# Exercício Programa Servidor TCP echo concorrente *multithread*

PSI 2653

Meios Eletrônicos Interativos I

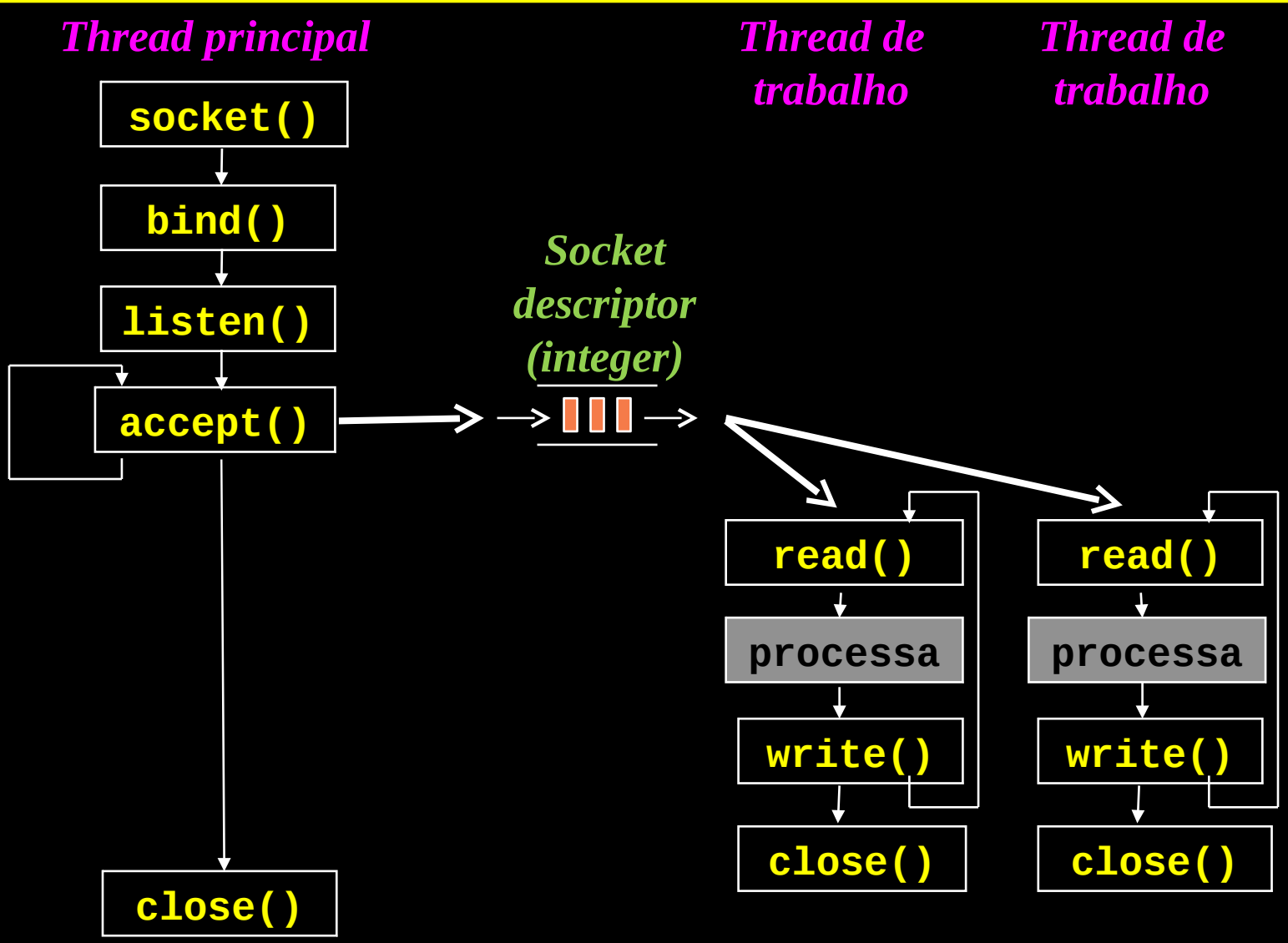
# Servidor TCP echo concorrente

- **Objetivo**
  - Desenvolvimento de um programa servidor TCP echo concorrente *multithreaded* utilizando fila sincronizada por semáforos
- **Composição do grupo**
  - 2 pessoas
- **Formato do trabalho**
  - Papel A4, folhas grampeadas (não encadernar!!)
  - Página de rosto informando:
    - Nome da disciplina, título do trabalho e nome dos autores
- **Entrega:**
  - Data entrega: 2 de junho
  - Entrega do trabalho impresso em sala de aula
  - Execução do programa em sala de aula
  - Serão descontados 2 pontos da nota para cada dia de aula em atraso

# Servidor TCP echo concorrente

- **Ambiente e linguagem**
  - Ambiente Linux
  - Linguagem C
  - Biblioteca pthreads
  - Interface sockets

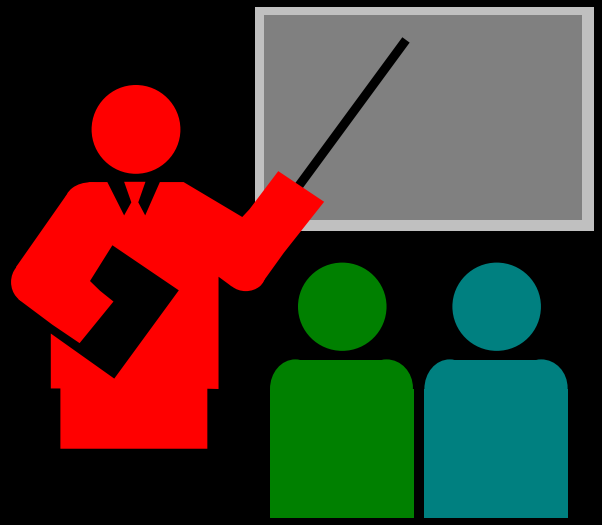
# Servidor TCP echo concorrente



# Servidor TCP echo concorrente

- **Detalhamento**
  - Servidor TCP concorrente *multithreaded*
  - Deve aguardar requisições na porta 2000
  - Dicas para etapas para implementação:
    - 1ª Etapa:
      - Fila sincronizada com semáforo
    - 2ª Etapa:
      - Servidor TCP echo concorrente com integração da fila sincronizada

# 1º Passo:
# Fila sincronizada com semáforo



# 1º Passo: Fila sincronizada com semáforo

- **Implemente uma biblioteca de filas reentrante com sincronização com semáforos.**
- **Deve realizar a sincronização (bloqueio e desbloqueio dos *threads*) no gerenciamento dos recursos (fila cheia → recurso slots; fila vazia → recurso itens).**
- **Utilize as primitivas de semáforos para realização da sincronização.**
- **Observação:**
  - A sincronização deve ser realizada nas funções da biblioteca de fila, não no programa principal !
  - Biblioteca reentrante: Biblioteca que garante a integridade da execução das funções da biblioteca mesmo quando existem acessos concorrentes ou paralelos pelos *threads* às funções da biblioteca.

# 1º Passo: Fila sincronizada com semáforo

- **Primitivas da biblioteca de filas:**

```
int  InitFila(struct fila *F)
int  FilaVazia(struct fila *F)
int  FilaCheia(struct fila *F)
void InserirFila(struct fila *F, int item)
int  RetirarFila(struct fila *F)
```

- **Primitivas de semáforos pthreads:**

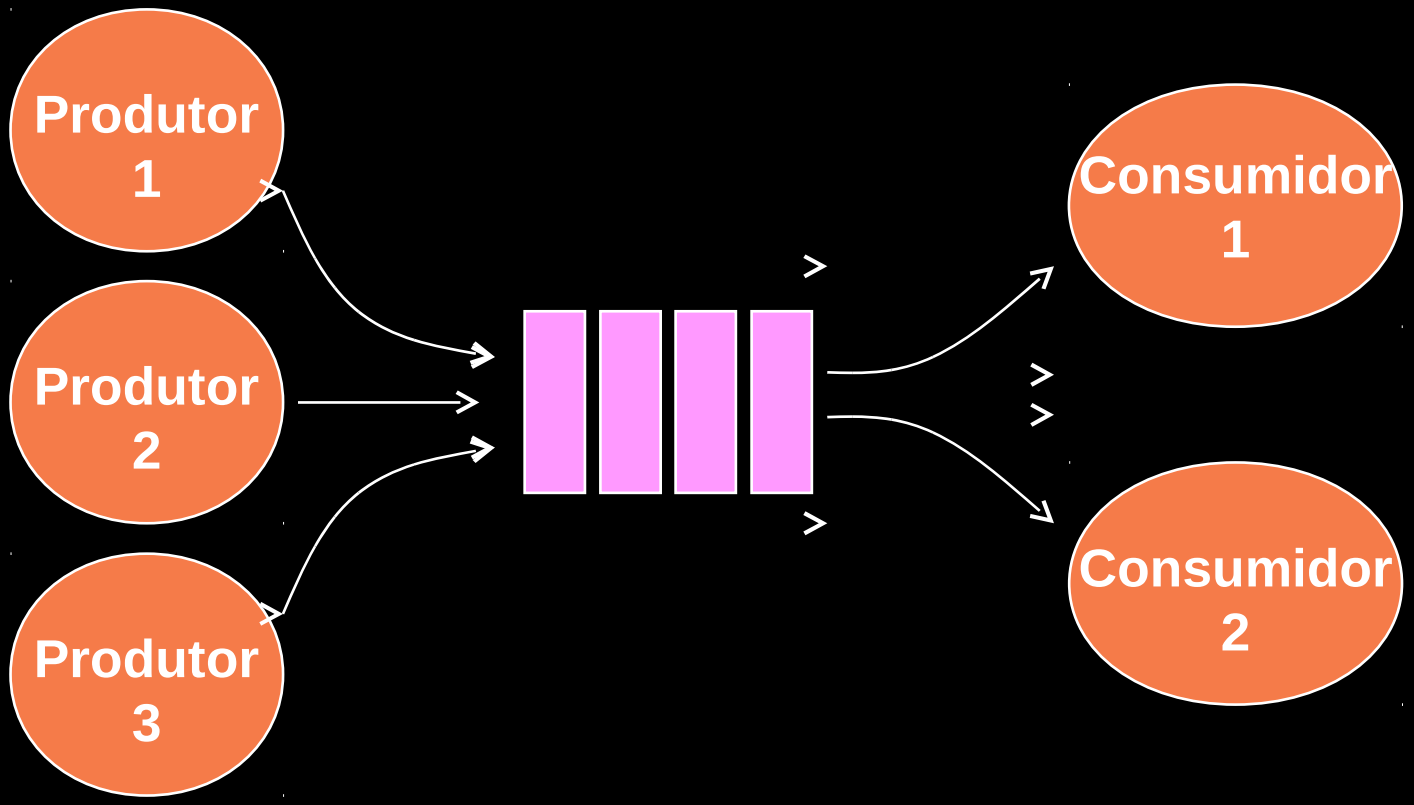
```
#include <semaphore.h>
int sem_init (sem_t *sem, int pshared, unsigned int value)
int sem_wait     (sem_t * sem)
int sem_trywait  (sem_t * sem)
int sem_post     (sem_t * sem)
int sem_getvalue (sem_t * sem, int * sval)
int sem_destroy  (sem_t * sem)
```

# 1º Passo: Fila sincronizada com semáforo

- **Utilizar**
  - 1 semáforo para exclusão mútua
  - 1 semáforo para bloqueio quando fila vazia
  - 1 semáforo para bloqueio quando fila cheia

# 1º Passo: Fila sincronizada com semáforo

- Testar a fila com o programa produtor-consumidor

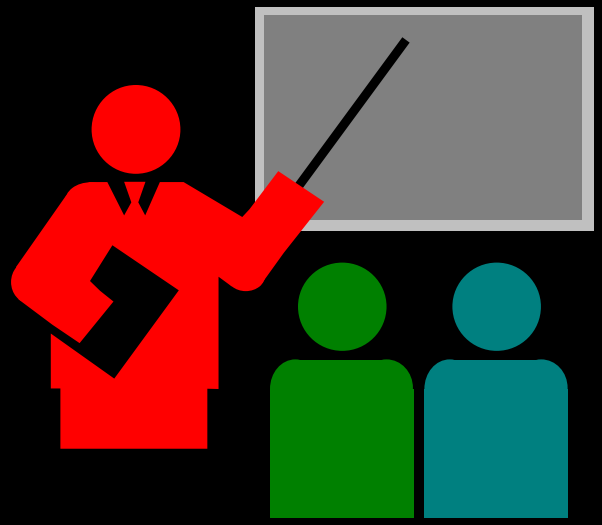


# 1º Passo: Fila sincronizada com semáforo

```
Produtor()
   {
   repetir
      {
      Produzir(E);
      InserirFila(F,E);
      }
   }
```

```
Consumidor()
   {
   repetir
      {
      E = RetirarFila(F);
      Processar(E);
      }
   }
```

# 2º Passo:
# Servidor TCP echo concorrente com integração da fila sincronizada



# 2º passo: Servidor TCP echo concorrente

- Implementação do servidor TCP echo
- Integração da fila sincronizada para repasse do atendimento aos clientes