ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES II

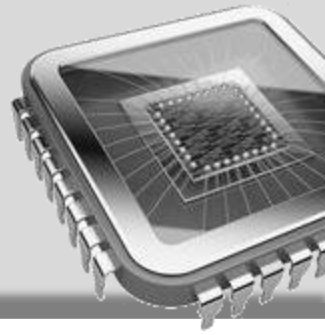
AULA 07: PROGRAMANDO COM THREADS EM LINGUAGEM C

Prof. Max Santana Rolemberg Farias
max.santana@univasf.edu.br
Colegiado de Engenharia de Computação

GESE
Grupo de Engenharia de Sistemas Embarcados

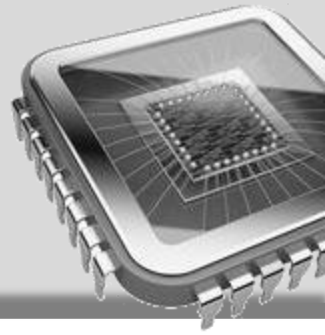
CEComp
Engenharia da Computação
UNIVASF

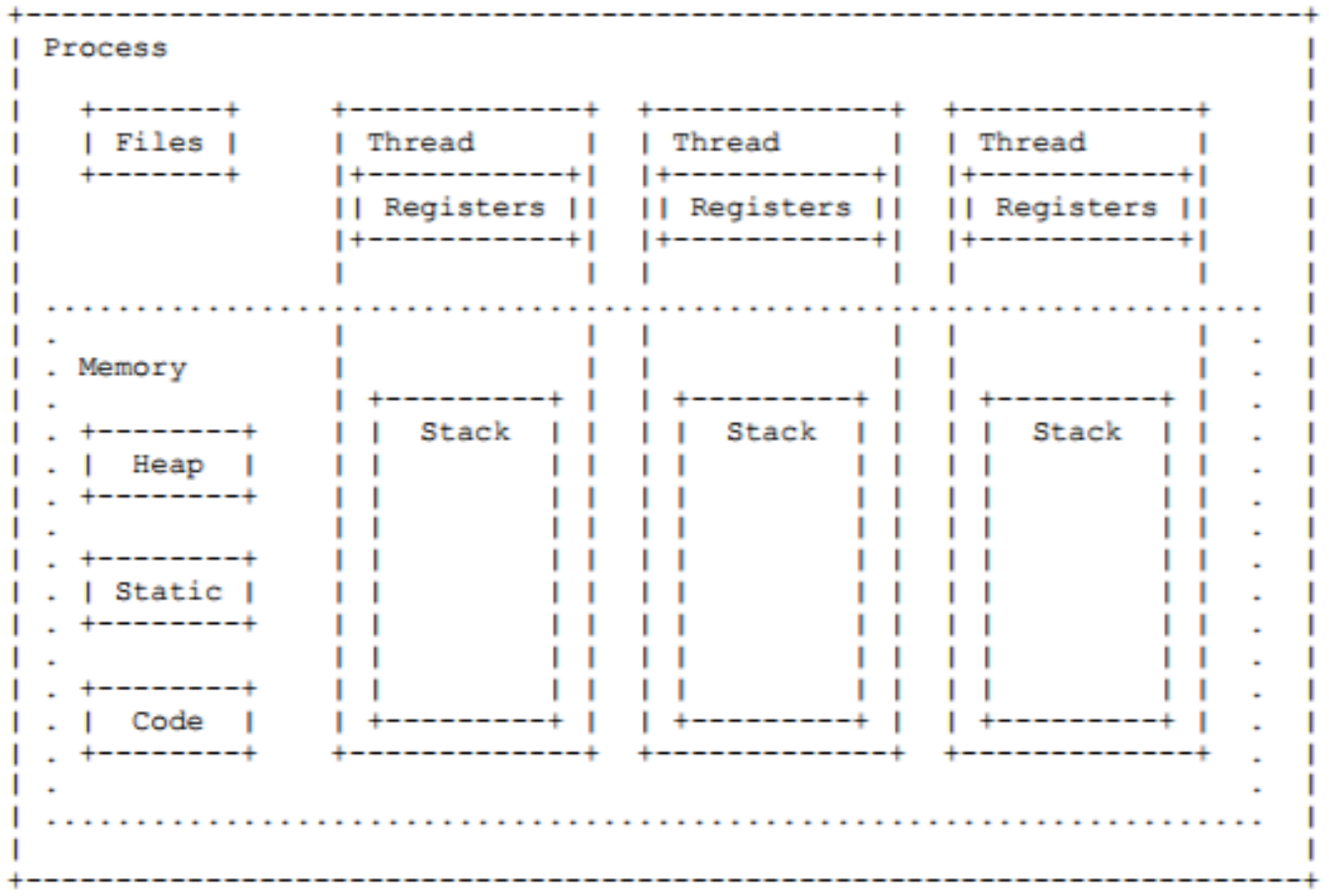
PROGRAMANDO COM THREADS



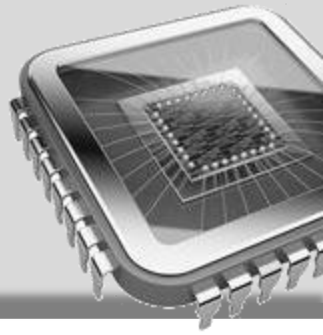
- Podemos usar threads para execução paralela em ambientes multiprocessados com memória compartilhada (SMP).
- Uma thread pode ser definida como um fluxo de instruções (**função**) independente, associado a um processo (**tarefa**).
- Em um programa multi-threaded, diversas funções podem ser selecionadas para execução simultânea.
- Threads **compartilham a mesma memória global** (**dados** e **heap**), mas cada uma possui sua própria pilha

PROGRAMANDO COM THREADS





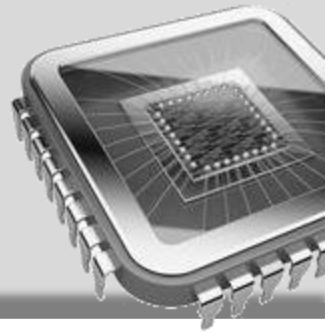
PROGRAMANDO COM THREADS



- Padrão IEEE POSIX 1003.1c
  - Define uma interface comum para manipulação de threads (Pthreads)

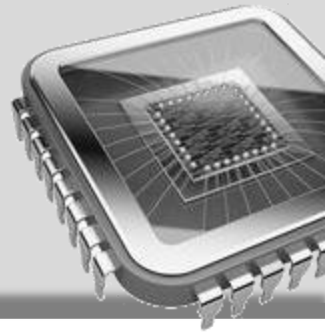
- Implementação dentro do Kernel (kernel-space)
  - Troca de contexto ocorrem sem interferência do usuário ou da aplicação
  - Podem se beneficiar de paralelismo em ambiente multiprocessados .

PROGRAMANDO COM THREADS
POR QUE USAR?



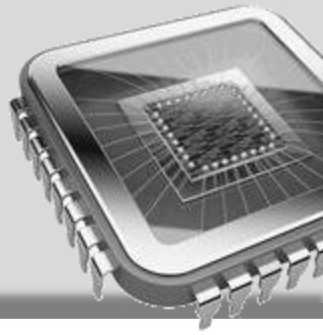
- Velicidade
  - Comunicação entre threads é mais rápida, baseada em memória.
- Economia de recurso
  - Compartilhamento de recurso favorece economia de memória e diminui tempo das operações de criação de threads em relação aos processos.
- Reponsivenesss (Intreatividade)
  - Uso de múltiplas threads em aplicação interativa permite que programas continue respondendo, mesmo que parte esteja bloqueada ou realizando operação demorada.

PROGRAMANDO COM THREADS
POR QUE USAR?



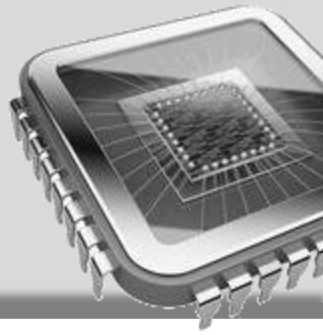
- Desempenho de E/S (I/O throghput)
  - E/S tradicional bloqueia processos
  - Como threads, apenas a thread responsável pela operação ´é bloqueada, permitindo a sobreposição de processamento com E/S.
- Portabilidade
  - POSIX threads são portáveis
  - Bibliotecas para threads são **emuladas em sistemas monoprocessados**. **Em computadores paralelos permite explorar o uso de todos os processadores.**

PROGRAMANDO COM THREADS
QUANDO USAR?



- Para poder se beneficiar do uso de threads, os programa devem ser organizadas em tarefas independentes, que podem ser executadas de maneira concorrente.

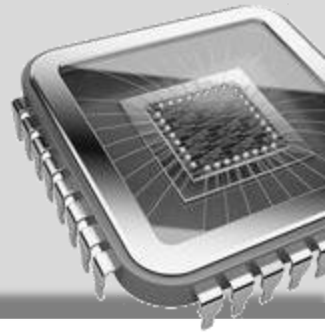
PROGRAMANDO COM THREADS
POSIX THREADS



- POSIX threads é um padrão para threads, o qual define uma API para criar e manipular threads.

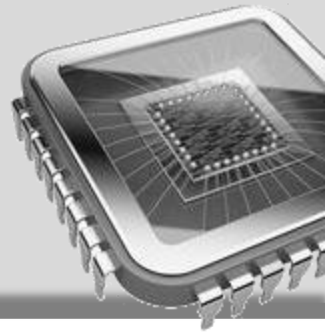
- As bibliotecas que implementam a POSIX threads são chamada Pthreads, sendo muito difundidas no Unix/Linux.

PROGRAMPROGRAMANDO COM THREADS PTHREADS API



- Apresenta 3 grupos de funções:
  - Funções para gerencias threads
    - Tratam da criação e manipulação de threads (criar e destruir) e do ajuste de seus atributos (joinable, scheduling, etc).
  - Funções para sincronizar a execução de threads e bloquear acesso a recursos
    - Mutexes: manipulam um mecanismo de controle de exclusão mútia, mutex e seus atributos.
    - Condition variables: incluem funções para criar, destruir, esperar e tratar sinais baseada em valores de variáveis.
  - Funções para gerenciar o escalonamento de threads.

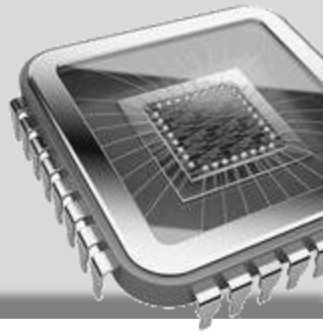
PROGRAMPROGRAMANDO COM THREADS
PTHREADS API: CRIANDO THREADS



```
int pthread_create(pthread_t thread, const pthread_attr_t
      atributo, (void*)*funcao(void*), void* param)
```

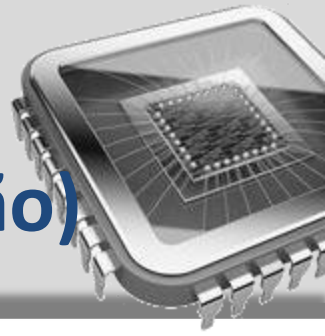
- **thread**: Objeto thread usado no programa;
- **atributo**: Atributo da thread. Se NULL, a thread usa os valores padrão;
- **funcao**: Ponteiro para a função que implementará as funcionalidades da thread;
- **param**: Parâmetro simples da função que implementará as funcionalidades da thread;
- **return**: Sucesso se zero, caso contrário retorna código do erro.

PROGRAMPROGRAMANDO COM THREADS
CRIANDO THREADS: ATRIBUTOS



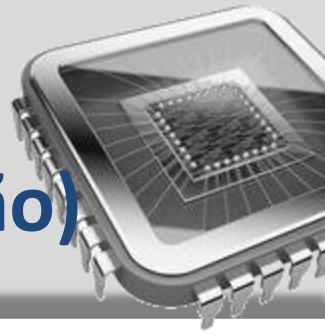
- A estrutura ***pthread_attr_t*** pode ser ajustada e passada como parâmetro para a criação de threads.
  - **Detachstate**
    - **PTHREAD_CRIATE_JOINABLE** (*default*), ou **PTHREAD_CREATE_DETACHED**
    - Os parâmetros ***joinable*** permite que outras threads esperem pela conclusão de uma thread, recuperando a condição de saída.
    - Quando ***detached***, recursos de uma thread são liberados imediatamente em sua conclusão e ***pthread_join*** não pode ser usado para sincronização (espera pelo fim).
    - ***pthread_detach()*** permite ajustar estado para ***detached*** em tempo de execução.

PROGRAMPROGRAMANDO COM THREADS
CRIANDO THREADS: ATRIBUTOS (continuação)



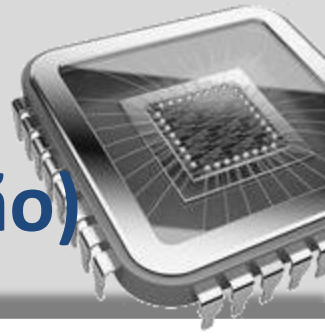
- A estrutura ***pthread_attr_t*** pode ser ajustada e passada como parâmetro para a criação de threads.
  - **Schedpolicy**
    - Seleciona a politica de escalonamento para a thread
    - **SCHED_OTHER**: Escalonamento normal, não tempo-real, é a política default
    - **SCHED_RR**: *Realtime, round-robin*
    - **SCHED_FIFO**: *Realtime, first-in first-out*
    - Política SCHED_RR e SCHED_FIFO são permitidas apenas para processo com privilégios de super usuário.
    - Políticas de escalonamento pode ser alterada em tempo de execução com o comando ***pthread_setschedparam()***

PROGRAMPROGRAMANDO COM THREADS
CRIANDO THREADS: ATRIBUTOS (continuação)



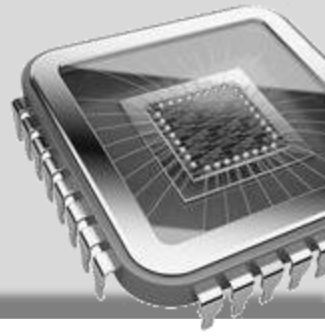
- A estrutura ***pthread_attr_t*** pode ser ajustada e passada como parâmetro para a criação de threads.
  - **Schedparam**
    - Contém parâmetros de escalonamento: Prioridade (***default=0***)
    - É relevante apenas para as políticas **SCHED_RR** e **SCHED_FIFO**
    - Pode ser alterada em tempo de execução como o comando ***pthread_setschedparam()***

PROGRAMPROGRAMANDO COM THREADS
CRIANDO THREADS: ATRIBUTOS (continuação)



- A estrutura ***pthread_attr_t*** pode ser ajustada e passada como parâmetro para a criação de threads.
  - **Inheritsched**
    - Indica se a política e prioridade de escalonamento são definidas pelos parâmetros, ou herdadas da main thread
    - **PTHREAD_EXPICIT_SCHED** e **PTHREAD_INHERIT_SCHED (***default)*
  - **Scope**
    - Define o âmbito de contenção (concorrência) da thread pelo uso de CPU
    - **PTHREAD_SCOPE_SYSTEM**: Thread compete pela CPU com todos os outros processo sendo executados no sistema (valor *default*)
    - **PTHREAD_SCOPE_PROCESS**: Thread compete apenas com outras threads do mesmo processo.

PROGRAMPROGRAMANDO COM THREADS
PTHREADS API: FINALIZANDO THREADS



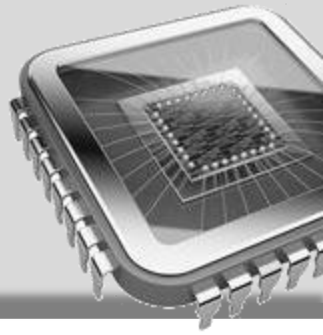
```
int pthread_exit(void* status)
```

- **status**: Status de finalização. Geralmente NULL.

A função ***pthread_exit()*** é usada para finalizar a thread de forma explícita.

Se a função ***main*** finaliza com a chamada de ***pthread_exit()***, todas as outras threads continuam seu processamento. Caso contrário, as outras threads são encerradas.

PROGRAMPROGRAMANDO COM THREADS
PTHREADS API: SINCRONIZANDO THREADS

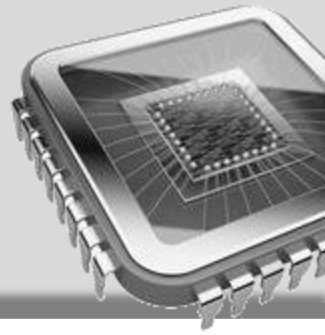


```
int pthread_join(pthread_t thread, void* *status)
```

- **thread**: Objeto thread a qual deseja esperar finalização;
- **status**: Estado de retoerno da thread especificada no primeiro parâmetro.
- **return**: Zero se sucesso ou código do erro

- ***Join*** é um mecanismo para sincronizar as threads.
- Bloqueia quem chamou o ***join*** até a thread alvo seja finalizada;
- O programador pode obter o estado de finalização da thread através do atributo status na chamada da função ***pthread_exit(status)***.

PROGRAMPROGRAMANDO COM THREADS
PTHREADS API: EXEMPLO 1



```
gcc hello.c –o hello pthread
```

```
msrfarias:~/workspace $ gcc hello.c -o hello -pthread
hello.c: In function ‘PrintHello’:
hello.c:21:4: warning: format ‘%d’ expects argument of type ‘int’, but argument 2 has type ‘long int’ [-Wformat=]
    printf(" ID %d\n", tid );
    ^
msrfarias:~/workspace $ ./hello
 ID 0
 ID 1
 ID 3
 ID 2
 ID 4
Thread 0 terminou
Thread 1 terminou
Thread 2 terminou
Thread 3 terminou
Thread 4 terminou
COUNT 36459
msrfarias:~/workspace $
```