

Threads em linguagem C/C++



UNIFEI

Universidade Federal de Itajubá

IMC – Instituto de Matemática e Computação

Prof. Carlos Minoru Tamaki

Criação e Término (aula08_ex01.c)

```
#include <stdio.h> /* standard I/O routines */
#include <unistd.h> /* unix standard routines */
#include <pthread.h> /* pthread functions and data structures */
/* function to be executed by the new thread */
void *OLA(void *argumentos) {
    printf("\nOla\n");
    pthread_exit(NULL);
}
int main ( ){
    pthread_t thread;
    int flag;
    printf("A criar uma nova thread\n");
    flag = pthread_create(&thread, NULL, OLA, NULL);
    if (flag!=0) printf("Erro na criação duma nova thread\n");
    Sleep(2);
    return 0;
}
```

Aguardando o término de uma Thread (aula08_ex02.c)

```
#define NUM_THREADS 5
void *funcao(void *argumentos){
    printf("\nOLA\n");
    return (NULL);
}
int main (){
    pthread_t threads[NUM_THREADS];
    int i;
    for(i=0;i<NUM_THREADS;i++)
        pthread_create(&threads[i], NULL, funcao, NULL);
    printf("Thread principal a esperar o termino das threads criadas \n");
    for(i=0;i<NUM_THREADS;i++)
        pthread_join(threads[i],NULL); /* Esperara a junção das threads */
    return 0; /* O programa chegará aqui. */
}
```

Passagem de Argumentos para threads (aula08_ex03.c)

- Passagem de um valor inteiro

```
int x=5;
```

```
//chamada da thread
```

```
pthread_create(&threads[i], NULL, funcao, (void*)&x);
```

```
:
```

```
void * funcao ( void * argumentos ){
```

```
    int valor = * (int *) argumentos;
```

```
    printf("recebi um inteiro: %d \n", valor );
```

```
}
```

Passagem de Argumentos para threads (aula08_ex04.c)

- Passagem de uma String

```
char mesg[ ]="Ola";
```

```
pthread_create(&threads[i], NULL, funcao, (void*)mesg);
```

```
:
```

```
void * funcao ( void * argumentos ){  
    char *message = (char *) argumentos ;  
    printf(" %s ", message );  
}
```

Passagem de Argumentos para threads (aula08_ex05.c)

- Passagem de múltiplos parâmetros usando uma estrutura

```
int x=5 ; float y=2.5
```

```
typedef struct { int a; float b; } ST;
```

```
ST v;
```

```
v.a=x; v.b=y;
```

```
pthread_create(&threads[i], NULL, funcao, (void*)&v);
```

```
:
```

```
void * funcao ( void * argumentos ){
```

```
    ST * in = (ST *) argumentos ;
```

```
    printf("recebi dois valores: %d %f ", in->a, in->b );
```

```
}
```

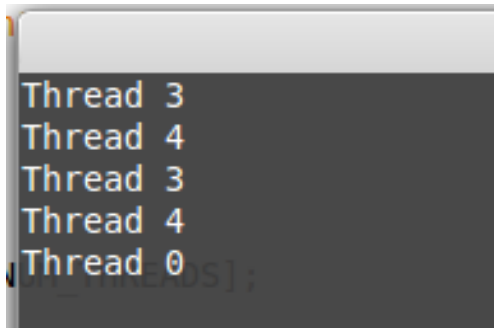
Condições de Corrida na Passagem de Argumentos para threads (aula08_ex06.c)

```
#define NUM_THREADS 5
```

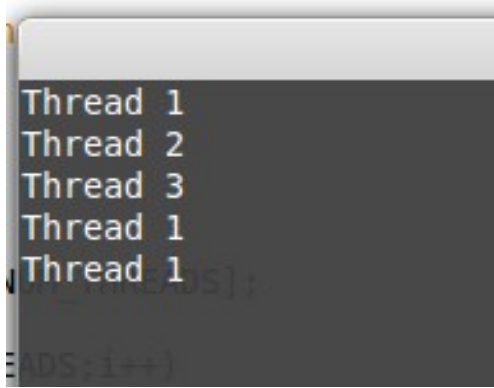
```
void *funcao(void *args){  
    int id;  
    id = *(int *)args;  
    printf("Thread %d\n",id);  
    return (NULL);  
}
```

```
int main (){  
    pthread_t threads[NUM_THREADS];  
    int i;  
    for(i=0;i< NUM_THREADS;i++)  
        pthread_create(&threads[i], NULL, funcao, &i);  
    for(i=0;i< NUM_THREADS;i++)  
        pthread_join(threads[i],NULL);  
    return 0;  
}
```

Prof. Minoru



```
Thread 3  
Thread 4  
Thread 3  
Thread 4  
Thread 0
```



```
Thread 1  
Thread 2  
Thread 3  
Thread 1  
Thread 1
```

A Passagem Correta dos argumentos para identificar uma thread (aula08_ex07.c)

```
void *funcao(void *args) {  
    int x = *(int *)args;  
    printf("Thread %d\n", x);  
    return (NULL);  
}  
  
int main () {  
    pthread_t threads[NUM_THREADS];  
    int i, ids[NUM_THREADS];  
    for (i = 0; i < NUM_THREADS; i++) ids[i]=i;  
    for(i=0;i < NUM_THREADS;i++)  
        pthread_create(&threads[i], NULL, funcao, &ids[i]);  
    for(i=0;i < NUM_THREADS;i++) pthread_join(threads[i],NULL);  
}
```


Condições de corrida (aula08_ex08.c)

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUM_THREADS 2
int saldo = 1000;
void *AtualizaSaldo(void *threadid) {
    int meu_saldo = saldo;
    int novo_saldo = meu_saldo + (long
int)threadid*100;
    printf("Novo saldo = %d\n",
novo_saldo);
    saldo = novo_saldo;
    pthread_exit(NULL);
}
```

```
int main (int argc, char *argv[]){
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;
    for(t=0; t < NUM_THREADS; t++){
        rc = pthread_create(&threads[t],
NULL, AtualizaSaldo, (void*)t);
        if (rc) exit(-1);
    }
    for(t=0; t<NUM_THREADS; t++)
        pthread_join(threads[t], NULL);
    printf("Saldo final = %d\n", saldo);
}
```

Ordem de execução (aula08_ex08.txt)

A ordem de execução não é garantida!

Thread 1	Thread 2	Saldo
Lê saldo: R\$ 1.000		R\$ 1.000
	Lê saldo: R\$ 1.000	R\$ 1.000
	Deposita R\$ 300	R\$ 1.000
Deposita R\$ 200		R\$ 1.000
Atualiza saldo R\$ 1.000 + R\$ 200		R\$ 1.200
	Atualiza saldo R\$ 1.000 + R\$ 300	R\$ 1.300

Sincronização (aula08_ex09.c)

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUM_THREADS 5
pthread_mutex_t meu_mutex =
PTHREAD_MUTEX_INITIALIZER;
int saldo = 1000;
void *AtualizaSaldo(void *threadid) {
    pthread_mutex_lock(&meu_mutex);
    saldo = saldo + (long int)threadid*100;
    printf("Novo saldo = %d\n", saldo);
    pthread_mutex_unlock(&meu_mutex);
    pthread_exit(NULL);
}
```

```
int main (int argc, char *argv[]){
    pthread_t threads[NUM_THREADS];
    long t;
    for(t=0; t<NUM_THREADS; t++)
        pthread_create(&threads[t], NULL,
AtualizaSaldo, (void *) (t + 1));
    for(t=0; t<NUM_THREADS; t++)
        pthread_join(threads[t], NULL);
    printf("Saldo final = %d\n", saldo);
}
```

Sincronização com MUTEX (aula08_ex09.txt)

- `int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)`
- `int pthread_mutex_lock(pthread_mutex_t *mutex)`
- `int pthread_mutex_unlock(pthread_mutex_t *mutex)`
- `int pthread_mutex_destroy(pthread_mutex_t *mutex)`

Sincronização com SEMÁFOROS (aula08_ex09.txt)

- `int sem_init(sem_t *sem, int pshared, unsigned int value)`
- `int sem_wait(sem_t *sem)`
- `int sem_post(sem_t *sem)`
- `int sem_destroy(sem_t *sem)`

Sincronização com CONDITIONS (aula08_ex09.txt)

- `int pthread_cond_init(pthread_cond_t *cond, const pthread_condattr_t *attr)`
- `int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)`
- `int pthread_cond_signal(pthread_cond_t *cond)`

Sites com Tutoriais de PThreads (aula08_ex09.txt)

- Oracle - The Pthreads Library ▶
- OpenGroup – Pthreads.h ▶
- Multithreaded Programming (POSIX pthreads Tutorial) ▶

Dúvidas?