Programação Concorrente

2023/2024

Projeto – Parte B

Na segunda parte do projeto os alunos implementarão duas novas verões do **old-photo-serial.c** usando *pipe* para efetuar a comunicação entre *threads*. Os alunos deverão implementar as seguintes aplicações paralelas:

- **old-photo-paralelo-B** - aplicação que paraleliza o processamento das imagens de uma forma semelhante à do **old-photo-paralelo-A**, mas usa *pipes* para distribuição dos trabalhos.
- **old-photo-pipeline** – aplicação que implementa um *pipeline* com vários estágios, cada um responsável por uma das quartos transformações do **old-photo-paralelo-A**.

1 old-photo-paralelo-B

Nesta parte do projeto os alunos deverão implementar uma aplicação paralela (chamada old-photo-paralelo-B) que processa imagens tal como na Parte A do projeto, mas usando um *pipe* para efetuar a distribuição das imagens pelas *threads*.

A aplicação deverá aplicar as 4 transformações a cada uma das imagens armazenadas numa diretoria de modo a produzir as imagens com efeito antigo, mas de modo a acelerar o processo e reduzir o tempo de processamento deverão ser usadas *threads*, tal como na primeira parte do projeto.

Este programe tem um *pipe*, através do qual o **main()** envia as referências para as imagens a serem processadas, que deverão ser lidas de forma concorrente pelas várias *threads* anteriormente criadas. Cada *thread* lê do *pipe* a referência para a próxima imagem a ser processada e processa-a num modo semelhante ao do **old-**

photo-paralelo-B, i.e. cada *thread* efetua as 4 transformaçoes e grava o resultado no disco, e volta a ler do *pipe* a referência para a próxima imagem a ser processada.

A aplicação paralela a desenvolver (chamada **old-photo-paralelo-B**) tem apenas um tipo de *threads*. Cada uma destas processa um sub-conjunto disjunto de imagens e para cada uma das imagens a si atribuídas gera a imagem envelhecida. O número de

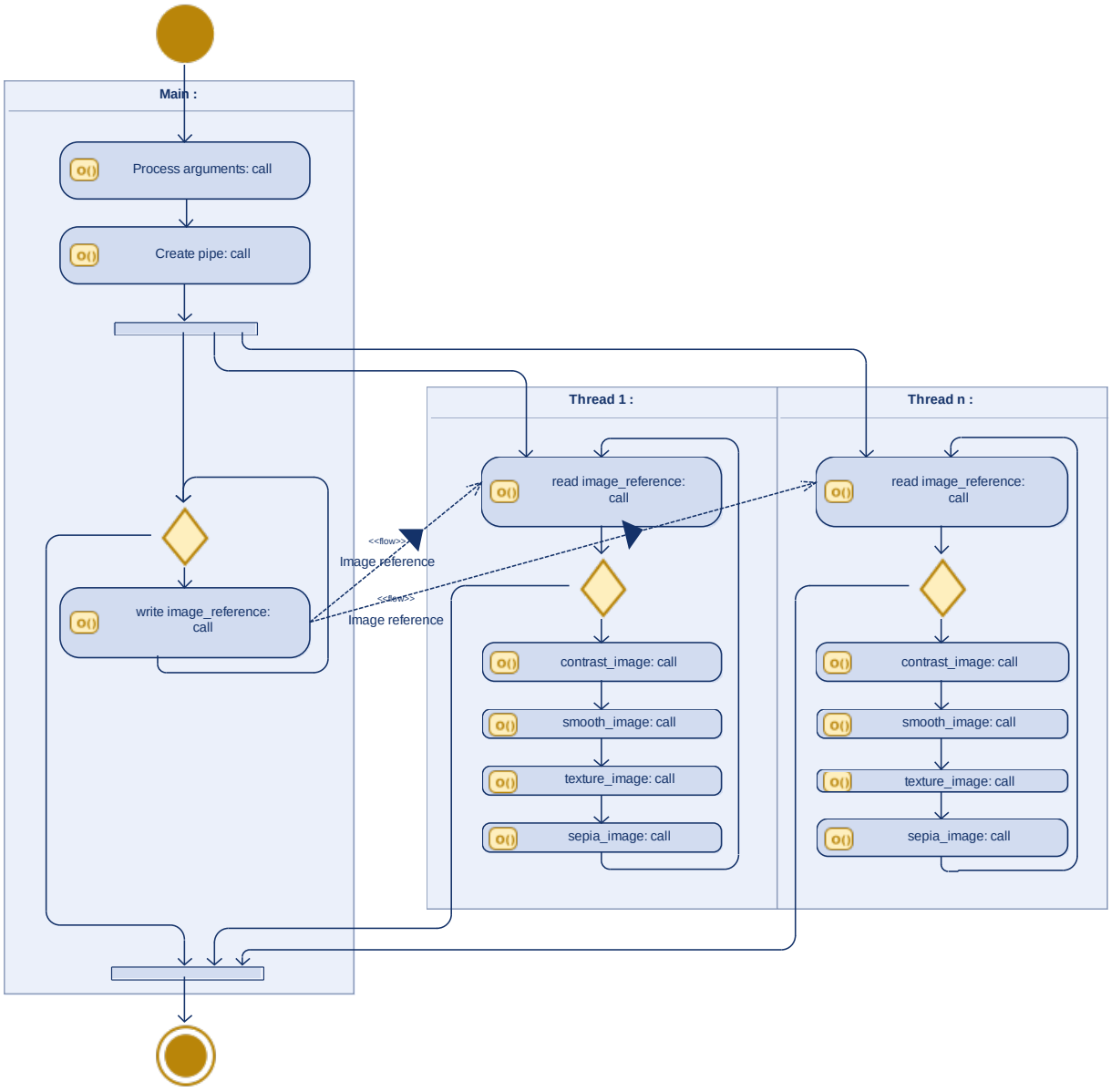


*Figura 1: Paralelização do **old-photo-paralelo-B***

threads criadas é definido pelo utilizador através de um argumento da linha de comando.

Como ilustra a Figura 1, o **main()** processa os argumentos, cria um *pipe*, cria um terminado número de *threads* (definido pelo utilizador) e depois envia através desse *pipe* todas as referência das imagens. Cada *thread* lê sequencialmente do *pipe* as referências para as imagens e aplica-lhes as quatro transformação.

Depois de enviar as referências para todas as imagens através do *pipe*, o **main()** deverá esperar pela terminação da *threads*.

As *threads* deverão terminar a sua execução quando não houver mais imagens para serem processadas.

2 old-photo-pipeline

Nesta parte do projeto os alunos deverão implementar uma aplicação paralela (chamada **old-photo-pipeline**) que processa imagens, tal como na Parte A do projeto, mas implementando um *pipeline*, com uma única transformação em cada um dos estágios desse *pipeline*.

A aplicação deverá efetuar as 4 transformações a cada uma das imagens armazenadas numa diretoria de modo a produzir as imagens com efeito antigo, mas de modo a acelerar o processo e reduzir o tempo de processamento deverá ser usado um *pipeline* com diversos estágios paralelos.

Cada um dos estágios, corresponde uma *thread* criada pelo **main()** no início do programa, que espera do estágio anterior (ou **main()**) o envio da referência para a próxima imagem a ser processada. Todos os envios de referências de imagens entre estágios (identificados nas images por **«flow» Image reference**) são realizados através dos vários *pipes* (um para cada estágio).

O **main()** envia través de um *pipe* específico a referência para as imagens a serem processadas no primeiro estágio, este estágio depois de aplicar a transformação envia a referência da imagem para o segundo estágio, voltando a ler uma referência enviada pelo **main()**. Os estágios seguintes têm uma lógica semelhante: cada um dos estágios lê as referências de um *pipe*, processam as imagens e escrevem a referência par o *pipe* seguinte. O último estágio deverá gravar a imagem no disco.

A aplicação paralela a desenvolver (chamada **old-photo-pipeline**) tem tantos tipos de *threads* como estágios do *pipeline*. Cada *thread*/estágio aplica apenas uma das transformações a todas as imagens. O programa deverá criar apenas um *pipeline*, pelo que apenas deverá ser criada uma *thread* de cada tipo/estágio.

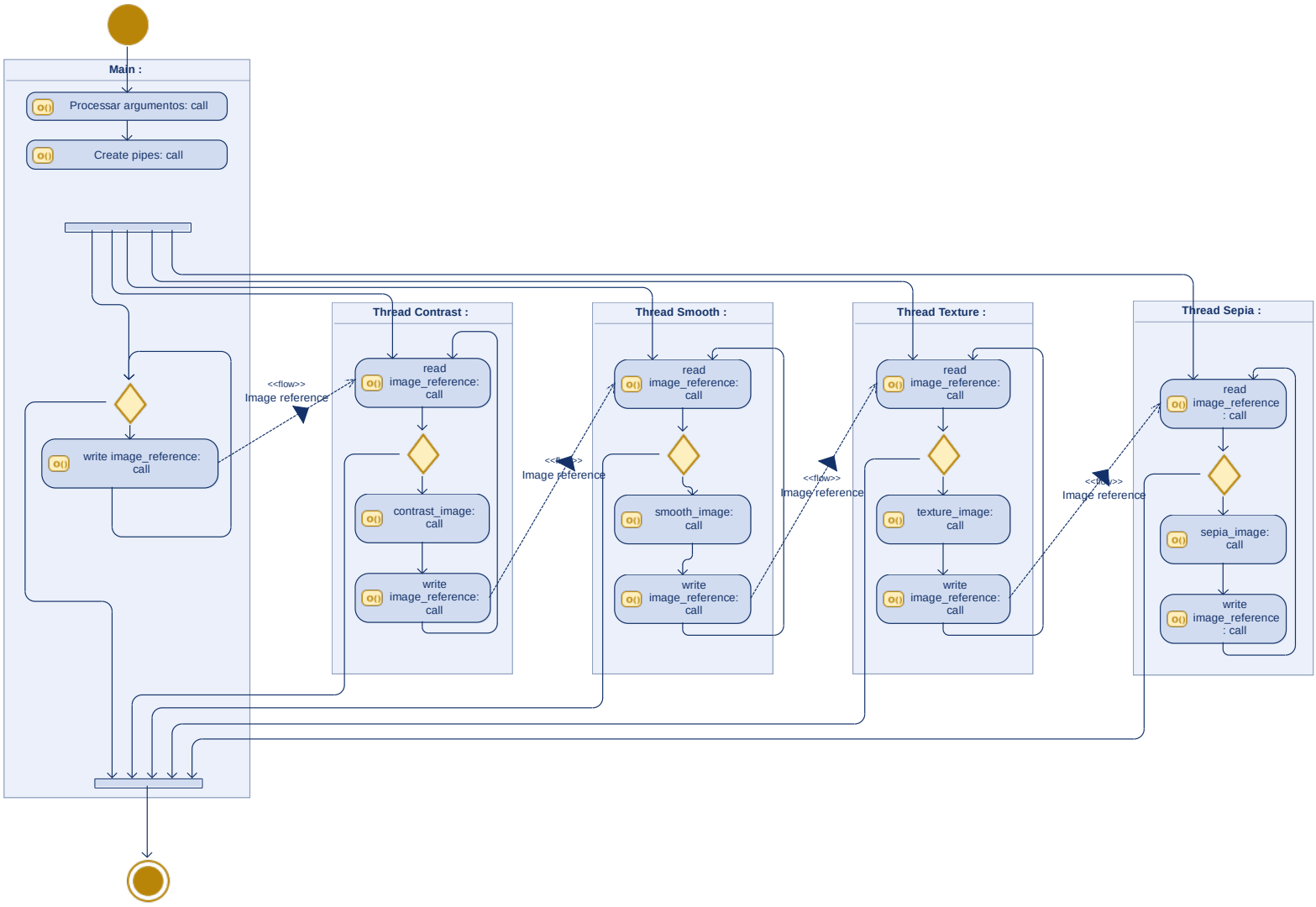


Figura 2: Paralelização do old-photo-paralelo-B

Como ilustra a Figura 2 o **main()** processa os argumentos, crias todos os *pipe* necessários, cria uma *thread* para cada estágio e envia através de um dos *pipes* as referência para as imagens. Cada *thread*/estágio lê do seu *pipe* as referências para as imagem, executa a sua transformação e envia a referência da imagem para o estágio/*thread* seguinte através doutro *pipe*.

O primeiro estágio deverá ler do disco a imagen, e últimos estágio deverá guardar no disco o resultado da aplicação das quarto transformações.

Os alunos poderão decidir criar mais estágios/*threads* (e correspondentes *pipes*) do que aqueles apresentados na figura, se daí obtiverem ganhos de desempenho.

Depois de enviar as referências de todas as imagens através do *pipe*, o **main()** deverá esperar pela terminação da *threads*. Cada uma das *threads*/estágios deverá terminar depois de ter processado todas as imagens.

3 Funcionalidades

As aplicações deverão ser desenvolvidas em **C** e executar em Linux/WSL/MAC OS X.

A lista de imagens a processar será fornecida a cada uma das aplicações paralelas através de um ficheiro de texto que contem em cada linha o nome de uma imagem a ser processada. Depois de lerem os nomes das imagens do ficheiro de configuração, para cada uma das imagens, a aplicação paralela deverá produzir uma imagem com efeito antigo e colocá-la numa pasta específica, tal como a aplicação **old-photo-serial.c**.

As imagens resultantes terão o nome da imagem original, mas serão armazenadas em diretorias específicas tal como descrito na Secção 3.4.

3.1 Dados de entrada da aplicação **old-photo-paralelo-B**

A diretoria onde se encontram as imagens originais e o número de *threads* são definidos pelo utilizador na linha de comando aquando da execução das aplicações. Na diretoria deverá existir o ficheiro **`image-list.txt`** .

Esta aplicação necessita de um ficheiro com a textura a aplica às imagens (**`paper-texture.png`**), esta imagem poderá ser colocada na diretoria da aplicação ou na diretoria onde se encontram as imagens.

Para a execução da aplicação **old-photo-paralelo-B** o utilizador deverá indicar a diretoria onde se encontram as imagens e o número de *threads* a criar como exemplificado de seguida.

```
./old-photo-paralelo-B ./dir-1 4
./old-photo-paralelo-B ./dir-2 8
./old-photo-paralelo-B . 1
```

O primeiro argumento corresponde à diretoria e o segundo ao número de *threads*.

O número de *threads* deve ser um qualquer número inteiro positivo e indica quantas *threads* efetuarão o processamento das imagens. Se, por exemplo, o utilizador indicar **1** como o número de *threads* a criar, o programa utilizará apenas uma *thread* **para além** do ***`main()`*** para efetuar o processamento de todas as imagens.

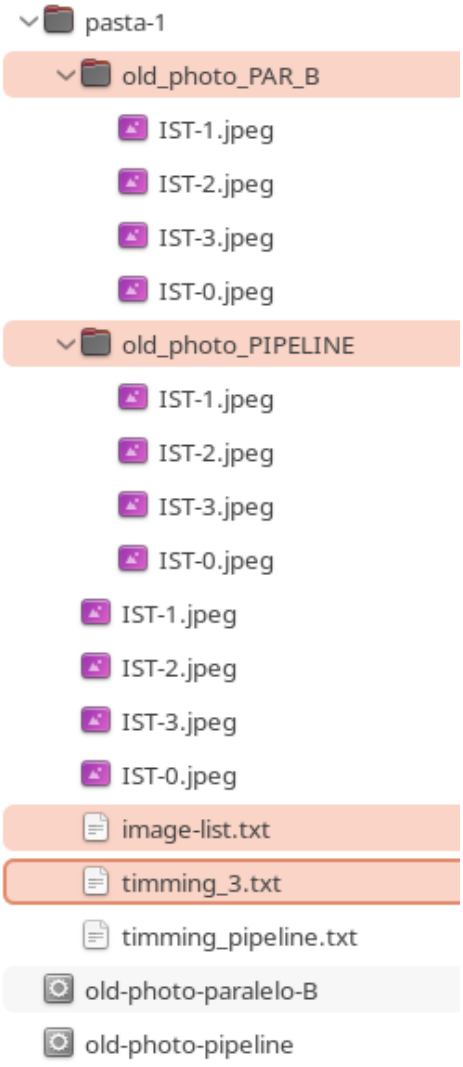
3.2 Dados de entrada da aplicação **old-photo-pipeline**

A diretoria onde se encontram as imagens originais é definida pelo utilizador na linha de comando aquando da execução das aplicações. Na diretoria deverá existir o ficheiro **`image-list.txt`** .

Esta aplicação necessita de um ficheiro com a textura a aplica às imagens (**`paper-texture.png`**), esta imagem poderá ser colocada na diretoria da aplicação ou na diretoria onde se encontram as imagens.

Para a execução da aplicação **old-photo-pipeline** o utilizador deverá indicar a diretoria onde se encontram as imagens como exemplificado de seguida.

```
./old-photo-pipeline ./dir-1
./old-photo-pipeline ./dir-2
./old-photo-pipeline .
```



O primeiro argumento corresponde à diretoria onde se encontram as imagens.

3.3 Imagens a processar

O ficheiro **`image-list.txt`** contém a lista das imagens que se encontram na diretoria e que devem ser processadas. Este ficheiro contêm um nome de ficheiro por linha. As aplicações deverão ler este ficheiro e só as imagens aí listadas deverão ser processadas. Assim, a diretoria indicada pelo utilizador poderá conter mais imagens do que aquelas a serem processadas.

Apenas deverão ser processadas imagens do formato Jpeg.

Serão fornecidos conjuntos de imagens (com ficheiro **`image-list.txt`**) de modo que os alunos tenham diversos conjuntos de dados variáveis e comparáveis. Os alunos podem naturalmente utilizar outras imagens durante o desenvolvimento e teste do projeto.

3.4 Resultados

A execução da aplicação produz um conjunto de novas imagens, correspondentes à transformações de cada uma das imagens iniciais.

O nome das novas imagens será o mesmo da imagem original, mas colocadas numa diretoria específica e relativa à pasta indicada na linha de comando.

Essas diretorias dever-se-ão chamar **old_photo_PAR_B** ou **old_photo_PIPELINE** (para a **old-photo-paralelo-B** ou **old-photo-paralelo-pipeline** respetivamente) e deverão ser criadas pelas aplicações a desenvolver na diretoria indicada na linha de comando pelo utilizador e onde se encontram as imagens originais.

Na imagem anterior, o utilizador executou as aplicações assim:

```
./old-photo-paralelo-B ./pasta-1 3
./old-photo-paralelo-pipeline ./pasta-1
```

3.5 Interrupção de execução

Se a aplicação for interrompida a meio do processamento das imagens, apenas parte dos resultados terão sido produzidos e guardados no disco.

Se o utilizador voltar a executar a aplicação, não deverá ser necessário voltar a produzir os ficheiros resultado já existentes. A aplicação só deverá processar e gastar tempo na criação dos ficheiros em falta.

Para verificar se um ficheiro existe os alunos poderão usar as funções *`access()`* como no seguinte exemplo:

```
if( access( nome_fich, F_OK ) != -1){
   printf("%s encontrado\n", nome_fich);
}else{
   printf("%s nao encontrado\n", nome_fich);
}
```

4 Avaliação de desempenho.

O relatório a produzir pelos alunos deverá conter os resultados da avaliação de desempenho (tempos de execução e speedups) do programa e de cada uma das *threads*, para tal será necessário instrumentar o código para recolher os tempos de processamento *do* ***old-photo-paralelo-B*** *e* ***old-photo-paralelo-pipeline***.

Com base nos tempos de execução medidos, os alunos deverão completar o ficheiro excel produzido na parte A do projeto com os resultados obtidos com o **old-photo-paralelo-B** e **old-photo-pipeline**.

Os alunos deverão executar estas duas aplicações e recolher os tempos de execução para os *datasets* usados na primeira parte.

4.1 old-photo-paralelo-B

Os alunos deverão incluir no código do **old-photo-paralelo-B.c** as funções de leitura de tempos (como no laboratório 3) e em cada execução produzir um ficheiro com esses resultados. Este ficheiro dever-se-á chamar **timing_<n>.txt** (em que **n** deverá ser substituído pelo número de *threads*) e deverá ser criado na pasta onde se encontram as imagens.

O ficheiro contém para o programa total e cada uma das *threads* a seguinte informação: numero de imagens processadas e tempo de execução. Este ficheiro também deverá conter o tempo de processamento de uma das imagens.

O seguinte exemplo exemplificar o ficheiro **timing_3.txt** Para um processamento de 4 imagens e três *threads*:

```
IST-2.jpg   1.10
IST-3.jpg   1.16
IST-1.jpg   2.57
IST-4.jpg   2.95
thread_0  2 2.57
thread_1  1 4.05
thread_2  1 1.16
total     4 8.99
```

A ordem pela qual aparecem estes valores podem ser diferentes, mas os tempos das imagens deverão aparecer todos juntos, assim como os tempos da threads, que deverão também aparecer todos juntos

4.2 old-photo-pipeline

Os alunos deverão incluir no código do **old-photo-pipeline.c** as funções de leitura de tempos (como no laboratório 3) e em cada execução produzir um ficheiro com esses resultados. Este ficheiro dever-se-á chamar **timing_pipeline.txt** e deverá ser criado na pasta onde se encontram as imagens.

O ficheiro deverá conter o tempo total de execução do programa, assim como o *timestamp* (tempo desde o início do programa) do começo e fim de processamento de cada imagem, como ilustra o seguinte exemplo (**timing_pipeline.txt**) para um processamento de 3 imagens:

```
IST_1.jpg start 0.001
IST_2.jpg start 2.211
IST_1.jpg end   3.012
IST_3.jpg start 3.735
IST_2.jpg end   4.097
IST_3.jpg end   5.735
total           5.741
```

5 Submissão do projeto

5.1 Prazo de submissão

O prazo para submissão da resolução da Parte B do projeto é dia **7 de Janeiro às 19h00** no FENIX.

Antes da submissão, os alunos devem criar grupos de dois alunos e registá-los no FENIX.

5.2 Ficheiros a submeter

Os alunos deverão submeter um ficheiro **.zip** contendo:

- todo o código da aplicação **old-photo-paralelo-B** numa diretoria específica

- todo o código da aplicação **old-photo-pipeline** numa diretoria específica
- a(s) `Makefile` para a compilação das aplicações
- todos os ficheiros de tempos (**timing_<n>.txt** e **timing_pipeline.txt**, descritos na Secção 4) organizados em diversas pastas
- ficheiro excel contendo os resultados de desempenho para os diversos *datasets* (como descrito na Secção 4).

Não incluir no **.zip** as imagens dos *datasets* utilizadas.

5.3 Relatório

Os alunos deverão submeter um pequeno documento/relatório (chamado **pconc-relatorio.pdf**). O modelo do relatório será fornecido posteriormente, mas deverá listar as funcionalidades implementadas e apresentar os seguintes resultados para várias execuções:

- Numero de *threads*
- Tempo total de execução da aplicação
- Throughput
- Características dos Computadores

6 Avaliação do projeto

Do projeto será dada tendo em consideração o seguinte:

- Número de aplicações e funcionalidades implementadas
- Modo de gestão das *threads* e recursos
- Estrutura e organização do código
- Tratamento de erros
- Comentários
- Relatório