# Sistemas Operacionais

Alisson Linhares

CAMPINAS,

2021/1

# Semáforos - 1/3

**#include <semaphore.h>**

- Para incluir a lib semaphore POSIX.

**int sem_wait (sem_t * sem);**

- Para bloquear semáforo ou esperar.

**int sem_post(sem_t *sem);**

- Para Liberar o semáforo.

**sem_init(sem_t *sem, int pshared, unsigned int value);**

- **sem**: semáforo que será inicializado.
- **pshared**: especifica se o semáforo é compartilhado entre processos ou threads. Um valor diferente de zero significa que o semáforo é compartilhado entre processos e um valor zero significa que ele é compartilhado entre threads.
- **value**: valor inicial do semáforo.

**sem_destroy(sem_t *mutex);**

- Para desalocar o semáforo.

# Semáforos - 2/3

```
#include <iostream>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

using namespace std;

sem_t mutex;

void* thread(void* arg) {
   char *id = (char*) arg;
   sem_wait(&mutex);
   cout << "Thread " << id << " entrou na região crítica." << endl;
   sleep(4);
   cout << "Thread " << id << " saindo da região crítica." << endl;
   sem_post(&mutex);
}
```

# Semáforos - 3/3

```
int main() {
   sem_init(&mutex, 0, 1);
   pthread_t t1,t2;
   pthread_create(&t1,NULL,thread,(char*)"A");
   pthread_create(&t2,NULL,thread,(char*)"B");
   pthread_join(t1,NULL);
   pthread_join(t2,NULL);
   sem_destroy(&mutex);
   return 0;
}

// Para compilar e rodar:
// g++ -lpthread main.cpp
// ./a.out
```

# LAB05

# Lab 05:

- Implemente o problema do jantar dos filósofos em C/C++, usando uma threads para modelar cada filósofo.
  - O programa deve ser configurável, permitindo que o professor escolha o número de filósofos, o tempos mínimo/máximo pensando e o tempo mínimo/máximo comendo.
  - User argc/argv para passar os parâmetros.