# Capítulo II – Modelos de Programação Distribuída

# Exemplos de linguagens com comunicação por mensagens,

# Ada
# OCCAM
# Linda

**From: M. Ben-Ari**
**Principles of Concurrent and Distributed Programming**
**Prentice Hall 1990**

Paula Prata,
Departamento de Informática da UBI
http://www.di.ubi.pt/~pprata

## Exemplo 1 - Linguagem Ada

- Desenvolvida pelo Dep. De Defesa dos EUA

- A comunicação baseia-se no facto de um processo poder invocar operações num outro processo

- O processo **Receptor** declara os procedimentos que podem ser invocados por processos externos:

**entry** proc1 (valor: **in** <*tipo*> ; resultado: **out** <*tipo*> )

nome do procedimento — informação trocada entre os processos

- O processo **Emissor** declara-se pronto para comunicar (encontro ou rendez-vous), indicando o nome do procedimento:

<nome do receptor>**.proc1 (...)**

…

# Exemplo 1 - Linguagem Ada

- O processo **Receptor** recebe a mensagem através da palavra chave **accept** a que se segue o código do procedimento externo:

**accept** proc1 (valor:in ; resultado:out )
**do**
<instruções>
**end proc1;**

**-** O **receptor** ao executar um **accept**
**bloqueia** até que exista uma chamada à respectiva **entry**

**-** Reciprocamente, **uma chamada a uma entry bloqueia o emissor** até que o receptor atinja o accept correspondente.

...

# Exemplo 1 - Linguagem Ada

**Quando se dá o encontro,**

. os dados de entrada são copiados do emissor para o receptor

. o código do procedimento é executado no receptor

. os parâmetros de retorno são copiados para o emissor

. ambos os processos prosseguem a execução em paralelo.

. Se vários processos chamam uma entry, **apenas um** é desbloqueado quando o receptor executa o accept correspondente.

. Para cada entry existe uma fila de espera, sendo o código do procedimento executado em exclusão mútua.

…

# Exemplo 1 - Linguagem Ada

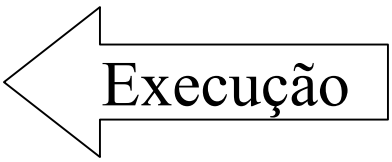
Para uma recepção selectiva de mensagens, existe a instrução:

SELECT

```
select
        when <condição 1>   =>  accept entry1 (…)
                                        do
                                                <instruções>
                                        end entry1
                                        <outras instruções>
        or
        when <condição 2>   =>  accept entry2 (…)
                                        do
                                                <instruções>
                                        end entry2
                                        <outras instruções>
        or
                        …
        else
                        <instruções>
end select
```

...

## Exemplo 1 - Linguagem Ada




1 - Todas as guardas são avaliadas
(uma alternativa cuja guarda é verdadeira diz-se um alternativa aberta)

2 – Se existem alternativas abertas, determina-se quais os accepts que têm um rendez-vous à espera

3 - Se existe algum, é executado
se existem vários, é escolhido um "ao acaso"

4 – Se nenhuma das guardas é verdadeira, ou nas verdadeiras não há processos suspensos, é executada a cláusula else caso exista

5 – Se não existe a cláusula else e nenhum processo pretende comunicar, o processo que executa o select é suspenso até que um outro se apresente numa alternativa aberta

6 – Se não há alternativas abertas, nem cláusula else, ocorre um erro

**Exemplos de linguagens com comunicação por mensagens**

**…**

**Exemplo 1 - Produtor/Consumidor em Ada**

Em Ada um processo é uma "task" =
especificação + corpo

Especificação do processo Buffer:

```
task Buffer is
        entry Append (I:in Integer);
        entry Take (I:out Integer);
endBuffer;
```

O processo Produtor executa o Append:
**Buffer.Append(I);**

O processo Consumidor executa o Take:
**Buffer.Take(I);**

…

# Exemplo 1 - Produtor/Consumidor em Ada

```
 task Buffer is
          B: array (0..N-1) of Integer;
          In_Ptr, Out_Ptr : Integer := 0;
          count : Integer := 0 ;
begin
  loop
    select
      when count < N   =>  accept Append (I:in Integer)
                                  do
                                        B(In_ptr) := I;
                                  end Append;
                                  count:= count + 1;
                                  In_ptr := (In_ptr + 1) mod N;
          or
      when count >0   =>   accept Take (I: out Integer)
                                  do
                                        I := B(Out_ptr) ;
                                  end take;
                              count:= count - 1;
                              Out_ptr := (Out_ptr + 1) mod N;
     end select;
  end loop;
end Buffer;
```

## Exemplos de linguagens com comunicação por mensagens

## …

## Linguagem Ada - Resumo

- A comunicação é síncrona

- Há identificação do processo Receptor mas não do processo emissor

- Comunicação é bidireccional (num único “rendez-vous”)

- Ada permite a criação dinâmica de processos (“tasks”)

# Exemplos de linguagens com comunicação por mensagens

…

# Exemplo 2 - Linguagem OCCAM

-Linguagem baseada no modelo de programação concorrente, **CSP** (**C**ommunicating **S**equential **P**rocesses" - [Hoare 78])

- O facto de o modelo da linguagem ser concorrente significa que as primitivas que exprimem

**processos concorrentes**
**comunicação**
**sincronização de processos**

estão ao mesmo nível de abstracção que as restantes estruturas da linguagem.

- Uma aplicação em OCCAM é um conjunto de processos concorrentes que comunicam através de canais.

- Um canal é usado para comunicar dados entre dois (**e apenas dois)** processos.

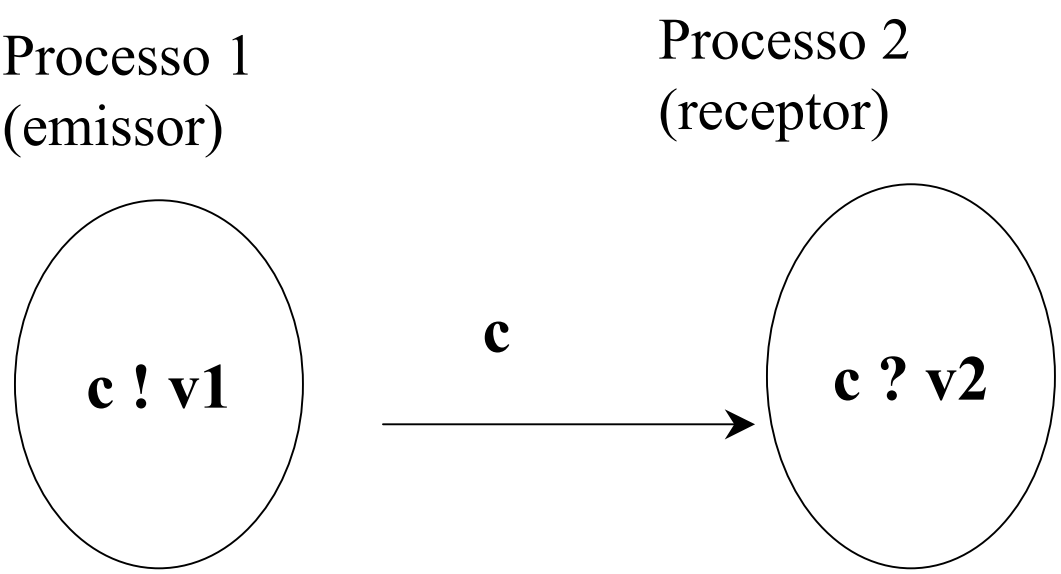
...

# Exemplo 2 - Linguagem OCCAM

- Comunicação unidireccional:
  Um processo usa o canal para input;
  O outro processo usa o canal para output.

- Não há comunicação através de memória partilhada.

Comunicação síncrona




- **escreve** o conteúdo da variável **v1** no canal **c**

- **lê** para a variável **v2** o conteúdo do canal **c**

...

## Exemplo 2 - Linguagem OCCAM

**Notas:**

=> O primeiro processo a executar um dos comandos acima é suspenso até que o outro atinja a instrução de comunicação correspondente

=> Quando os dois processos estão prontos a comunicar a mensagem é transferida de **v1** para **v2**.

=> Um canal é uma variável do tipo CHAN à qual está associado um protocolo que define a estrutura da mensagem.

...

## Exemplo 2 - Linguagem OCCAM

- Um programa em OCCAM tem 3 tipos de processos primitivos:

input **?**
output **!**
atribuição **:=**

estes podem ser combinados em processos maiores através dos construtores:

IF
WHILE,

SEQ - força a execução sequencial

PAR – força a execução concorrente

ALT - permite a um processo seleccionar uma de um conjunto de possíveis fontes de mensagens.

...

# Exemplo 2 - Linguagem OCCAM

Execução sequencial das instruções delimitadas por endentação

Exemplo i)

```
SEQ
    canal1 ? outro_valor
    valor  := outro_valor  + 1
    canal2 ! Valor
```

Exemplo ii)

**P1**

```
SEQ
    canal1 ! A
    canal2 ? B
```

**P2**

```
SEQ
    canal1 ? X
    canal2 ! Y
```

...

# Exemplo 2 - Linguagem OCCAM

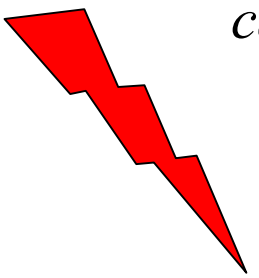
Exemplo iii)

**P1**

```
SEQ
    canal1 ! A
    canal2 ? B
```

**P2**

```
SEQ
    canal2 ! X
    canal1 ? Y
```



**Os dois processos ficariam suspensos indefinidamente**

Exemplo iv)

```
PAR
        SEQ
                canal3 ? valor1
                valor := valor1 +1
        SEQ
                canal4 ! valor2
                valor2 := valor2 +1
```

**O processo "pai" (principal) activa 2 processos que poderão ser executados em paralelo**

...

# Exemplo 2 - Linguagem OCCAM

Apenas será executada a acção associada com o canal que primeiro estiver pronto a comunicar.

Exemplo v)

```
ALT
        canal1 ? Valor1
                <instruções>
        canal2 ? Valor1
                <instruções>
        canal3 ?Valor1
                <instruções>
```

Se mais do que um processo “puder” comunicar ao mesmo tempo, é feita uma escolha “arbitrária”.

...

# Exemplo 2 - Linguagem OCCAM

**As guardas de uma alternativa podem ter associadas uma condição:**

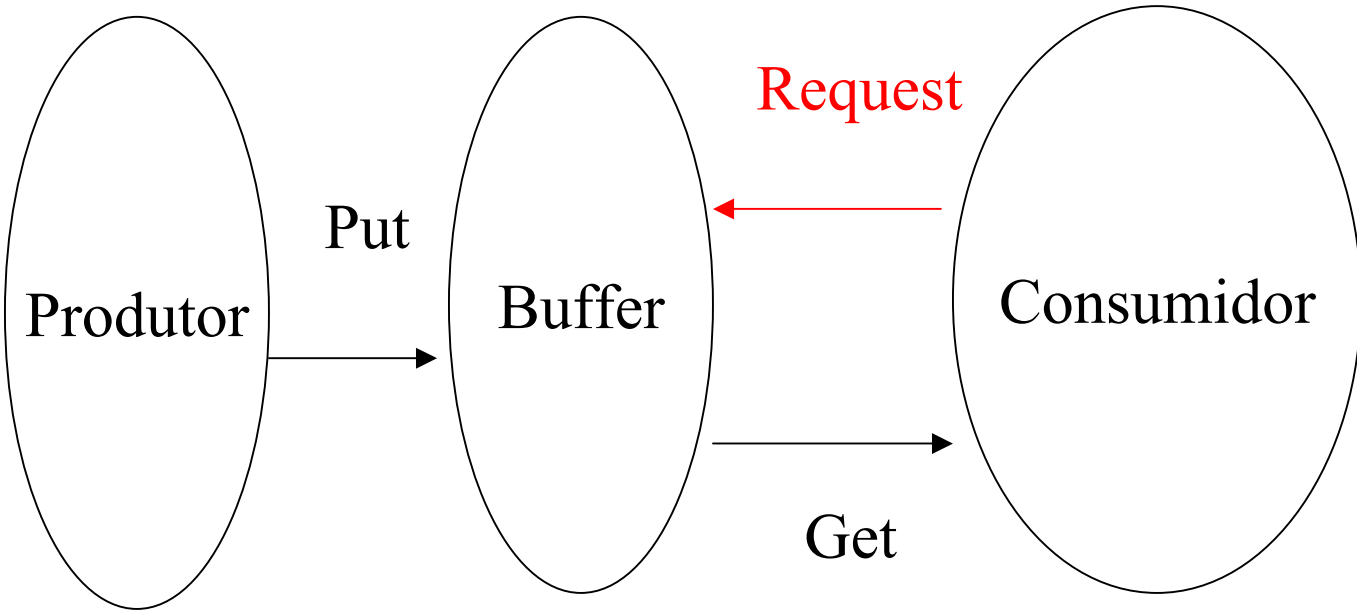
Exemplo vi)

Executa o bloco cuja condição for *true* *e cujo* canal estiver pronto a comunicar.

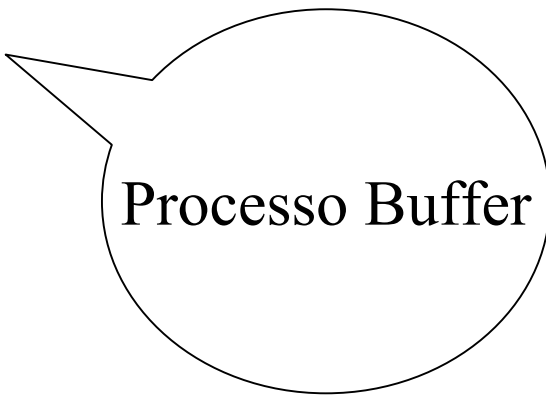
```
ALT
        (valor < 0)  &  canal1 ? Valor1
                <instruções>

        (valor  = 0)  &  canal2 ? Valor1
                <instruções>

        (valor > 0 )  &  canal3 ?Valor1
                <instruções>
```

...

## Exemplo 2 - Produtor/Consumidor em OCCAM




```
CHAN of INT Put, Get, Request:
PAR
        VAL INT BufSize IS 32:
        INT Top, Base, Contents:
        [BufSize] INT Buffer:
        SEQ
                Contents := 0
                Top := 0
                Base := 0
                INT Any:                                    ...
```



...

## Exemplo 2 - Produtor/Consumidor em OCCAM

...

```
WHILE TRUE

  ALT
    Contents < BufSize  & Put ? Buffer[Top]

                              SEQ
                                Contents := Contents + 1
                                Top := (Top + 1 ) REM BufSize

    Contents > 0  & Request ? Any

                              SEQ
                                Get ! Buffer[Base]
                                Contents := Contents - 1
                                Base := (Base + 1 ) REM BufSize

        ...
```

## Exemplos de linguagens com comunicação por mensagens

## …

## Linguagem OCCAM - Resumo

- A comunicação é síncrona e unidireccional

- Comunicação entre dois processos é feita através de um canal partilhado pelos dois processos

- Criação de  processos é estática

## Exemplos de linguagens com comunicação por mensagens

…

### Exemplo 3 - Linguagem Linda [1]

Um modelo de programação pode ser visto como:

*Modelo de Computação* <= permite construir uma actividade computacional isolada (ex.lo , C, Pascal, …)

+

*Modelo de Coordenação* <= permite a ligação de várias actividades num conjunto, fornecendo os mecanismos para a sua criação e intercomunicação

---

[1] Linguagem criada em 1988 por um grupo de investigação de Yale coordenado por David Herlenter

…

## Exemplo 3 - Linguagem Linda

Linda é uma linguagem de **coordenação:**

- não fornece quaisquer mecanismos para a computação mas apenas para a coordenação das suas actividades

- como linguagem de coordenação pode ser usada em conjunto com outras linguagens: C, C++, Pascal, Scheme, Prolog, Modula2, Java [2])

---

[2] http://java.sun.com/products/javaspaces

- http://www.cs.yale.edu/HTML/YALE/CS/Linda/linda.html

...

## Exemplo 3 - Linguagem Linda

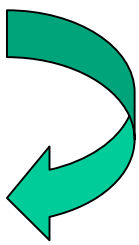
Linda é baseada num modelo de comunicação **generativa**:

**os processos**

- não partilham explicitamente quaisquer variáveis

- não trocam mensagens directamente

- acedem a um espaço de endereçamento único e geram estruturas de dados para dentro desse **espaço**



**espaço de tuplos - TS (Tuple Space):**

"saco" não ordenado de tuplos que são endereçados pelo seu **conteúdo**

...

## Exemplo 3 - Linguagem Linda

Um **tuplo** é uma sequência ordenada de campos de um determinado tipo, onde cada **campo**

- pode conter um valor de facto (valor actual)

ou

- pode ser um espaço onde um valor pode ser instanciado (valor formal)

**Exemplos:**

("Joana", 27, 1.84, 'A')
(45, "Guida", false)
("Joana", idade:integer, altura:double, nota:char)
(numero:integer, "Guida", sócia:boolean)

(Obs. os tuplos com campo formais são também denominados "anti-tuplos")

…

## Exemplo 3 - Linguagem Linda

O **espaço de tuplos - TS:**
- é partilhado por todos os processos
- é um saco, não um conjunto ordenado
(pode haver tuplos idênticos)
- é anónimo (não se sabe que processo criou ou possui a mensagem)

**Operações em Linda:**

**Output (T)** - adiciona o tuplo T ao espaço de tuplos

**Input (T)** - retira do TS um tuplo correspondente a T. Se não existir nenhum, o processo é suspenso até que um tuplo com as características desejadas seja depositado.

**Read (T)** - como input, mas não retira o tuplo do TS

**Try_Input (T) -** Versão não bloqueante de Input
**Try_Read (T)** - Versão não bloqueante de Read

…

**Exemplo 3 - Multiplicação de Matrizes em Linda** C = A*B)

```
Processo principal
… // declaração de variáveis
Output ( ‘A’, 1, (1, 2, 3));
Output ( ‘A’, 2, (4, 5, 6));      } linhas da primeira matriz
Output ( ‘A’, 3, (7, 8, 9));

Output ( ‘B’, 1, (1, 0, 2));
Output ( ‘B’, 2, (1, 2, 1));      } colunas da segunda matriz
Output ( ‘B’, 3, (1, 1, 1));

Output (“next”, 1);        <= contador de tarefas
for I in 1..3 loop
        for J in 1..3 loop
                Input (‘C’, I, J , C: integer)
                print  C(I, J)
        end loop;
end loop;
```

…

## Exemplo 3 - Multiplicação de Matrizes em **Linda** C = A*B)

O cálculo de cada Cij é feito por processos trabalhadores, “worker processes”.

- Cada um destes processos obtém o número da tarefa a realizar devolvendo-o de seguida ao TS incrementado de uma unidade.

- O valor do contador corresponde ao elemento a calcular.

- O código é independente do número de processos usado.

…

**Exemplo 3 - Multiplicação de Matrizes em Linda** C = A*B)

```
task body workers is
… // declaração de variáveis
begin
   loop
          Input (“next”, element: integer);
          Output ( “next”, element+1);
          exit when element > 3*3;
          I := (element -1 ) / 3 + 1;      <= nº da linha
          J := (element-1) mod 3 +1;   <=nº da coluna
                //  obtém a linha:
           row_tuple := Read (‘A’, I, v1:vector);
                // obtém a coluna
           col_tuple := Read (‘B’, J, v2:vector);

           x:= inner_product (v1, v2);
           Output (‘C’, I,  J,  X);
   end loop;
end workers;
```

## Exemplos de linguagens com comunicação por mensagens

…

## Problemas possíveis na comunicação por mensagens:

- O processo receptor pode não existir

- As mensagens podem perder-se na emissão ou na recepção

- Pode ocorrer uma situação de excepção durante a execução remota do procedimento