# Projeto de Sistemas Operativos 2017-18
# `heatSim`
# Exercício 4

LEIC-A / LEIC-T / LETI
IST

**Resumo**

Este documento constitui um guia para a realização do quarto exercício do projeto da disciplina de Sistemas Operativos.

Este exercício consiste em estender o simulador `heatSim` por forma a permitir que a simulação possa ser terminada a meio e continuar posteriormente a partir do estado intermédio guardado em memória secundária.

## 1 Introdução

Os alunos devem ler primeiro todos os guias anteriores. O ponto de partida deste último exercício é a **versão de memória partilhada** do `heatSim` (desenvolvida no exercício 3).

Tendo este exercício como base, pretende-se eliminar uma limitação do projeto desenvolvido até ao momento: para computações mais demoradas, caso a execução seja terminada abruptamente (por exemplo, devido a uma falha do sistema operativo ou da máquina, ou a uma terminação do processo por Ctrl+C), toda a computação efetuada até ao momento é perdida e é necessário recomeçar completamente após a recuperação do sistema. A extensão consistirá em salvaguardar estados intermédios da matriz durante a simulação longa. Caso a simulação seja terminada a meio, o `heatSim` poderá ser lançado posteriormente e continuar a computação a partir do último estado salvaguardado.

## 2 Salvaguarda e Recuperação

Pretende-se então suportar a salvaguarda durável (em memória secundária) de estados intermédios da matriz atual. Esta extensão permitirá que o `heatSim` possa ser terminado (de forma inesperada ou ordeira) e, posteriormente, continuar a sua execução a partir do último estado salvaguardado.

### 2.1 Formato do ficheiro

A matriz guardada no ficheiro de salvaguarda deverá seguir o formato usado pela função *dm2dPrint*, usada nos exercícios anteriores.

Por exemplo, esta cadeia de caracteres, representando uma matriz $2 \times 2$, respeita esse formato:

```
10.0000 10.0000
10.0000 10.0000
```

## 2.2 Requisitos base

Aqui especificamos os requisitos base do projeto. É recomendado que os alunos comecem por construir uma solução que cumpra estes requisitos antes de passarem à próxima secção.

O `heatSim` passa a receber dois argumentos adicionais de linha de comandos: `fichS periodoS`. Estes influenciam o programa da seguinte forma:

- O argumento `fichS` indica o nome do ficheiro que contém o estado intermédio da matriz atual.

  Caso o ficheiro já exista, o `heatSim` deverá assumir que este ficheiro tem uma matriz salvaguardada de uma execução anterior incompleta. Nesse caso, o `heatSim` deve carregar a matriz e iniciar a computação paralela a partir desse estado.

  A contagem do número de iterações (para o limitar ao valor máximo passado por argumento) deve ignorar as iterações já realizadas em execuções anteriores (que produziram a matriz no ficheiro); ou seja, o contador de iterações deve começar do zero.

  Caso o ficheiro especificado não exista ou o seu conteúdo esteja corrompido, a simulação deve iniciar a partir do estado inicial da matriz.

- O argumento `periodoS` é um inteiro igual ou superior a zero que define a periodicidade das salvaguardas. Nesta fase do exercício, `periodoS` deve ser interpretado como o número de iterações que devem ser completadas até que a matriz seja salvaguardada. Caso este argumento seja zero, a salvaguarda periódica é desativada.

  Obviamente, deve-se ter o cuidado de assegurar que todas as fatias da matriz salvaguardadas em ficheiro correspondem à mesma iteração do algoritmo paralelo.

- Uma vez iniciada a salvaguarda da matriz no ficheiro, a computação paralela por parte das tarefas trabalhadoras deve prosseguir em paralelo; ou seja, enquanto o estado da matriz após a iteração $i$ é escrito para o ficheiro, as tarefas trabalhadoras devem poder continuar a simulação nas iterações seguintes. Para assegurar este requisito, a salvaguarda deve ser feita por um processo filho que corre em paralelo com o processo principal e utiliza uma cópia da matriz mais recente do pai (do momento em que o filho foi criado).

- Sempre que a execução do `heatSim` termina com sucesso, o ficheiro de salvaguarda deve ser eliminado depois do resultado ser impresso no *stdout*. Isto deve acontecer mesmo que o ficheiro não tenha sido alterado pela execução atual.

## 2.3 Requisitos avançados

Adicionalmente, devem ser assegurados os seguintes requisitos avançados. Recomenda-se que o grupo aborde os requisitos abaixo apenas após ter resolvido a funcionalidade básica descrita acima. Não sendo obrigatório, é recomendado seguir a ordem proposta abaixo.

1. A rescrita do ficheiro de salvaguarda com novo conteúdo da matriz deve ter em conta que o sistema ou o processo podem terminar abruptamente enquanto o conteúdo anterior do ficheiro está a ser substituído pelo novo conteúdo. A solução deve assegurar que, mesmo na presença desse tipo de falhas, o ficheiro fica num estado consistente. Mais concretamente: caso a substituição do conteúdo do ficheiro seja terminada abruptamente, o ficheiro deverá manter o conteúdo anterior (consistente).

   Para cumprir este requisito a escrita do novo conteúdo deve ser feita para um ficheiro auxiliar cujo nome é dado pelo nome do ficheiro de salvaguarda (original) sufixado pelo caracter '~'. Só após a escrita no ficheiro auxiliar estar completa e persistida é que o ficheiro auxiliar deve substituir o ficheiro original.

2. No caso de matrizes grandes, a escrita em ficheiro pode ser mais demorada que a periodicidade definida para a salvaguarda da matriz. Numa situação em que um estado anterior da matriz ainda está a ser escrito em ficheiro, lançar um novo processo para salvaguardar um estado mais recente da matriz pode ser contraproducente pois ambos competirão pelo mesmo recurso (o ficheiro) e atrasar-se-ão. Para evitar estas situações, deverá ser implementada a seguinte política: quando é alcançado o período para iniciar nova salvaguarda, o processo pai só lança o novo processo filho caso o filho anterior já tenha terminado.
3. O argumento `periodoS` deve passar a ser interpretado como o número de segundos entre salvaguardas (ou seja, deixa de ser lido como número de iterações).
4. Além das salvaguardas periódicas descritas no ponto anterior, a matriz atual também deve ser salvaguardada quando o processo recebe um Ctrl+C (*signal SIGINT*). Neste caso, o `heatSim` não deve avançar para iterações posteriores e deve terminar mal o processo filho termine.

### 2.4 Notas de implementação

- Para ler uma matriz guardada em ficheiro de salvaguarda no formato especificado, o projeto deverá usar a função `readMatrix2dFromFile` que existe na nova versão da biblioteca `matrix2d` disponibilizada na página dos laboratórios.
- Caso achem conveniente, os alunos poderão alterar/estender o código e interface da biblioteca `matrix2d`.
- Para o acesso a ficheiros, deverão ser usadas as funções `fopen`, `fwrite`, `fread`, `fflush`, `fclose`, etc. da biblioteca `stdio`.
- A manipulação de ficheiros deverá recorrer às funções `rename` e `unlink`.
- Para testar se um processo filho já terminou, deverá usar a função `waitpid` (variante do `wait`) com a opção não bloqueante.
- Tenha atenção que, quando carrega Ctrl+C na consola, o *signal* respetivo é enviado tanto ao processo pai como ao processo filho (caso exista). Deverá pois ter o cuidado de redefinir o tratamento do Ctrl+C por parte do filho.

## 3 Experimente

Para testar os requisitos deste exercício, é conveniente inserir atrasos artificiais (por exemplo, com `sleep`) na função que imprime o conteúdo da matriz em ficheiro e/ou nas iterações da simulação.

Corra o `heatSim` com `periodoS` nulo (ou seja, com salvaguarda periódica desativada) com uma simulação suficientemente demorada (para demorar cerca de 1 minuto); guarde o resultado num ficheiro e anote o tempo que demorou (pode usar o comando `time`).

De seguida, corra a mesma simulação mas com `periodoS` colocado a 10 (segundos). Consulte o conteúdo do ficheiro de salvaguarda (por exemplo, com o comando `cat`) e confirme que, a cada 10 segundos, o conteúdo é diferente. A meio da simulação (por exemplo, ao fim de 30 segundos), termine o `heatSim` emitindo um `SIGKILL` com o comando `kill` na consola. De seguida, lance de novo o `heatSim` e deixe a simulação terminar. No final, confirme que: i) a execução de recuperação (a partir do ficheiro guardado pela execução terminada a meio) demora menos tempo que a simulação completa; ii) os resultados obtidos por ambas são iguais. Nota importante: como a escrita da matriz em ficheiro pode perder alguma precisão dos valores *double*, o carregamento posterior da matriz poderá introduzir pequenas diferenças face ao valor da matriz da execução sem falhas; isto poderá levar a pequenos desvios nos resultados finais obtidos por cada execução (sem falhas vs. interrompida e recuperada).

# 4 Entrega e avaliação

Os alunos devem submeter um ficheiro no formato zip com o código fonte e o ficheiro *Makefile*. O exercício deve obrigatoriamente compilar e executar nos computadores dos laboratórios.

A data limite para a entrega do primeiro exercício é 2/12/2017 às 23h59m. A submissão é feita através do Fénix.

Após a entrega, o corpo docente disponibilizará a codificação da respetiva solução.