SISTEMAS DE INFORMAÇÃO

ENGENHARIA DA COMPUTAÇÃO

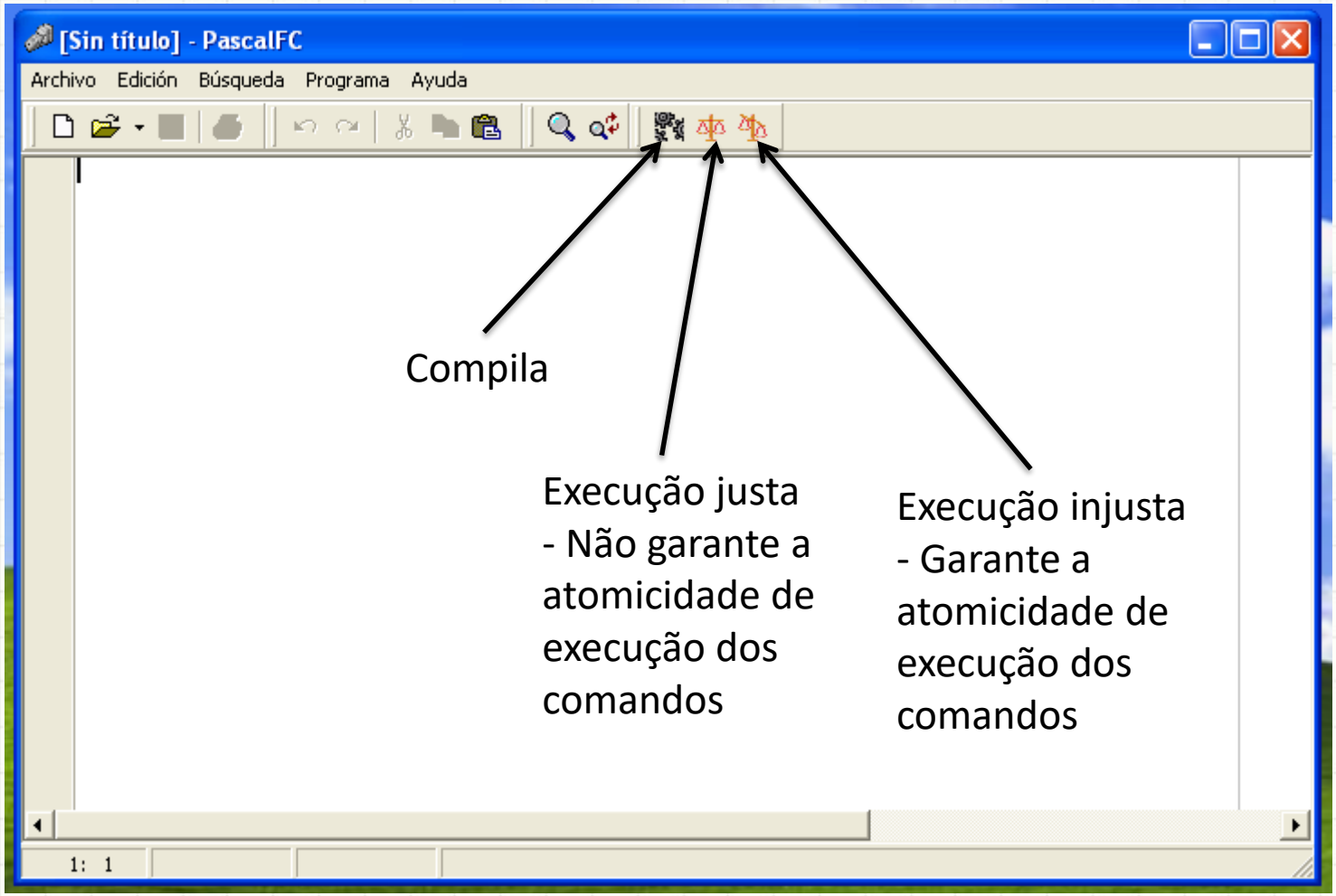
SISTEMAS OPERACIONAIS

Semáforos com PascalFC

Prof. Jean Carlos Hennrichs

UNOESC

Ambiente do PascalFC



Primeiros códigos no PascalFC

```
PROGRAM teste;
   VAR n1,n2 : integer;
BEGIN
   WRITE('Entre com o primeiro número: ');
   READLN(n1);
   WRITE('Entre com o segundo número: ');
   READLN(n2);
   IF (n1 >= n2)
      THEN WRITELN('O maior número é o ',n1)
   ELSE
      WRITELN('O maior número é o ',n2);
END.
```

Primeiros códigos no PascalFC

```
PROGRAM Concorre;
PROCESS Type TP(n : integer);
BEGIN
   WHILE (TRUE) DO
   BEGIN
      sleep(random(10));
      writeln('Processo ',n,' executando!!!');
   END;
END;
VAR
   Proc : ARRAY[1..2] OF TP;
   I : integer;
BEGIN
   COBEGIN
      FOR I := 1 TO 2
        DO Proc[I](I);
   COEND;
END.
```

Primeiros códigos no PascalFC

```
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.
```

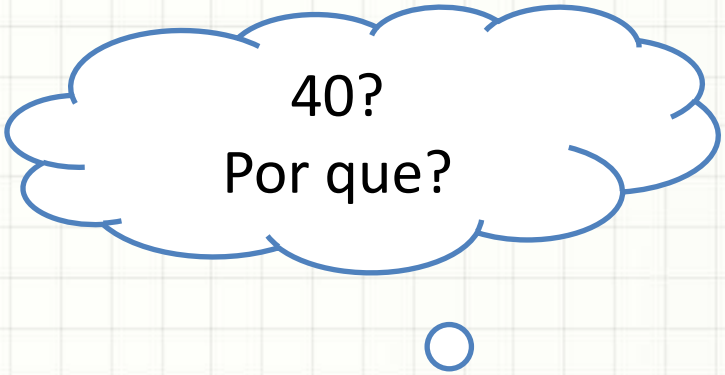
Primeiros códigos no PascalFC

```
1  Program soma;
2  var
3     cont:integer;
4     mutex:semaphore;
5
6  Process P1;
7  Begin
8     Wait(mutex);
9        cont:=cont+10;
10    Signal(mutex);
11 end;
12
13 Process P2;
14 Begin
15    Wait(mutex);
16       cont:=cont+30;
17    Signal(mutex);
18 end;
19
20 Begin
21    initial(mutex,1);
22    cont:=0;
23    cobegin
24       P1;
25       P2;
26    coend;
27    writeln('O valor de cont é:', cont);
28 end.
```

Que valor será impresso?

Primeiros códigos no PascalFC

```
1  Program soma;
2  var
3     cont:integer;
4     mutex:semaphore;
5
6  Process P1;
7  Begin
8     Wait(mutex);
9        cont:=cont+10;
10    Signal(mutex);
11 end;
12
13 Process P2;
14 Begin
15    Wait(mutex);
16       cont:=cont+30;
17    Signal(mutex);
18 end;
19
20 Begin
21    initial(mutex,1);
22    cont:=0;
23    cobegin
24       P1;
25       P2;
26    coend;
27    writeln('O valor de cont é:', cont);
28 end.
```



Primeiros códigos no PascalFC

```
1  Program soma;
2  var
3     cont:integer;
4     mutex:semaphore;
5
6  Process P1;
7  Begin
8     Wait(mutex);
9        cont:=cont+10;
10    Signal(mutex);
11 end;
12
13 Process P2;
14 Begin
15    Wait(mutex);
16       cont:=cont+30;
17    Signal(mutex);
18 end;
19
20 Begin
21    initial(mutex,1);
22    cont:=0;
23    cobegin
24       P1;
25       P2;
26    coend;
27    writeln('O valor de cont é:', cont);
28 end.
```

Agora façamos a seguinte modificação nesse código para verificar a imprevisibilidade de execução das Threads dentro do cobegin/coend.

Antes do comando Wait de P1 e P2 inclua a linha abaixo. No lugar do X coloque o número do 1 ou 2, de acordo com o Process que está editando:

```
sleep(random(3));
```

```
writeln('PX');
```

Depois de cada Wait de P1 e P2, inclua:

```
writeln('-->PX');
```

Compile, Execute e veja através das mensagens a ordem de execução. Talvez precise executar várias vezes para compreender.

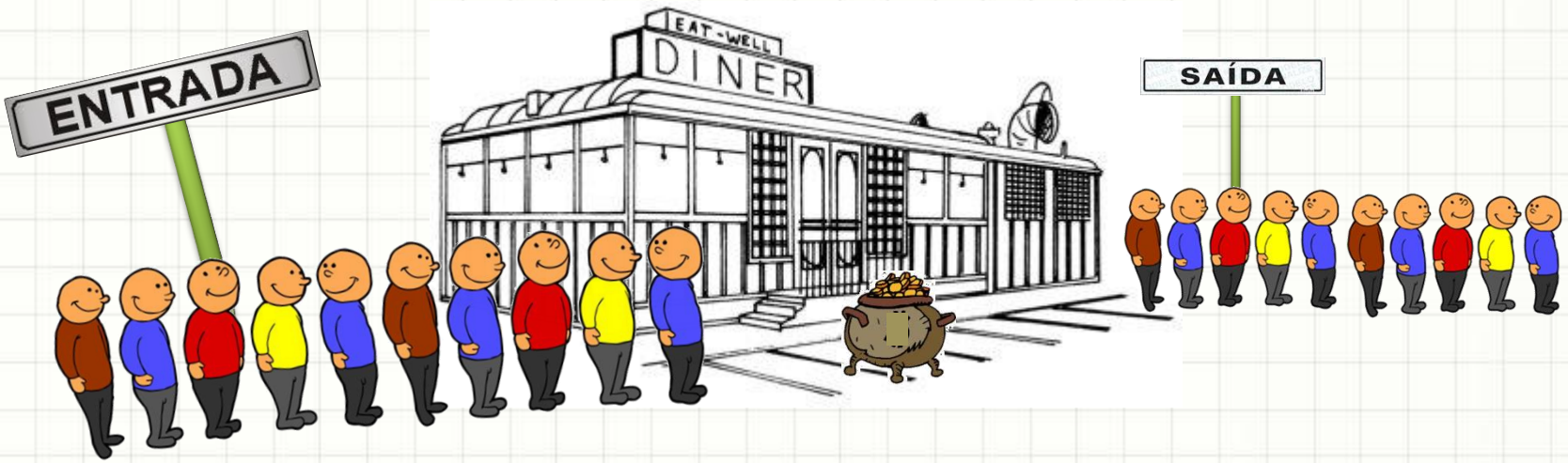
Primeiros códigos no PascalFC

```
1  program produtorConsumidor;
2  var
3     buffer:array[1..5] of integer;
4     cheio:semaphore;
5     vazio:semaphore;
6     i:integer;
7
8  process produtor;
9  var
10    msg,poe:integer;
11 begin
12    repeat
13       msg:=random(20);
14       wait(vazio);
15       poe:=(poe mod 5)+1;
16       buffer[poe]:=msg;
17       writeln('[', buffer[1]:2, ',', buffer[2]:2, ',', buffer[3]:2, ',',
18          buffer[4]:2, ',', buffer[5]:2 ,'] P=', msg:2);
19       sleep(random(5));
20       signal(cheio);
21    forever
22 end;
23
```

Primeiros códigos no PascalFC

```
24 process consumidor;
25 var
26    msg,sai:integer;
27 begin
28    repeat
29       wait(cheio);
30       sai:=(sai mod 5)+1;
31       msg:=buffer[sai];
32       buffer[sai]:=0;
33       writeln('[', buffer[1]:2, ',', buffer[2]:2, ',', buffer[3]:2, ',',
34          buffer[4]:2, ',', buffer[5]:2 ,'] C=', msg:2);
35       sleep(random(3));
36       signal(vazio);
37    forever
38 end;
39
40 begin
41    initial(cheio,0);
42    initial(vazio,4);
43    writeln('===== Inicio =====');
44    writeln('[', buffer[1]:2, ',', buffer[2]:2, ',', buffer[3]:2, ',',
45       buffer[4]:2, ',', buffer[5]:2, ']');
46    cobegin
47       produtor;
48       consumidor;
49    coend;
50 end.
```

Problema do restaurante



Problema: Um restaurante universitário possui apenas 5 mesas com capacidade para apenas 1 pessoa por mesa. Para controlar a entrada há na porta um pote com 5 fichas. Cada estudante que entra pega uma ficha, e almoça. Cada estudante que sai devolve a ficha no pote. Não se entra sem ficha, e na saída é obrigado a devolver a ficha ao pote. As demais pessoas devem esperar na fila caminhando para passar o tempo. Como implementar isso usando concorrência sendo que cada estudante é considerado um thread.

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Semáforo pode ser mesa ou ficha

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Protocolo de Entrada. Faz um DOWN no semáforo

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Região Crítica

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Protocolo de Saída. Faz um UP no semáforo

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Qtd de alunos na fila. Threads que serão criadas

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

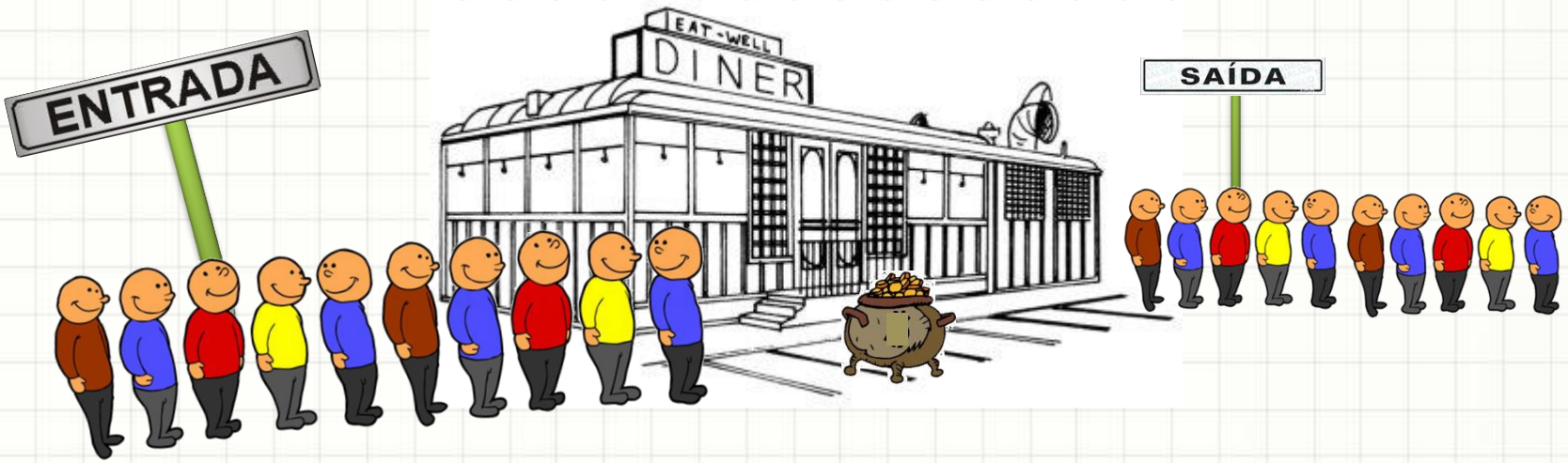
Problema do Restaurante

```
1  Program Restaurante;
2  var
3     mesa:semaphore;
4
5  procedure caminha(n:integer);
6  begin
7     writeln('Aluno ', n, ' caminhando');
8     sleep(random(20));
9  end;
10
11 procedure almoca(n:integer);
12 begin
13    writeln('Aluno ', n, ' almocando');
14    sleep(random(10));
15 end;
16
17 Process type Tpaluno(nid:integer);
18 begin
19    repeat
20       caminha(nid);
21       wait(mesa);
22       almoca(nid);
23       signal(mesa);
24    forever;
25 end;
```

Qtd de mesas. Neste caso é um Semáforo Contador

```
26
27 var
28    N:array[1..20] of Tpaluno;
29    i:integer;
30
31 begin
32    initial(mesa,5);
33    cobegin
34       for i:=1 to 20 do
35          N[i] (i*1);
36    coend;
37 end.
```

Problema do restaurante Xique



Problema: Um restaurante possui apenas 5 mesas com capacidade para apenas 1 pessoa por mesa. Para controlar a entrada há na porta um pote com 7 fichas. Cada pessoa que entra pega uma ficha, vai tomar um aperitivo e aguarda para pegar uma mesa. Assim que pegar uma mesa vai almoçar. Cada pessoa que acaba de almoçar, libera a mesa e vai tomar um café. Após o café, antes de sair, devolve a ficha no pote. Não se entra sem ficha, e na saída é obrigado a devolver a ficha ao pote. Não se almoça sem possuir uma mesa. As demais pessoas devem esperar na fila caminhando para passar o tempo. Implementar isso usando concorrência sendo que cada pessoa é considerado um thread.