MC504/MC514 - Sistemas Operacionais

Leitores e escritores

Islene Calciolari Garcia

Primeiro Semestre de 2016

Objetivos

- Leitura: acesso compartilhado
- Escrita: acesso exclusivo
- Paralelismo $\times$ ausência de starvation

Leitores e escritores

```
semaforo sem_dados = 1;
```

Leitor:

```
while(true)
    wait(sem_dados);
    le_dados();
    signal(sem_dados);
```

Escritor:

```
while(true)
    wait(sem_dados);
    escreve_dados();
    signal(sem_dados);
```

Leitores e escritores

- Problema: apenas um leitor pode fazer acesso ao banco de dados por vez
- Veja o código: l-e-sem-concorrencia.c

Vários leitores simultâneos

```
semaforo sem_dados = 1, sem_nl = 1;
int nl; /* Leitores ativos num dado instante */
```

Leitor:

```
while(true)
    wait(sem_nl);
    nl++; if (nl == 1) wait(sem_dados);
    signal(sem_nl);
    le_dados();
    wait(sem_nl);
    nl--; if (nl == 0) signal(sem_dados);
    signal(sem_nl);
```

Vários leitores simultâneos

Escritor:

```
while(true)
wait(sem_dados);
escreve_dados();
signal(sem_dados);
```

- Problema: os escritores podem morrer de fome
- Veja o código: l-e-starvation.c

Leitores simultâneos com locks e variáveis de condição

Primeira tentativa

```
int nl = 0;       /* Número de leitores ativos */
mutex_t lock_nl; /* Lock para o contador nl   */

mutex_t lock_dados; /* Lock para os dados      */
```

Leitores simultâneos com locks e variáveis de condição

Primeira tentativa

Leitor:

```
mutex_lock(&lock_nl);
nl++;
if (nl == 1) mutex_lock(&lock_dados); /* <= */
mutex_unlock(&lock_nl);
le_dados();
mutex_lock(&lock_nl);
nl--;
if (nl == 0)
  mutex_unlock(&lock_dados); /* <= */
mutex_unlock(&lock_nl);
```

Leitores simultâneos com locks e variáveis de condição

Primeira tentativa

- Problema: Uma thread leitora faz o lock e outra faz o unlock
- Tipos de lock:
  - FAST
  - RECURSIVE
  - ERROR CHECKING
- Reescrever utilizando variáveis de condição!

Leitores simultâneos com locks e variáveis de condição

```
mutex_t lock_cont; /* Lock para os contadores  */
int nl = 0;  /* Número de leitores ativos */
int ne = 0;  /* Número de escritores ativos */

cond_t  cond_escr; /* Escritores esperam */
cond_t  cond_leit; /* Leitores esperam */
```

Veja o código: l-e-broadcast.c

Leitores simultâneos com locks e variáveis de condição

Leitor:

```
mutex_lock(&lock_cont);
while (ne > 0)
  cond_wait(&cond_leit, &lock_cont);
nl++;
mutex_unlock(&lock_cont);
le_dados();
mutex_lock(&lock_cont);
nl--;
if (nl == 0)
  cond_signal(&cond_escr);
mutex_unlock(&lock_cont);
```

Leitores simultâneos com locks e variáveis de condição

Escritor:

```
mutex_lock(&lock_cont);
while (nl > 0 || ne > 0)
  cond_wait(&cond_dados, &lock_cont);
ne++;
mutex_unlock(&lock_cont);
escreve_dados();
mutex_lock(&lock_cont);
ne--;
cond_signal(&cond_escr);
cond_broadcast(&cond_leit);
mutex_unlock(&lock_cont);
```

Leitores e Escritores

Algoritmo proposto em aula por Robson R. S. Peixoto

Leitor:

```
mutex_lock(&lock_dados);
mutex_lock(&lock_nl);
nl++;
mutex_unlock(&lock_nl);
mutex_unlock(&lock_dados);
le_dados();
mutex_lock(&lock_nl);
nl--;
if (nl == 0)
    cond_signal(&cond);
mutex_unlock(&lock_nl);
```

Leitores e Escritores

Algoritmo proposto em aula por Robson R. S. Peixoto

Escritor:

```
mutex_lock(&lock_dados);
mutex_lock(&lock_nl);
while (nl > 0)
  cond_wait(&cond, &lock_nl);
mutex_unlock(&lock_nl);
escreve_dados();
mutex_unlock(&lock_dados);
```

glibc: Leitores e escritores

RWLock

- pthread_rwlock_rdlock(pthread_rwlock_t *rwlock);
- pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);
- pthread_rwlock_unlock(pthread_rwlock_t *rwlock);

- Qual é a política implementada?

Para pensar...

- Se uma thread está com lock de leitura, poderia pedir promoção para lock de escrita? Como isto seria implementado?
  `pthread_rwlock_rd2wrlock(pthread_rwlock_t *rwlock);`
- O que deve(ria) acontecer após a execução deste código?

```
pthread_rwlock_wrlock(&rwlock);
pthread_rwlock_rdlock(&rwlock);
```

Outros problemas semelhantes...

- Banheiro unissex
- Travessia dos babuínos
- Modus Hall