MC514–Sistemas Operacionais: Teoria e Prática

1s2010

Futex - Fast User-Mode Mutex

Implementação de Mutexes

Sinalizador de eventos

```
class event  {
  public:
    event () : val (0) { }
    void ev_signal () {
      ++val;
      futex_wake (&val, INT_MAX); }
    void ev_wait () {
      futex_wait (&val, val); }
  private
    int val;
};
```

Mutex: primeira proposta

```
class mutex {
  public:
   mutex () : val (0) { }
   void lock () {
     int c;
     while ((c = atomic_inc (val)) != 0)
       futex_wait (&val, c + 1); }
   void unlock () {
     val = 0; futex_wake (&val, 1); }
   private:
    int val;
};
```

Mutex: primeira proposta

- atomic_inc como descrito no artigo:
  - Incrementa atomicamente val
  - Retorna valor anterior
- Garante exclusão mútua
- Se a fila não estiver vazia, uma thread irá conseguir pegar o lock após um unlock.
- Se não há espera, a última chamada de sistema é desnecessária
- Livelock
- Overflow (2^{32})

Mutex: segunda proposta

- Significado para `val`
  - 0: unlocked
  - 1: locked, sem espera
  - 2: locked, com espera
- `cmpxchg(var, old, new)`
  - var $\leftarrow$ new se var $==$ old
  - retorna valor de var antes da operação

Mutex: segunda proposta

```
class mutex {
  public:
   mutex () : val (0) { }
  void lock () {
  int c;
  if ((c = cmpxchg (val, 0, 1)) != 0)
    do {
      if (c == 2 || cmpxchg (val, 1, 2) != 0)
        futex_wait (&val, 2);
       } while ((c = cmpxchg (val, 0, 2))!= 0);
   }
```

Mutex: segunda proposta

```
    void unlock () {
      if (atomic_dec (val) != 1) {
        val = 0;
        futex_wake (&val, 1);
      }
    }
  private:
    int val;
};
```