Sistemas Operacionais II N

Algoritmos para exclusão mútua entre ene processos

Algoritmo de Dijkstra (1965)

```
Variáveis: want, inside: array[1..n] of boolean;
    turn: 1..n;
```

Antes:	Depois:
... want[i] := true ini: loop inside[i] := false if not want[turn] then turn := i exit when turn = i endloop inside[i] := true for k:=1 to n st k ≠ i if inside[k] then goto ini endfor	REGIÃO CRÍTICA want[i] := false inside[i] := false ...

Algoritmo de Eisenberg e McGuire (1972)

```
Variáveis: want, inside: array[1..n] of boolean;
    turn: 1..n;
```

Antes: want[i] := true	Depois:

ini: inside[i]:= false;
k := turn
loop
if not want[k]
then k:=(k mod n) + 1
else k:=turn
exit when k == i
endloop
inside[i] := true
for k:=1 to n st k ≠ i
if inside[k] then goto ini
endfor
if turn ≠ i and want[turn]
then goto ini
turn := i
	REGIÃO CRÍTICA
	k:=(turn mod n) + 1
	loop
	exit when want[k]
	k:=(k mod n) + 1
	endloop
	turn := k
	want[i] := false
	inside[i] := false
	...

Algoritmo de Lamport (1974)

```
Variáveis: choosing: array[1..n] of boolean;
           number: array[1..n] of integer;
```

Antes:	Depois:
... choosing[i] := true number[i] := max(number[1]..number[n]) + 1 choosing[i] := false for j:=1 to n st j ≠ i loop exit when not choosing[j] endloop loop exit when number[j] == 0 or (number[i],i) < (number[j],j) endloop endfor	REGIÃO CRÍTICA number[i] := 0 ...

Algoritmo de Peterson (1981)

```
Variáveis: stage: array[1..n] of 1..n-1;
           last: array[1..n-1] of 1..n;
```

Antes:	Depois:
... for j := 1 to n-1 stage[i] := j last[j] : = i for k:=1 to n st k ≠ i loop exit when stage[i] > stage[k] or last[j] ≠ i endloop endfor endfor	REGIÃO CRÍTICA stage[i] := 0 ...

Algoritmo de Block e Woo (1990)

```
Variáveis: want: array[1..n] of 0..1;
           last: array[1..n] of 1..n;
Cada processo: stage: integer
```

Antes:	Depois:
... stage:= 0 want[i] := 1 repeat stage := stage + 1 last[stage] := i loop exit when last[stage] ≠ i or stage = ∑ want endloop until last[stage] == i	REGIÃO CRÍTICA want[i] := 0 ...

Algoritmo de Toscani

Variáveis: want: array[1..n, 1..n] of boolean;
last: array[1..n, 1..n] of integer;
Cada processo: stage: integer

Antes:

```
...
for j := 1 to n st j ≠ i
   want[i,j] := true
   last[min(i,j), max(i,j)] := i
   loop
      exit when not want[j,i]
         or last[min(i,j), max(i,j)] ≠ i
   endloop
endfor
```

Depois:

```
REGIÃO CRÍTICA
for j := 1 to n
   want[i,j] := false
endfor
...
```