# Programação Concorrente

## Locks Recursivos

Prof. Eduardo Alchieri

# Locks Recursivos

- Estrutura protegida por um lock (mutex).
- typedef struct estrutura {

```
        mutex_t lock;
        Tipo1 campo1;
        Tipo2 campo2;
        Tipo3 campo3;
} Estrutura;
```

# Locks Recursivos

- Funções
- void funcao1(Estrutura *e) {

```
    mutex_lock(&e->lock);
    Acessa campo1;
    mutex_unlock(&e->lock);
}
```

- void funcao2(Estrutura *e) {

```
    mutex_lock(&e->lock);
     Acessa campo2;
    mutex_unlock(&e->lock);
 }
```

# Locks Recursivos

- void funcao3(Estrutura *e) {

```
    mutex_lock(&e->lock);
    Acessa campo3;
    mutex_unlock(&e->lock);
}
```

- E se funcao3 invocasse funcao1 ?
- void funcao3(Estrutura *e) {

```
      mutex_lock(&e->lock);
      If (...)
        funcao1(e);
     mutex_unlock(&e->lock);
  }
```

# Locks Recursivos

- Soluções:
  - Replicação de código e função auxiliar não atômica
  - Locks Recursivos
    - A mesma thread/processo consegue acesso ao lock, mas threads/processos diferentes são bloqueados

# Locks Recursivos

- Implementação: utilizando locks simples e variáveis condição
- typedef struct {

```
    pthread_t thr;
    cond_t cond;
    mutex_t lock;
    int c;
} rec_mutex_t;
```

# Locks Recursivos

```
int rec_mutex_lock(rec_mutex_t *rec_m) {
 pthread_mutex_lock(&rec_m->lock);
 if (rec_m->c == 0) { /* Lock livre */
      rec_m->c = 1;
      rec_m->thr = pthread_self();
} else{ /* Mesma thread */
  if (pthread_equal(rec_m->thr, pthread_self())){
      rec_m->c++;
  }else { /* Thread deve esperar */
      while (rec_m->c != 0)
          pthread_cond_wait(&rec_m->cond,&rec_m->lock);
        rec_m->thr = pthread_self();
        rec_m->c = 1;
   }
pthread_mutex_unlock(&rec_m->lock);
return 0;
}
```

# Locks Recursivos

```
int rec_mutex_unlock(rec_mutex_t *rec_m) {
        pthread_mutex_lock(&rec_m->lock);
         rec_m->c--;
         if (rec_m->c == 0)
              pthread_cond_signal(&rec_m->cond);
         pthread_mutex_unlock(&rec_m->lock);
         return 0;
}
```

# Locks Recursivos

- Verificação de erros
- int rec_mutex_unlock(rec_mutex_t *rec_m) {

```
        pthread_mutex_lock(&rec_m->lock);
       if (rec_m->c == 0 || !pthread_equal(rec_m->thr, pthread_self())) {
              pthread_mutex_unlock(&rec_m->lock);
             return ERROR;
      }else{
            rec_m->c--;
            if (rec_m->c == 0)
                 pthread_cond_signal(&rec_m->cond);
            pthread_mutex_unlock(&rec_m->lock);
           Return 0;
      }
}
```