STR: aula_4

Trabalho 1 – parte

RTLib – A realtime kernel library

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)

RtLib – Realtime kernel library

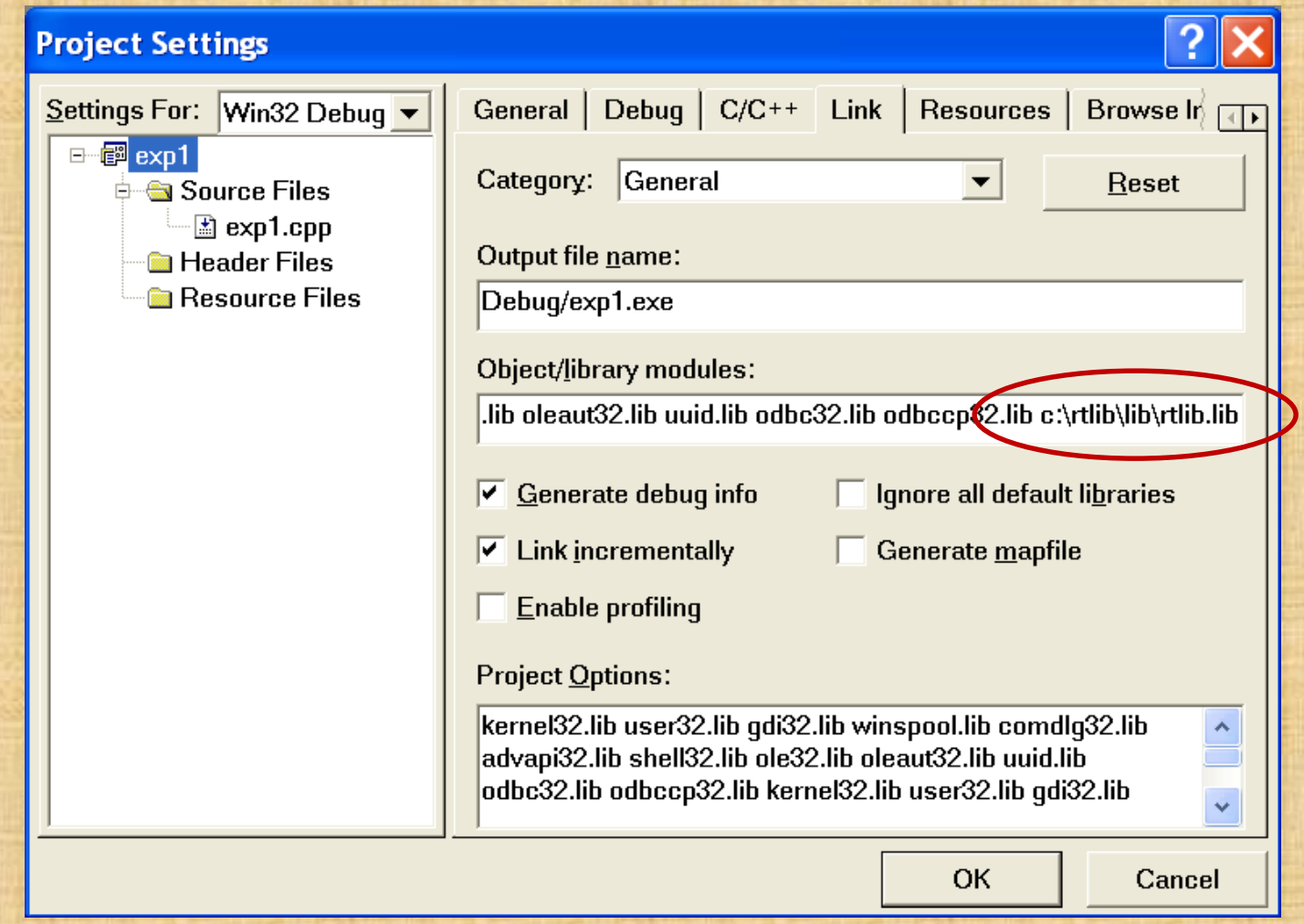
Basic functionality

- **Concurrent tasks creation**
- **Sinchronization with SEMAPHORES**
- **Message Sending/receiving with MAILBOXES**

- **Works in a cooperative fashion – So, every task should allow other tasks to run by invoking Rtdelay(int miliseconds), or any synchronization or mailbox functions**

Accessing RtLib

Include library c:\rtlib\lib\rtlib.lib



RtLib – Realtime library

Add the header file "c:\rtlib\lib\kernel.h"

```
#include"c:\rtlib\lib\kernel.h"
#include<stdio.h>
#include <stdlib.h>
#include <conio.h>


RTtask(Task1)
{
    printf("\nTask1: Hello...");
    while(1)
    {
        int r = 100.0*rand()/RAND_MAX;
        printf("\n\t\tTask1 working..... %02x",r);
        RTdelay(500);
    }
```

Example: Simple task

```
#include"c:\rtlib\lib\kernel.h"
#include<stdio.h>
#include <stdlib.h>
#include <conio.h>

RTtask(Task1) {
   printf("\nTask1: Hello...");
   while(1)   {
      int r = 100.0*rand()/RAND_MAX;
      printf("\n\t\tTask1 working..... %02x",r);
      RTdelay(500);
   }
}

int main(int argc, char **argv) {
   InitRTkernel();
   RTcreateTask(Task1,2,"Task1");
   int t;
   do{
         t=RTgetkey();
   } while(t!=27);
   return 0;
}
```

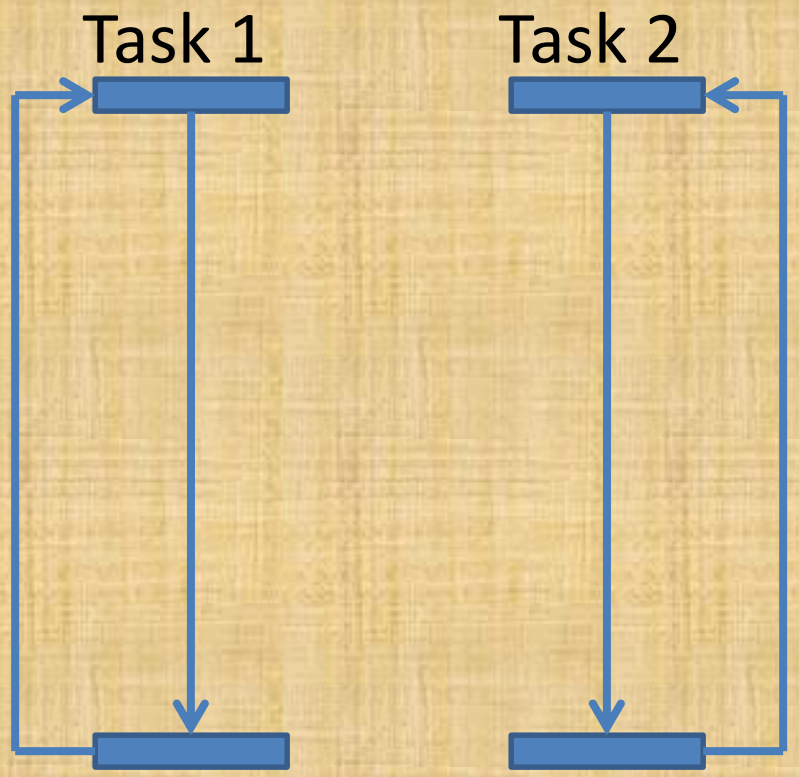
Tasks run indefinidely until main function finishes (see below)

Kernel initialization & Task creation

Goes through a infinite look that if interrupted, finalizes the program

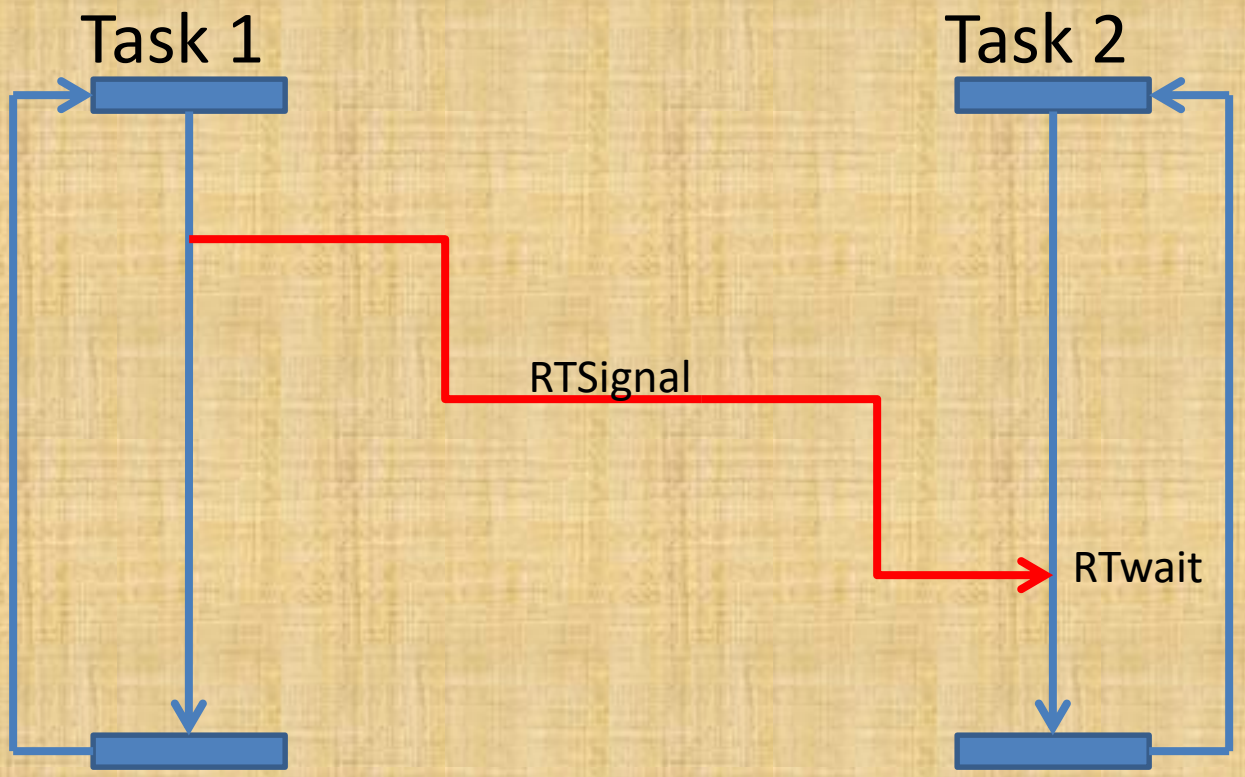
Two parallel tasks

- Two paralel tasks running independently



Two parallel tasks with SINCHRONIZATION

- Two paralel tasks running independently



Recomendações

- **Implementar tudo no simulador**
- **Implementar primeiro as funções movimento em X e Z (e y apenas depois desses eixos)**
- **Reutilizar as funções:**
  - **Goto(x,z) – necessita de move_x, move_z ....**
  - **Put_piece() , get_piece – necessita de goto_xz(), move_y ….**
- **Testes no kit real periodicamente para verificar consistência com resultados simulados**
- **Testes no kit real só podem ser efectuados na presença do respectivo professor/assistente/monitor.**