

STR: aula_7

Trabalho 2 – Estação de lavagem

JAVA concurrency

Planeamento das aulas

- **Semana 21 Set:**
 - **Conceitos básicos de I/O (com experimentação)**
- **Semanas 28 Set, (5 Out), 12 Out:**
 - **1ª parte do trabalho 1 (em C)**
- **Semanas 19 Out, 26 Out:**
 - **2ª parte do trabalho 1 (RTlib)**
- **Semana 2 Nov:**
 - **Exercícios sobre Java**
- **Semanas 9, 16, 23 Nov:**
 - **Trabalho 2: Java em Tempo Real**
- **Semanas 30 Nov, 7 Dez, 14 Dez:**
 - **Trabalho 3 (plc)**



HERE

Concurrency

Computer users take it for granted that their systems can **do more than one thing at a time**. They assume that they can continue to work in a word processor, while other applications download files, manage the print queue, and stream audio. Even a single application is often expected to do more than one thing at a time. **For example, that streaming audio application must simultaneously read the digital audio off the network, decompress it, manage playback, and update its display.** Even the word processor should always **be ready to respond to keyboard and mouse events, no matter how busy it is reformatting text or updating the display.** Software that can do such things is known as *concurrent software*.

Access to critical resources

How to handle

- Define a semaphore for exclusive access

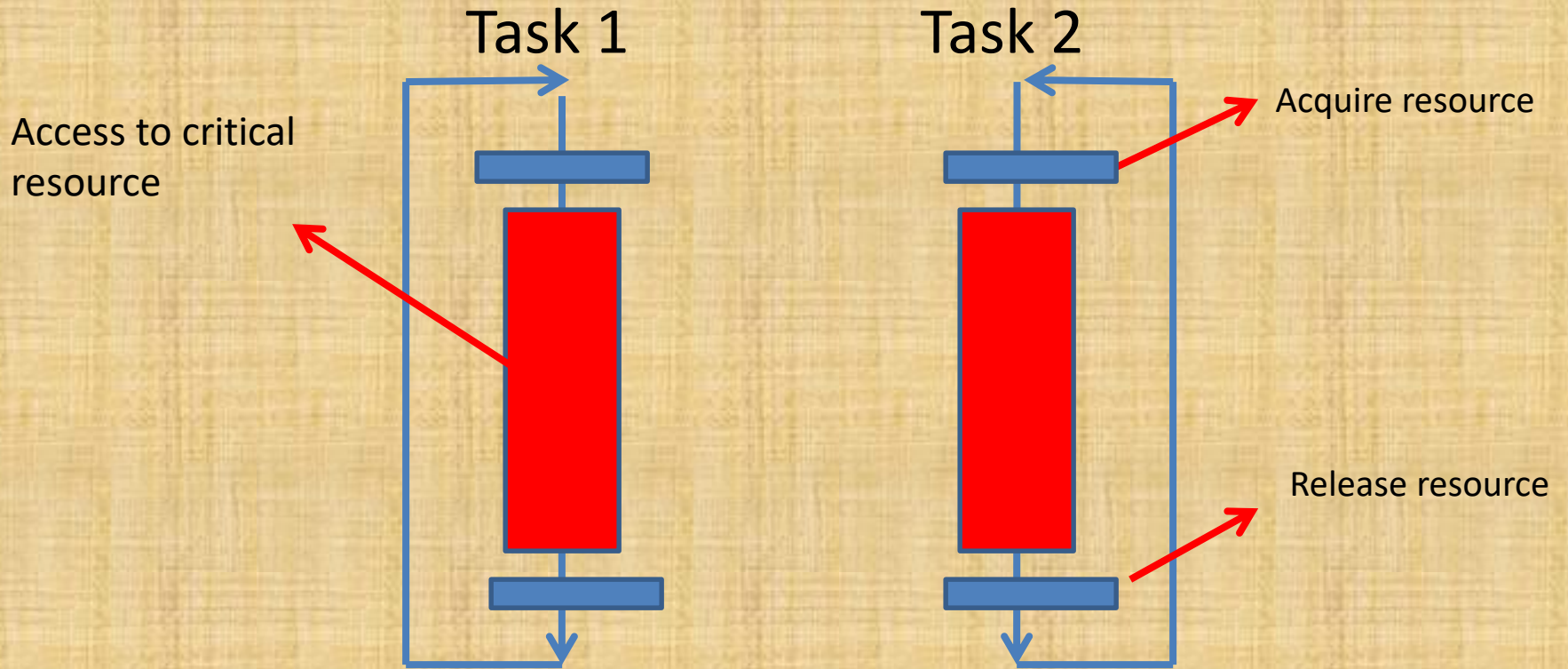
// the second argument defines the number of resources available

// OR number of simultaneous accesses allowed

S3=RTcreateSemaphore(COUNT,1,"S3");

- Perform **RTwait(S3)** just before using the resource
- Perform **Rtsignal(S3)** just after using the resource (so that other tasks can use it)

How it works



Example: Exclusive access to keyboard

```
RTtask(Task1)
{
  while ( true )
  {
    char str[256]= "";
    // other code before
    RTwait(S3); ←
    printf( "\nTask1 str=");
    RTgets(str);
    printf( "\nTask1 got string=%s", str);
    if(str[0]==27)
        terminate_prog=true;
    RTsignal(S3); ←
    // other code after....
  }
}
```

```
RTtask(Task2)
{
  while ( true )
  {
    char str[256]= "";
    RTwait(S3);
    printf( "\nTask2 str=");
    RTgets(str);
    printf( "\nTask2 got string=%s", str);
    if(str[0]==27)
        terminate_prog=true;
    RTsignal(S3);
  }
}
```

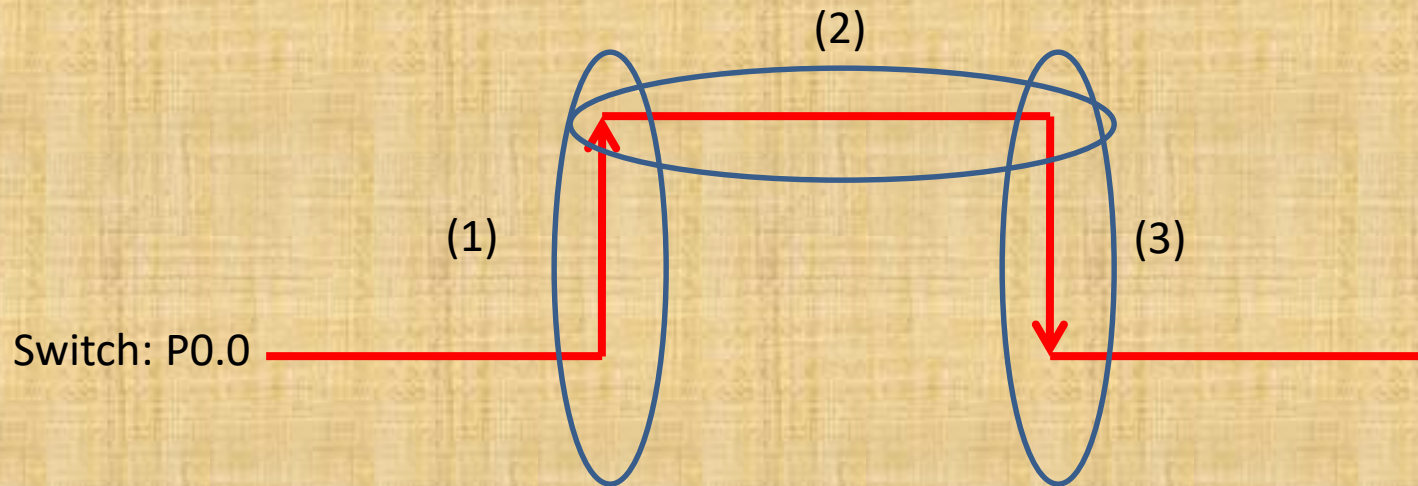
Try it here....



C:\rtlib\demos\vc++6\critical_resource

Events versus states

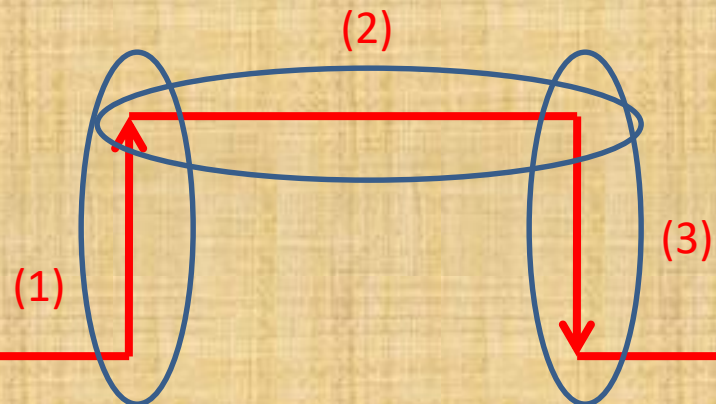
- (1) Detecting change from “zero” to “one”
- (2) Reading the states
- (3) Detecting change from “one” to “zero”



How to detect events

```
RTtask(task1)
{
  int P00_preve=0;
  int P00_curr=0;
  While(true)
  {
    P00_curr= in_port(0) & (1<<0); // read state of bit 0 of port 0 (2)
    if(P00_curr && ! P00_prev){
      // p00 has been changed to "one" (1)
      //....
    }
    if(!P00_curr && P00_prev){
      // p00 has been changed to "zero" (3)
      //.....
    }
    //IMPORTANT:
    P00_prev = P00_curr; // save current_state as previous_state
    RTdelay(1)
  }
}
```

Switch: P0.0



Several messages in a mailbox

The problem is that a mailbox only stores the address of the records

```
// sending a message
TRequest req = new TRequest();
req->x=0;
req->z=0;
req->op=PUT;
Rtput(mbx1, req);
```

```
// Receiving a message
TRequest *req = (Trequest *)RTget(mbx1);
// handle the request
//goto_xz(...);
//put_piece(..)...
delete(req);
```

```
// Emptying a mailbox
Int success;
do {
    RTgetTimed(mbx1,1,success); // RTgetTimed(mBox,timeout,success)
} while(success);
```