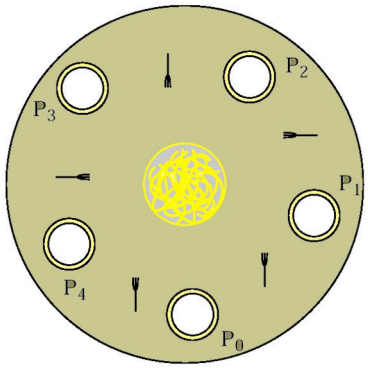
Sistemas Operacionais II N

Monitores

Aula Anterior

- Semáforos
  - Evolução
  - Semântica
  - Implementação
  - Semáforos nomeados POSIX
  - Padrões básicos de uso
  - Aplicações

Problema dos filósofos



Problema dos filósofos

```
#define N 5
#define LEFT(i) (i+N-1)%N
#define RIGHT(i) (i+1)%N
#define THINKING 0
#define HUNGRY 1
#define EATING 2
int state[N];
sema_t mutex; // = 1
sema_t Sem[N]; // = 0
void philosopher(int i) {
  while(TRUE){
    think();
    take_forks(i);
    eat();
    put_forks(i);
    }
  }
void take_forks(int i) {
  sema_wait(&mutex);
  state[i] = HUNGRY;
  test(i);
  sema_post(&mutex);
  sema_wait(&Sem[i]);
  }
```

```
void put_forks(int i)
  {
  sema_wait(&mutex);
  state[i] = THINKING;
  test(LEFT);
  test(RIGHT);
  sema_post(&mutex);
  }

void test(int i)
  {
  if ( state[i] == HUNGRY &&
    state[LEFT(i)]!=EATING &&
    state[RIGHT(i)]!=EATING )
    {
    state[i] = EATING;
    sema_post(&Sem[i]);
    }
  }
```


Problema dos leitores e escritores

```
#include <stdio.h>
#define TRUE 1

sema_t mutex; // = 1
sema_t db; // = 1
int rc = 0;

void writer(void)
  {
  while (TRUE)
    {
    make_data();
    sema_wait(&db);
    write_data();
    sema_post(&db);
    }
  }
```

```
void reader(int i)
  {
  while (TRUE)
    {
    sema_wait(&mutex);
    ++ rc;
    if (rc == 1)
      sema_wait(&db);
    sema_post(&mutex);

    read_database();

    sema_wait(&mutex);
    -- rc;
    if (rc == 0)
      sema_post(&db);
    sema_post(&mutex);
    }
  }
```



```
sema_t mutex; // = 1
sema_t dbw; // = 1
int nr = 0;
int nw = 0;
sema_t mx; // = 1
sema_t dbr; // = 1

void writer(void){
  while (TRUE) {
    sema_wait(&mx);
    ++ nw;
    if (nw == 1)
      sema_wait(&dbr);
    sema_post(&mx);

    sema_wait(&dbw);
    write_data();
    sema_post(&dbw);

    sema_wait(&mx);
    -- nw;
    if (nw == 0)
      sema_post(&dbr);
    sema_post(&mx);
    }
  }
```

```
void reader(int i)
  {
  while (TRUE)
    {
    sema_wait(&dbr);
    sema_wait(&mutex);
    ++ rc;
    if (rc == 1)
      sema_wait(&dbw);
    sema_post(&mutex);
    sema_post(&dbr);

    read_database();

    sema_wait(&mutex);
    -- rc;
    if (rc == 0)
      sema_post(&dbw);
    sema_post(&mutex);
    }
  }
```

```
sema_t mutex; // = 1
sema_t dbw; // = 1
int nr = 0;
int nw = 0;
sema_t mx; // = 1
sema_t dbr; // = 1
sema_t mz; // = 1
void writer(void){
  while (TRUE) {
    sema_wait(&mx);
    ++ nw;
    if (nw == 1)
      sema_wait(&dbr);
    sema_post(&mx);

    sema_wait(&dbw);
    write_data();
    sema_post(&dbw);

    sema_wait(&mx);
    -- nw;
    if (nw == 0)
      sema_post(&dbr);
    sema_post(&mx);
    }
  }
```

```
void reader(int i)
  {
  while (TRUE)
    {
    sema_wait(&mz);
    sema_wait(&dbr);
    sema_wait(&mutex);
    ++ rc;
    if (rc == 1)
      sema_wait(&dbw);
    sema_post(&mutex);
    sema_post(&dbr);
    sema_post(&mz);
    read_database();
    sema_wait(&mutex);
    -- rc;
    if (rc == 0)
      sema_post(&dbw);
    sema_post(&mutex);
    }
  }
```


Hoje

- Monitores
  - Definição
  - Semântica
  - Regras
  - Comparação com Sleep e Semáforos
  - Diferentes Implementações
  - Semáforos em Java
  - Testes

Monitores
(Brinch Hansen-1973, Hoare-1974)

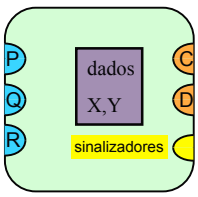

```
monitor meu_monitor;
  X,Y: integer;
  C,D: condition;
procedure P (args)
  { ...
  wait(D);
  ... };
procedure Q (args)
  { ...
  wait(C);
  ... };
procedure R (args)
  { ...
  signal(C);
  ... };
end meu_monitor;
```

Exemplo de monitor conforme Hoare


```
monitor meu_monitor;
  X,Y: integer;
  C,D: condition;
procedure P (args)
  { ...
  wait(D);
  ... };
procedure Q (args)
  { ...
  wait(C);
  ... };
procedure R (args)
  { ...
  signal(C);
  ... };
end meu_monitor;
```

Exemplo de monitor conforme Hoare

meu_monitor;



Regras de todas as implementações

Somente um processo pode estar dentro do monitor a cada instante (exclusão mútua)

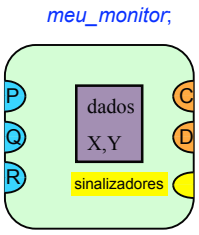
Wait: bloqueia incondicionalmente o processo

Signal: em fila vazia é perdido, não é memorizado e não tem nenhum efeito

Signal: em fila não vazia desbloqueia um processo dessa fila (o primeiro), o qual toma o monitor imediatamente, retirando o processo que sinalizou, que vai para a fila dos sinalizadores, saindo temporariamente do monitor.

```
monitor meu_monitor;
  X,Y: integer;
  C,D: condition;
procedure P (args)
  { ...
  wait(D);
  ... };
procedure Q (args)
  { ...
  wait(C);
  ... };
procedure R (args)
  { ...
  signal(C);
  ... };
end meu_monitor;
```

Diferença para Sleep…




```
public class Semaphore
{
  int value;
  public  Semaphore(int initialValue) { value = initialValue; }
  public synchronized void P() {
    while (value <= 0 ) {
      try {
        wait();
      catch(InterruptedException e) { }
    }
    value--;
  }
  public synchronized void V() {
    p++;
    notify();
  }
}
```

Semáforo com Monitores em Java

Aulas Próxima Semana

- No laboratório 104, prédio 67 (novo)
- Aulas práticas
- Com Daniel Guimarães Jr
  - Resolução de problemas com semáforos
  - Resolução de problemas com monitores