Linguagem C

#include <stdio.h>

#include <errno.h>

#include <stddef.h> /\* \_threadid variable \*/

#include <process.h> /\* \_beginthread, \_endthread \*/

#include <time.h> /\* time, \_ctime \*/

#include <stdlib.h> // necessário p/ as funções rand() e srand()

#include <pthread.h>

#include <iostream>

#include <semaphore.h>

int flag = 0;

int vetorRand[5];

int indice = 0;

pthread\_mutex\_t meu\_mutex;

sem\_t meu\_semaforo;

void sort(int vetor[], int n) {

int k, j, aux;

for (k = 1; k < n; k++) {

for (j = 0; j < n - 1; j++) {

if (vetor[j] > vetor[j + 1]) {

aux = vetor[j];

vetor[j] = vetor[j + 1];

vetor[j + 1] = aux;

}

}

}

}

/\*

void \*threadbody(void \*parg)

{

vetorRand[indice] = \*((int\*)parg);

indice++;

flag = 1;

return (NULL);

}

void \*print(void \*parg)

{

// while(indice <= 4);

printf("valor passado de arg:%d\n", \*((int\*)parg));

sort(vetorRand, 5);

for(int i = 0; i<=4; i++)

printf("%d\n", vetorRand[i]);

return (NULL);

}

\*/

void \*teste1(void \*parg)

{

// sem\_wait(&meu\_semaforo);

pthread\_mutex\_lock(&meu\_mutex);

for(int x = 0; x <= 5; x++){

printf("%d\n", x);}

pthread\_mutex\_unlock(&meu\_mutex);

// sem\_post(&meu\_semaforo);

return (NULL);

}

void \*teste2(void \*parg)

{

//sem\_wait(&meu\_semaforo);

pthread\_mutex\_lock(&meu\_mutex);

for(int x = 10; x <= 15; x++){

printf("%d\n", x);}

pthread\_mutex\_unlock(&meu\_mutex);

// sem\_post(&meu\_semaforo);

return (NULL);

}

int main ()

{

pthread\_t threads[6];

pthread\_t teste1x;

pthread\_t teste2x;

int param;

int \*pparam = &param;

pthread\_mutex\_init(&meu\_mutex, NULL);

sem\_init(&meu\_semaforo, 0, 1);

pthread\_create(&teste1x, NULL, teste1, NULL);

pthread\_create(&teste2x, NULL, teste2, NULL);

/\*srand(time(NULL));

for(int i = 0; i <= 4; i++){

param = rand()%100;

pthread\_create(&threads[i], NULL, threadbody, &param);

while(flag == 0);

flag = 0;

}

pthread\_join(threads[4], NULL);

param = 10;

pthread\_create(&threads[5], NULL, print, &param);\*/

pthread\_join(teste1x, NULL);

pthread\_join(teste2x, NULL);

pthread\_mutex\_destroy(&meu\_mutex);

//sem\_destroy(&meu\_semaforo);

system("PAUSE");

return 0; /\* O programa chegará aqui. \*/

}

Linguagem ADA

With Ada.Integer\_Text\_IO, Ada.Text\_IO, Ada.Calendar;

Use Ada.Integer\_Text\_IO, Ada.Text\_IO, Ada.Calendar;

procedure teste\_task is

protected type Resource is

entry Seize;

procedure Release;

private

Busy : Boolean := False;

end Resource;

protected body Resource is

entry Seize when not Busy is

begin

Busy := True;

end Seize;

procedure Release is

begin

Busy := False;

end Release;

end Resource;

X,Z,J:Integer;

control1: Resource;

control2: Resource;

control3: Resource;

indice: Integer:= 1;

i: Integer:= 1;

Y: Integer;

function TesteValor(teste1: in Integer; testeX: out Integer) return Integer;

vetor: array(1..30) of Character;

type vetorInt is array(1..6) of Integer;

vetor1: vetorInt:= (2,2,2,2,2,2);

vetorResource : array(1..6) of Resource;

TesteVetor : array (1..6) of Integer:= (-1,-1,-1,-1,-1,-1);

type valor is access function(teste1: in Integer; testeX: out Integer) return Integer;

function TesteValor(teste1: in Integer; testeX: out Integer) return Integer is

Sum : Integer := 0;

-- teste : valor;

begin

Sum := teste1;

testeX:= (Sum + 10);

return Sum;

end TesteValor;

value : valor := TesteValor'Access;

function TestVetor(vector: in vetorInt) return Integer is

choice : Integer := 0;

-- teste : valor;

begin

choice:=vector(1);

return choice;

end TestVetor;

task type Mostra1 is

--entry mostra(Item: in Integer);

end Mostra1;

task body Mostra1 is

begin

delay 2.0;

while indice < 29 loop

control1.Seize;

--vetorResource(1).Seize;

vetor(indice):='a';

indice:=indice+1;

control2.Release;

--vetorResource(2).Release;

end loop;

end Mostra1;

task type Mostra2 is

-- entry mostra(Item: in Integer);

end Mostra2;

task body Mostra2 is

begin

delay 2.0;

while indice < 30 loop

control2.Seize;

--vetorResource(2).Seize;

vetor(indice):='b';

indice:=indice+1;

control3.Release;

--vetorResource(3).Release;

end loop;

end Mostra2;

task type Mostra3 is

--entry mostra(Item: in Integer);

end Mostra3;

task body Mostra3 is

begin

delay 2.0;

while indice < 31 loop

control3.Seize;

--vetorResource(3).Seize;

vetor(indice):='c';

indice:=indice+1;

control1.Release;

--vetorResource(1).Release;

end loop;

for i in 1..30 loop

Put(vetor(i));

end loop;

end Mostra3;

task type origem is

entry exchange(mens\_in: in integer; mens\_out: out integer);

end origem;

task body origem is

K : integer := 222;

L : integer;

begin

accept exchange (mens\_in : in integer; mens\_out : out integer) do

mens\_out := K;

L := mens\_in;

end exchange;

Put(L);

end origem;

A : Mostra1;

B : Mostra2;

C : Mostra3;

D : origem;

task type destino is

end destino;

task body destino is

N: integer;

M: integer;

begin

D.exchange(20,M);

Put(M);

end destino;

E : destino;

begin

-- vetorResource(2).Seize;

-- vetorResource(3).Seize;

control2.Seize;

control3.Seize;

X:= TesteValor(50,Y);

Z:= TestVetor(vetor1);

--Put(Y);

--Put(X);

--Put(Z);

Put\_Line("Teste finalizado!!");

-- D.exchange(20, Z);

-- Put(Z);

end teste\_task;