MO806/MC914

Tópicos em Sistemas Operacionais

2s2006

Mutex locks e variáveis de condição

Semáforos

- Exclusão mútua
- Sincronização

Exemplo de exclusão mútua

Leitor-Escritor

```
semaforo sem_dado = 1;
```

Leitor:

```
while(true)
    wait(sem_dado);
    le_dado();
    signal(sem_dado);
```

Escritor:

```
while(true)
    wait(sem_dado);
    escreve_dado();
    signal(sem_dado);
```

Exemplo de sincronização

Produtor-Consumidor

```
semaforo cheio = 0, vazio = N;
```

Produtor:

```
while (true)
   wait(vazio);
   f = (f + 1) % N;
   buffer[f] = produz();
   signal(cheio);
```

Consumidor:

```
   while (true)
      wait(cheio);
      i = (i + 1) % N;
      consome(buffer[i]);
      signal(vazio);
```

Mutex locks

$\Rightarrow$ Exclusão mútua

- pthread_mutex_lock
- pthread_mutex_unlock

Exemplo de exclusão mútua

Leitor-Escritor

```
mutex_lock lock_dado = 1;
```

Leitor:

```
while(true)
    mutex_lock(lock_dado);
    le_dado();
    mutex_unlock(lock_dado);
```

Escritor:

```
while(true)
    mutex_lock(lock_dado);
    escreve_dado();
    mutex_unlock(lock_dado);
```

Variáveis de condição

$\Rightarrow$ Sincronização

- pthread_cond_wait
- pthread_cond_signal
- pthread_cond_broadcast
- precisam ser utilizadas em conjunto com mutex_locks

Thread 0 acorda Thread 1

```
int s;                    /* Veja cond_signal.c */
```

Thread 1:

```
  mutex_lock(&mutex);
  if (preciso_esperar(s))
     cond_wait(&cond, &mutex);
   mutex_unlock(&mutex);
```

Thread 0:

```
  mutex_lock(&mutex);
  if (devo_acordar_thread_1(s))
     cond_signal(&cond);
  mutex_unlock(&mutex);
```

Thread 0 acorda alguma thread

```
int s;                  /* Veja cond_signal_n.c */
```

Thread i:

```
  mutex_lock(&mutex);
  if (preciso_esperar(s))
     cond_wait(&cond, &mutex);
  mutex_unlock(&mutex);
```

Thread 0:

```
  mutex_lock(&mutex);
  if (devo_acordar_alguma_thread(s))
     cond_signal(&mutex);
  mutex_unlock(&mutex);
```

Thread 0 acorda todas as threads

```
int s;              /* Veja cond_broadcast.c */
```

Thread i:

```
mutex_lock(&mutex);
if (preciso_esperar(s))
   cond_wait(&cond, &mutex);
mutex_unlock(&mutex);
```

Thread 0:

```
mutex_lock(&mutex);
if (devo_acordar_uma_ou_mais_threads(s))
   cond_broadcast(&cond);
mutex_unlock(&mutex);
```

Thread 0 acorda todas as threads

mas algumas delas voltam a dormir

```
int s;                  /* Veja cond_broadcast2.c */
```

Thread i:

```
  mutex_lock(&mutex);
  while (preciso_esperar(s))  /* <===== */
     cond_wait(&cond, &mutex);
  mutex_unlock(&mutex);
```

Thread 0:

```
  mutex_lock(&mutex);
  if (devo_acordar_uma_ou_mais_threads(s))
     cond_broadcast(&cond);
  mutex_unlock(&mutex);
```

Importância do teste com while

- Thread i vai dormir porque uma condição C é verdadeira
- Thread j acorda thread i porque detectou que C é falsa.
- Antes de thread i voltar a rodar, alguma outra thread tornou C verdade novamente.

- Veja o código `teste_cond_wait.c`