Programação Concorrente – Aula 14

Gilmar Gimenes Rodrigues

1 Continuação de Pthreads.

Várias threads dentro do mesmo processo. Eles compartilham:

- Dados estáticos (e no heap).
- Descritores de arquivos.
- PID.
- Diretório corrente.
- Signal handles.

1.1 Criação:

```
int pthread_create(pthread_t *tid,
                   const pthread_attr_t *attr,
                   void * (*func)(void*), void *arg);
```

1.2 Join

```
int pthread_join(pthread_t, void **status);
```

1.3 Self

```
int pthread_t pthread_self(void);
```

1.4 Detach

```
int pthread_detach (pthread_t tid);
```

1.5 Finalizar

```
int pthread_exit(void *status);
```

1.6 Mutexes

```
int pthread_mutex_lock(pthread_mutex_t * mptr);
int pthread_mutex_unlock(pthread_mutex_t * mptr);
```

1.6.1 Inicialização:

Se estático:

```
pthread_mutex_t meu_mutex = PTHREAD_MUTEX_INITIALIZER;
```

se alocado dinamicamente:

```
pthread_mutex_init(...)
```

1.7 Condition variables:

```
int pthread_cond_wait(pthread_cond_t *cptr, pthread_mutex_t *mptr);
```

Precisa ser executado somente pela thread que possui o mutex, i.e., deu lock no mutex.

```
int pthread_cond_signal(pthread_cond_t *cptr);
```

1.7.1 Inicialização:

estática:

```
pthread_cond_t minha_cond = PTHREAD_COND_INITIALIZER;
```

dinâmica:

```
pthread_cond_init(...);

int pthread_cond_broadcast(pthread_cond_t * cptr);
// Acorda todas as threads que estão esperando,
// enquanto o signal acorda apenas uma.

int pthread_cond_timewait(pthread_cond_t *cptr, pthread_mutex_t *mptr,
                          const struct timespec *abstime);
```