanhada por excertos de código. O relatório, com o nome `report.pdf`, deve ser enviado juntamente com o código.
- A **data limite** para a entrega do trabalho é **14/05/2018, às 13:00h**.
- Oportunamente serão dadas outras informações sobre a realização e entrega do trabalho, nomeadamente, quanto à estrutura de diretórios a utilizar.