# INE 5410 - Laboratório

# AULA 02 – SISTEMAS CONCORRENTES

---

## 1. INTRODUÇÃO

Conforme foi visto na aula passada, utilizando a linguagem Pascal-FC tem-se a possibilidade de executar diversos processos num ambiente de concorrência, permitindo-se experimentar as diversas características de sistemas concorrência, entre elas a competição por recursos e o não determinismo.

Sistemas concorrentes são aqueles em que diversas threads ou processos podem ser executados de modo a explorar o paralelismo disponível nas plataformas em que são executados.

## 2. OBJETIVO DA AULA

O objetivo desta aula prática é utilizar a linguagem Pascal-FC para analisar alguns exemplos de aplicações envolvendo concorrência de modo a verificar a necessidade de se exercer controle sobre a concorrência estabelecida entre os vários processos que os compõem.

## 3. EXEMPLOS DE APLICAÇÕES ENVOLVENDO CONCORRÊNCIA

Iremos definir aqui algumas aplicações envolvendo o uso da concorrência entre processos na linguagem Pascal-FC.

### 3.1. Atualização distribuída de uyma variável

- Execute o código a seguir no ambiente Pascal-FC ... observe o resultado e tente explicar os resultados obtidos.

```
PROGRAM atualiza;
VAR
   conta : integer;
PROCESS P1;
VAR
  I : integer;
BEGIN
  FOR I := 1 TO 20 DO
    conta := conta + 1
END; (* P1 *)

PROCESS P2;
```

```
VAR
  I : integer;
BEGIN
  FOR I := 1 TO 20 DO
    conta := conta + 1
END; (* P2 *)

BEGIN
   conta := 0;
COBEGIN
  P1;
  P2
COEND;
WRITELN(Contagem total: ’,conta)
END.
```

- Insira dois novos processos no programa e verifique os resultados.

## 3.2. Ordenando Valores

- O código a seguir refere-se a um programa de ordenação de valores de um vetor de 50 posições (predefinido no programa, criado com o auxílio de uma função randômica. Execute o programa e verifique o resultado, assim como o tempo utilizado para finalizar a ordenação.

```
PROGRAM ordena;
VAR
vetor : ARRAY[1..10] of integer;

PROCESS Type OrdProc(pid : INTEGER);
VAR
  I, BUFFER   : integer;
  ORD : BOOLEAN;

BEGIN
  REPEAT
     SLEEP(5);
     ORD := FALSE;
     FOR I := 1 TO 9 DO
        IF vetor[i] > vetor[i+1]
           THEN
             BEGIN
               ORD := TRUE;
               BUFFER := vetor[I];
               vetor[I] := vetor[I+1];
               vetor[I+1] := buffer
             END;
  UNTIL NOT ORD;
END;
VAR
J : INTEGER;
P : ARRAY[1..15] OF OrdProc;
```

```
BEGIN
   FOR J:= 1 TO 10
    DO
      BEGIN
        vetor[J] := random(15);
        write(vetor[J], ' ');

      END;

COBEGIN
  P[1](1);
COEND;
   writeln;
   FOR J:= 1 TO 10
    DO write(vetor[J], ' ')
END.
```

- Duplique o número de processos e verifique o tempo para a ordenação;
- Repita a execução do programa com 4, 8 e 16 processos, sempre verificando o tempo de ordenação.
- Tente resolver o problema do programa de ordenação com diversos processos, utilizando variáveis de proteção. Execute o programa e verifique o resultado.