Universidade Tecnológica Federal do Paraná – UTFPR
Coordenação de Ciência da Computação - COCIC
Ciência da Computação

BCC34G – Sistemas Operacionais
Threads – Programação

Prof. Rogério A. Gonçalves
Prof. Rodrigo Campiolo

Introdução

- POSIX (IEEE 1003.1c) especifica uma API para criação e sincronismo de *threads.*
- A API especifica o comportamento da biblioteca de *threads*, a implementação fica para o desenvolvimento da biblioteca.
- Comum em sistemas operacionais UNIX (Solaris, Linux, Mac OS X).

POSIX e Pthreads

- Os *threads* que usam a API de *thread* POSIX são chamados de `Pthreads`.
- A especificação POSIX determina que os registradores do processador, a pilha e a máscara de sinal sejam mantidos individualmente para cada *thread*.
- A especificação POSIX especifica como os sistemas operacionais devem emitir sinais a `Pthreads`, além de especificar diversos modos de cancelamento de *thread*.

Threads no Linux

- Linux se refere a eles como *tarefas* ao invés de *threads.*
  - O Linux aloca o mesmo tipo de descritor para processos e tarefas.
  - Para criar tarefas-filhas, o Linux usa a chamada ***fork***, baseada no Unix.
  - Para habilitar os *threads*, o Linux oferece uma versão modificada, denominada ***clone***.

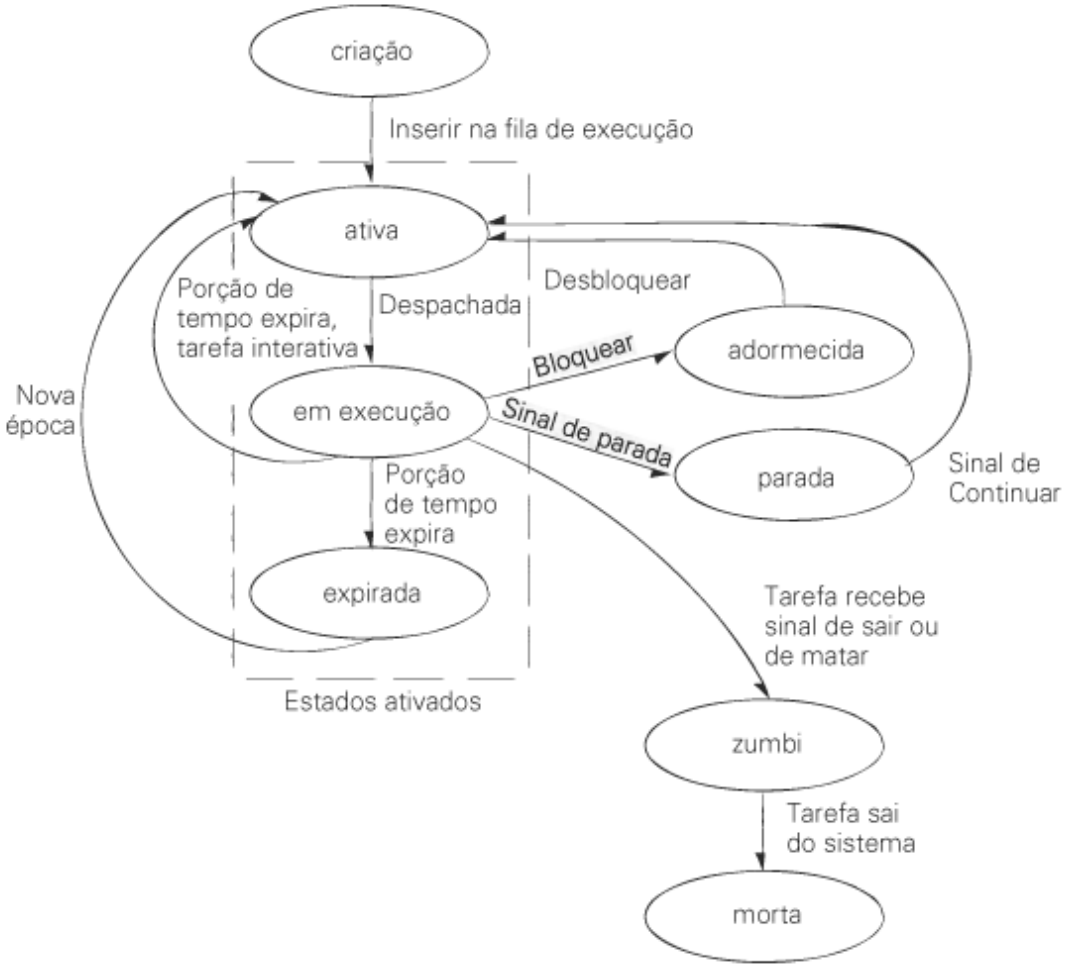
Threads no Linux

- A criação de *thread* é feita por meio da chamada do sistema **clone()**
  - Clone aceita argumentos que determinam os recursos que devem ser compartilhados com a tarefa-filha.
  - **clone()** permite que uma tarefa filha compartilhe o espaço de endereços da tarefa pai (processo)

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

Threads no Linux

Diagrama de transição de estado de tarefa do Linux.



Threads no Linux

- ***pthread_create:*** Cria uma nova *thread*.

```
int pthread_create(pthread_t * thread,
                   const pthread_attr_t * attr,
                   void * (*start_routine)(void *),
                   void *arg);
```

- **Retorno:**
  - 0 se conseguiu criar a *thread*.
  - Ou um código de erro.
    - *Veja: http://linux.die.net/man/3/pthread_create*

Threads no Linux

- ***pthread_join:*** Aguarda pelo término de outra *thread*.

```
int pthread_join(pthread_t th, void **thread_return);
```

- A *thread* principal que cria outras *threads* deve aguardar o término das *threads* filhas.
- Pois pode acontecer da *thread* principal terminar antes.
    - *Veja: http://linux.die.net/man/3/pthread_join*

Threads no Linux

- ***pthread_exit:*** Termina a *thread*.

```
void pthread_exit(void *retval);
```

  - *Veja: http://linux.die.net/man/3/pthread_exit*

Tutorial:
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Threads no Linux

- ***pthread_cancel:*** Cancela a execução de uma *thread*.

```
int pthread_cancel(pthread_t thread);
```

  - *Veja: http://linux.die.net/man/3/pthread_cancel*

Tutorial:
https://computing.llnl.gov/tutorials/pthreads

Threads no Linux

- ***pthread_kill:*** Envia um sinal para a *thread*.

```
#include <signal.h>

int pthread_kill(pthread_t thread, int sig);
```

  - *Veja: http://linux.die.net/man/3/pthread_kill*

Tutorial:
https://computing.llnl.gov/tutorials/pthreads

Threads no Linux

exemplo-thread.c

```
gcc -lpthread exemplo-pthread.c -o exemplo-pthread.exe
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );

int main() {
     /* Declara duas threads */
     pthread_t thread1, thread2;
     char *message1 = "Olá eu sou a Thread 1";
     char *message2 = "Olá eu sou a Thread 2";
     int  iret1, iret2;

     /* Cria duas threads independentes, cada uma irá executar a função */
     iret1 = pthread_create( &thread1, NULL, print_message_function, (void*) message1);
     iret2 = pthread_create( &thread2, NULL, print_message_function, (void*) message2);

     /* Aguarda até que todas as threads completem antes de continuar. */
     /* Pode acontecer de executar algo que termine o processo/thread principal antes das threads terminarem */
     pthread_join(thread1, NULL);
     pthread_join(thread2, NULL);

     printf("Thread 1 retornou: %d\n", iret1);
     printf("Thread 2 retornou: %d\n", iret2);

     exit(0);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
}
```

Atividades

1) Implementar um programa que realize a soma de vetores utilizando threads com a biblioteca pthreads.
2) Implementar um programa multithread com pthreads que calcule a soma de cada linha de uma matriz MxN e devolva o resultado em um vetor de tamanho M. Verifique o tempo de execução do programa para 1 thread, 2 threads, 4 threads e 8 threads.
3) Elaborar um programa para Linux, utilizando a biblioteca pthread, que crie um processo contendo 10 threads. Entre as threads, 50% devem ser CPU bound e 50% devem ser IO bound. Verifique o comportamento das threads (Sugestão: ps xms e ps -Fl).