



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

Sistemas Embarcados: (ELF74)

Prof: DaLuz



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

REENTRÂNCIA

Definição:

- ☐ Funções **reentrantes** são aquelas que **podem** ser chamadas por **mais** de uma **tarefa** e ainda assim, **sempre** trabalham **concorrentemente**, mesmo que o **RTOS** **chaveie** de uma tarefa para outra no **meio** da execução da função.



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

REENTRÂNCIA

Regras:

- ☐ Uma função **reentrante** não pode usar variáveis de uma forma não atômica, a menos que elas estejam armazenadas na **pilha** da tarefa que chamou a função ou são variáveis **privadas** desta **pilha**.
- ☐ Uma função **reentrante** não pode chamar outra função que não seja ela própria uma função **reentrante**.
- ☐ Uma função **reentrante** não pode usar o **hardware** de uma forma não atômica.



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

REENTRÂNCIA

Regras:

- ☐ A melhor maneira de compreender a **reentrância** e em particular a primeira regra anterior, é entender onde os compiladores C armazenam as variáveis. Saber:
 - ☐ **Pilha** x posição fixa de **memória**.

REENTRÂNCIA

Regras:

- ❏ **Static int** – é armazenada em uma localização física da memória e portanto é compartilhada por qualquer tarefa.
- ❏ **Public int** – A única diferença da static é que as funções de outros módulos podem acessar public int.
- ❏ **Var. Globais e Var. inicializada** - uma variável está na memória, pode estar na pilha para variáveis locais ...
- ❏ **Ponteiro** – localização fixa na memória, portanto compartilhada ... Se uma função alterar o valor ...
- ❏ **Parâmetros** – Estes são criados na pilha, portanto várias tarefas podem executar a chamada da função.
- ❏ **Ptr_parm** – Estes são criados na pilha, portanto em teoria podem ser reentrantes, mas se a função alterar o valor dele, temos que verificar onde isso se encontra, pois pode gerar problemas.
- ❏ **Local** – Está na pilha.



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

REENTRÂNCIA

Regras:

Não Reentrante:

```
int g_var = 1;

int f()
{
    g_var = g_var + 2;
    return (g_var);
}

int g()
{
    return (f() + 2);
}
```

Reentrante:

```
int f(int i)
{
    return (i + 2);
}

int g(int i)
{
    return (f(i) + 2);
}
```



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

PARALELISMO

Técnicas:

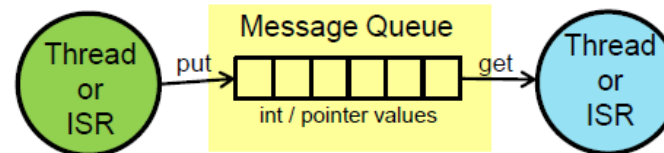
- 📖 **Comunicação** entre tarefas.
- 📖 **Sincronização** entre tarefas
- 📖 **Compartilhamento** de Recursos e Exclusão Mútua
- 📖 Técnicas de **Agendamento / Escalonamento** de tarefas

COMUNICAÇÃO

Entre tarefas:

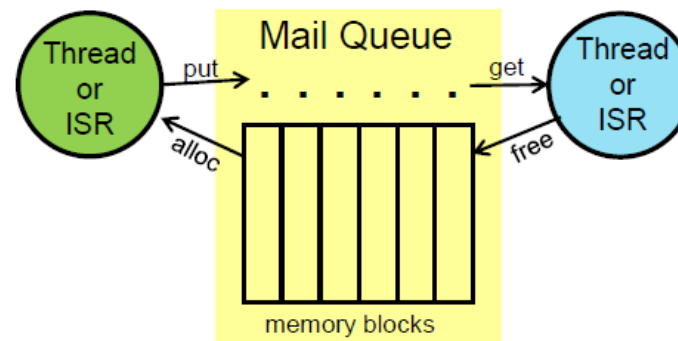
Fila de **Números** ou **Ponteiros**:

Message: Integer or Pointer



Fila de **Mensagens**:

Mail: Memory Blocks

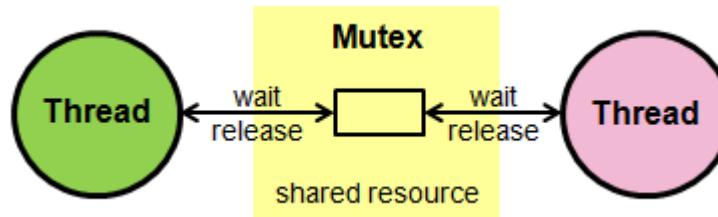


- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

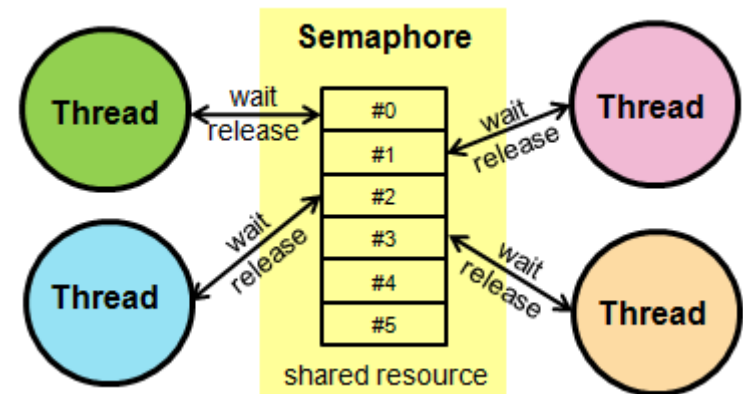
SINCRONIZAÇÃO

Entre tarefas:

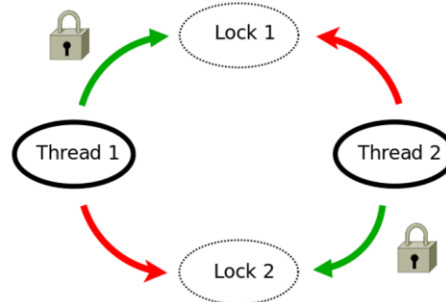
 **Mutex:**



 **Semáforo:**



 **Risco:**



UTFPR

- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

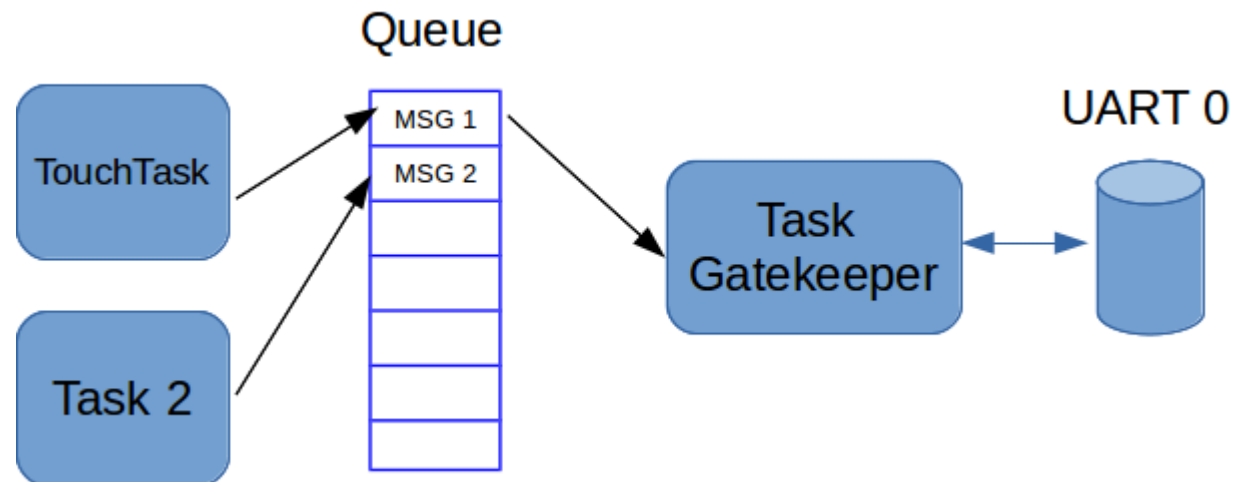


- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

SINCRONIZAÇÃO

GateKeeper:

- ☐ Acesso ao **recurso** é efetuado por uma **única** tarefa.
- ☐ Tarefas que pretendam **acessar** o recurso comunicam com a *Gatekeeper*, e.g. via mensagens.





- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

FreeRTOS

Software:



DownLoad:

<https://code.visualstudio.com/download>



DownLoad: (Obs: Usar Extensão do VSCode)

<https://platformio.org/>

Ref. *



- Reentrância
- Paralelismo
- Comunicação
- Sincronização
- FreeRTOS
- Referências

Referências:

Continuação dos Labs ...

* Refs ↔ Renesas.com, Pixabay.com, wikimedia.org, flickr, community.arm.com, Undergraduated course Renesas / CWS71-Prof. Douglas P. B. R. e Robson L., ytchannel Gustavo W. D., *ARMv7-M Architecture Reference Manual*, CSW40-Sistemas Microcontrolados – Prof. Guilherme P., toshiba.semicon-storage.com, microncontrollerslab.com, lfelectronics.com.br, elf74-Prof. Hugo V. N., stm.st.com, jblopen.com, microsoft.com.