MC504/MC514 - Sistemas Operacionais

Implementação de Mutexes

Islene Calciolari Garcia

Segundo Semestre de 2013

Futex: Prototype

```
long sys_futex (
    void *addr1,
    int op,
    int val1,
    struct timespec *timeout,
    void *addr2,
    int val3);

int syscall(SYS_futex, addr1, FUTEX_XXXX,
            val1, timeout, addr2, val3);
```

FUTEX_WAIT

```
/* Retorna -1 se o futex não bloqueou e
              0 caso contrário */
int futex_wait(void *addr, int val1) {
  return syscall(SYS_futex, addr, FUTEX_WAIT,
                 val1, NULL, NULL, 0);}
```

- Bloqueio até notificação
- Não há bloqueio se `*addr1 != val1`
- Veja o código `ex0.c`

FUTEX_WAKE

```
/* Retorna o número de threads acordadas */
int futex_wake(void *addr, int n) {
    return syscall(SYS_futex, addr, FUTEX_WAKE,
                   n, NULL, NULL, 0);}
```

- Quantas threads acordar?
  - 1
  - 5
  - INT_MAX (todas)
- Veja os códigos `ex1.c` e `ex2.c`

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;
};
```