Cinco mecanismos:

-> Pthreads (**mutexes, variáveis de condição e barreiras**)

-> Processos (**memória compartilhada e semáforos**)

**Mutexes:**

-> usados para garantir exclusão mútua em regiões críticas (semelhantes a semáforos binários).

->Chamadas:

- criação: pthread\_mutex\_init()

- uso: pthread\_mutex\_lock(), pthread\_mutex\_unlock(), pthread\_mutex\_trylock()

- destruição: pthread\_mutex\_destroy()

* Criando um mutex:

1 - pthread\_mutex\_t mtx = PTHREAD\_MUTEX\_INITIALIZER;

2 -pthread\_mutex\_t mtx;

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

trylock = trava o mutex caso esteja livre, ou retorna imediatamente (sem bloquear), devolvendo EBUSY.

* evita bloqueio
* é preciso ter cuidado com condições de disputa

RCS relacionados devem estar protegidas pelo mesmo mutex

* acesso aos mesmos dados compartilhados

**Variáveis de condição:**

* Mecanismo de sincronização entre threads.
* Usadas em conjunto com mutexes.
  + criação: pthread\_cond\_init()
  + destruição: pthread\_cond\_destroy()
  + uso: pthread\_cond\_wait(), pthread\_cond\_signal(), pthread\_cond\_broadcast()

Pthread\_cond\_wait():

* bloqueia a thread até que cond seja sinalizada.
* O acesso à variável de condição cond deve estar protegido por mutex.
  + mutex é automaticamente destravado quando a thread bloqueia.
  + quando a condição for sinalizada, a thread retoma a execução com mutex travado para seu uso.
  + é preciso destravar mutex ao final da região crítica.

Pthread\_cond\_signal():

* **Acorda uma thread bloqueada por uma variável de condição.**
  + deve ser invocada com mutex travado.
  + thread sinalizada só retoma execução depois que o mutex for destravado por quem executou pthread\_cond\_signal()

Pthread\_cond\_broadcast():

* **acorda todas as thread bloqueadas por uma variável de condição**.

SE NENHUMA THREAD ESTIVER BLOQUEADA ESPERANDO PELA CONDIÇÃO, A SINALIZAÇÃO É PERDIDA.

* SEMELHANTE A SLEEP() E WAKEUP()