# TÓPICOS ESPECIAIS – SISTEMAS EMBARCADOS
## *PROGRAMAÇÃO PARALELA E TEMPO REAL*

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

# Sobre threads POSIX

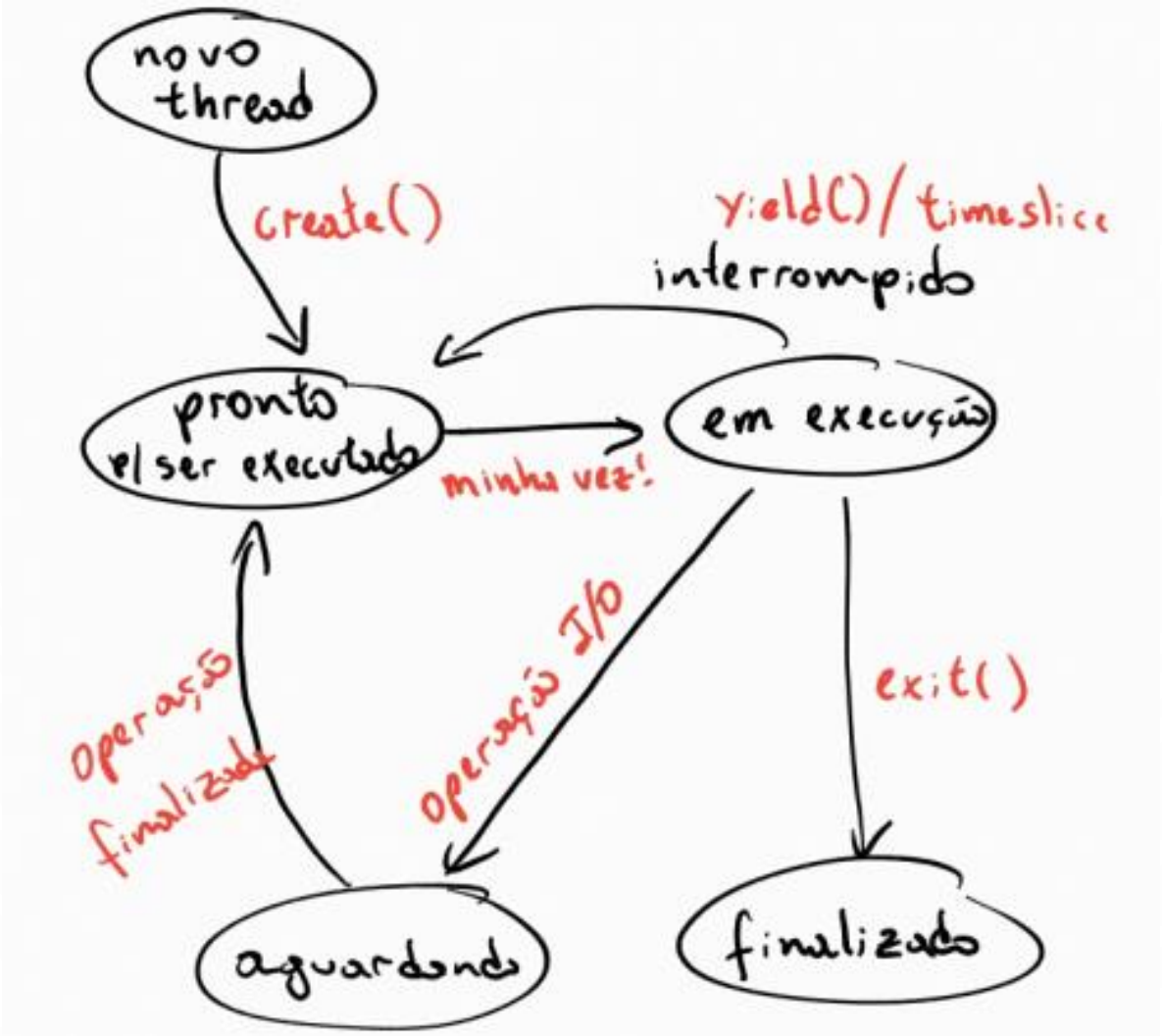
```
#include <pthread.h>
int errCode;

int main() {
     pthread_t tid[MAX]; // array de threads
     /*
          parte sequencial
     */
     // começa a paralelizar
     for(int i = 0; i < MAX; i++) {
          errCode = pthread_create(&tid[i], NULL, <nome_função_a_processar>, (void *) i);
          // se errCode == 0 SUCESSO na criação. Outros códigos em <errno.h>
     }
     // finaliza paralelização juntando resultados
     for (int i = 0; i < MAX; i++) {
          errCode = pthread_join(tid[i],NULL);
     }
```

pthread_t: Handle para pthread, isto é, um valor que permite identificar a thread;

```
int pthread_create(
      pthread_t *tid,
      const pthread_attr_t *attr, //atributos da thread (pode ser NULL)
      void *(*start_routine)(void *) //ponteiro para função a executar,
      void *arg
);
```

A assinatura da função a processar precisa ter retorno void* e parâmetro void*
Exemplo: void* test(void* num); dentro da função faz typecast para o tipo do dado desejado

# Sobre threads POSIX – estados da thread



# Sobre threads POSIX

```
...
    // finaliza paralelização juntando resultados – aguarda thread terminar
    for (int i = 0; i < MAX; i++) {
        errCode = pthread_join(tid[i], NULL);
    }
    Uma vez a thread reunida, não mais existe,
seu thread_id não é mais válido e não pode ser reunida a outra thread

    Poderia ser criado um ponteiro void * thread_res para ser usado no _join
    //errCode = pthread_join(tid[i], &thread_res);
    if errCode != 0 //ERRO NA FINALIZAÇÃO DA THREAD
    else cout << “Thread finalizada com sucesso... com retorno: “ << (char *)thread_res;

Para finalizar explicitamente uma thread

void pthread_exit (void *retval);

ESTE VALOR EM *retval na saída thread é associado ao ponteiro *thread_res de status no JOIN!

Identificador da thread (thread ID) com: pthread_t pthread_self (void);

Para bloquear uma thread, permitindo que outra entre em execução: int sched_yield (void);
```

```
int pthread_join(
     pthread_t tid,
     void **status // status de saída (return_thread)
);
```

- https://github.com/josenalde/parallel_programming_rtos/blob/main/src/pthread_validade_simple.cpp
- https://github.com/josenalde/parallel_programming_rtos/blob/main/src/pthread_validade_simple_2.cpp
- https://github.com/josenalde/parallel_programming_rtos/blob/main/src/pthread_validate.cpp