

Processos e Threads

Comunicação Interprocessos

Troca de mensagens, Monitores e Barreiras

Troca de mensagens

- Primitivas para estabelecimento do canal e envio e recepção de mensagens
- Primitivas bloqueantes e não bloqueantes
- Problemas análogos em processos distribuídos

msgget()

```
msqid = msgget (key_t key, int msgflg);
```

- key: an integer usually got from `ftok()` or `IPC_PRIVATE`.
- msgflg:
 - `IPC_CREAT`: used to create a new resource if it does not already exist.
 - `IPC_EXCL | IPC_CREAT`: used to ensure failure of the call if the resource already exists.
 - `rwxrwxrwx`: access permissions.
- returns: `msqid` (an integer used for all further access) on success. `-1` on failure.

Buffer

```
struct msgbuf {  
    long mtype;    /* message type */  
    char mtext[1]; /* message data */  
};
```

- mtype: indica o tipo de mensagem
- mtext: vetor ou estrutura

msgsnd()

```
int msgsnd (int msqid, struct msgbuf *msgp,  
            int msgsz, int msgflg);
```

- msqid: id obtained by a call to msgget.
- msgsz: size of msg text (mtext) in bytes.
- msgp: message to be sent. (msgp->mtype must be positive).
- msgflg: IPC_NOWAIT.
- returns: msgsz on success. -1 on error.

msgrcv()

```
int msgrcv (int msqid, struct msgbuf *msgp,  
           int msgsz, long msgtyp, int msgflg);
```

- msqid: id obtained by a call to msgget.
- msgsz: maximum size of message to receive.
- msgp: allocated by user to store the message in.

msgrcv()

```
int msgrcv (int msqid, struct msgbuf *msgp,  
            int msgsz, long msgtyp, int msgflg);
```

- msgtyp:

- = 0 \Rightarrow get first message on queue.

- > 0 \Rightarrow get first message of matching type.

- < 0 \Rightarrow get message with least type which is \leq $\text{abs}(\text{msgtyp})$.

msgrcv()

```
int msgrcv (int msqid, struct msgbuf *msgp,  
            int msgsz, long msgtyp, int msgflg);
```

- msgflg:
 - IPC_NOWAIT: Return immediately if message not found.
 - MSG_NOERROR: The message is truncated if it is larger than msgsz.
 - MSG_EXCEPT: Used with msgtyp > 0 to receive any msg except of specified type.
- returns : size of message if found. -1 on error.

Monitor

```
monitor example
```

```
integer i;
```

```
condition c;
```

```
procedure producer();
```

```
end;
```

```
procedure consumer();
```

```
end;
```

```
end monitor;
```

Monitor Produtor-Consumidor

```
monitor ProducerConsumer
  condition full, empty;
  integer count;

  procedure insert(item: integer);
  begin
    if count = N then wait(full);
    insert_item(item);
    count := count + 1;
    if count = 1 then signal(empty);
  end;
```

Monitor Produtor-Consumidor

```
function remove: integer;  
begin  
    if count = 0 then wait(empty);  
    remove := remove_item;  
    count := count - 1;  
    if count = N - 1 then signal(full);  
  
end monitor;
```

Produtor-Consumidor em Java

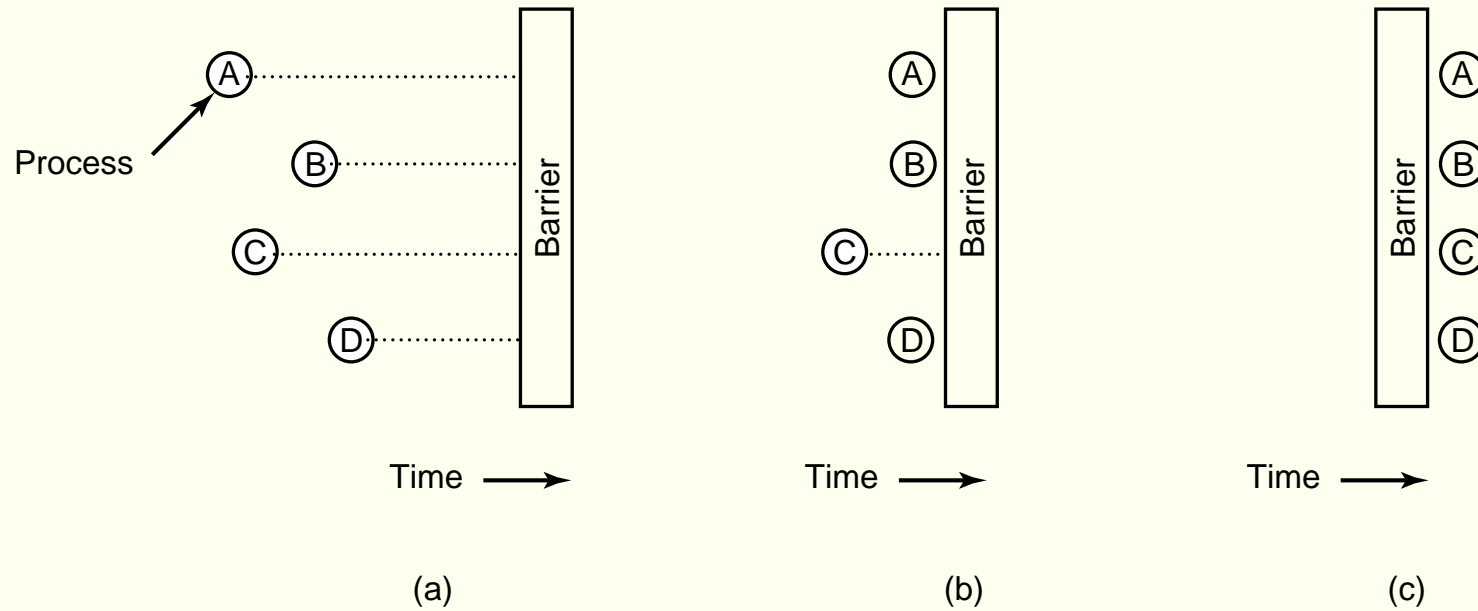
```
static class our_monitor {
    private int buffer[] = new int[N];
    private int count = 0, lo = 0, hi = 0;

    public synchronized void insert(int val) {
        if (count == N) go_to_sleep();
        buffer[hi] = val;
        hi = (hi + 1) % N;
        count = count + 1;
        if (count == 1) notify();
    }
}
```

Produtor-Consumidor em Java

```
public synchronized int remove() {
    int val;
    if (count == 0) go_to_sleep();
    val = buffer[lo];
    lo = (lo + 1) % N;
    count = count - 1;
    if (count == N - 1) notify();
    return val;
}
private void go_to_sleep()
{try{wait();} catch(InterruptedException exc){};}
}
```

Barreiras



- Programação baseada em passos