

## Sistemas Operativos

Adaptação de guião desenvolvido por Artur Carneiro Pereira

Ano lectivo 2025/2026

# Aula Prática N° 10

## IPC com memória partilhada e semáforos

### Objectivo

Estudo da comunicação entre processos (IPC) usando memória partilhada e semáforos em Unix.

### Guião

#### 1. *Introdução ao uso de memória partilhada*

- a) Na directória base, analise o módulo descrito nos ficheiros `sharedMemory.h` e `sharedMemory.c`, que define um conjunto de operações efectuadas sobre um bloco de memória partilhada e sua implementação em Unix.
- b) O programa `reg-cri.c` ilustra os passos de criação/ligação, destruição, anexação/desanexação e acesso a uma região de memória partilhada. Tratando-se de uma aplicação concorrente, evidencia também a necessidade de acesso com exclusão mútua a uma região crítica. Procure identificar o código que constitui a região crítica.
- c) Crie o ficheiro executável (`make reg-cri`) e execute-o diversas vezes, usando as opções disponíveis, procurando determinar a sequência pela qual elas devem ser utilizadas. Em particular, faça-o incrementar 1000 vezes a variável, e procure interpretar o resultado obtido em sucessivas execuções. Compare os resultados com aqueles obtidos com o programa `incrementer` do guião sobre *threads*.
- d) Altere o código de modo que as regiões críticas sejam executadas em exclusão mútua. Dica: analise o ficheiro `semaphore.h` e use um semáforo para obter a exclusão mútua.

## 2. Introdução ao uso de semáforos

- a) Na directória `servcli`, analise o módulo descrito nos ficheiros `semaphore.h` e `semaphore.c`, já usado na última alínea do exercício anterior, que define um conjunto de operações efectuadas sobre agregados de semáforos e respectiva implementação em Unix.
- b) Analise o programa descrito no ficheiro `server.c`. Este programa representa aquilo que se costuma designar de *servidor*, isto é, um programa que presta serviços a outros que lhos solicitam, os chamados *clientes*. Neste caso, recebe *strings* e converte os seus caracteres alfabéticos minúsculos em caracteres alfabéticos maiúsculos, antes de os devolver ao programa que solicitou a conversão. A comunicação é implementada através de quatro funções cujos protótipos estão declarados no ficheiro de interface `comm.h`.
- c) Analise o código do programa descrito no ficheiro `client.c`, que constitui o *cliente* que interaculta com o *servidor* referido acima. A comunicação é implementada através de quatro funções cujos protótipos estão declarados no ficheiro de interface `comm.h`.
- d) O módulo `comm-shm.c` representa a implementação das funções de comunicação, quer do lado do *servidor*, quer do lado do *cliente*, recorrendo a memória partilhada e a semáforos. Procure entender como se processa a interacção e qual é o papel desempenhado por cada um dos semáforos presentes: R, S e A.
- e) Regiões de memória partilhada e semáforos são recursos do sistema operativo. Em Unix, eles designam-se recursos IPC. O comando `ipcs` lista os recursos IPC correntemente atribuídos. Consulte no manual *on-line* a descrição do comando `ipcs` (`man ipcs`). Execute-o e interprete a listagem apresentada.
- f) Crie o executável *servidor* (`make server`) e execute-o numa janela terminal.
- g) Execute de novo o comando `ipcs` (numa outra janela terminal) e constate as alterações entretanto ocorridas.
- h) Crie o executável *cliente* (`make client`) e execute-o numa outra janela terminal. Lance pelo menos mais um processo cliente numa nova janela terminal. Constate como decorre a interacção comutando entre as diferentes janelas.
- i) Os programas *servidor* e *clientes* não contemplam mecanismos de terminação. Termine-os, usando a combinação de teclas CRTL-C. Considere as várias situações.
- j) Volte a lançar o *servidor* e procure entender o que ocorre.
- k) Consulte no manual *on-line* a descrição do comando `ipcrm` (`man ipcrm`) que possibilita a remoção de recursos IPC atribuídos pelo sistema de operação e remova os recursos que tinham sido reservados pelo *servidor*.
- l) Como poderia tornar a interacção independente da velocidade relativa de execução dos *clientes*? Note que, na versão actual, enquanto um *cliente* não ler o resultado do processamento da sua mensagem pelo *servidor*, mais nenhum *cliente* consegue processar o que quer que seja.