

## Prática laboratório de informática – Threads e Parâmetros main().

Exercício 01: Leia o artigo abaixo transcrito da revista Mecatrônica Atual. Alguns trechos foram omitidos por razões da avaliação, mas o escopo geral foi mantido para não prejudicar a leitura e entendimento. As questões desta avaliação são baseadas no artigo.

### ARTIGO: Ethernet Industrial - Parte 2 - O problema do determinismo da Ethernet (Publicado na revista Mecatrônica Atual - Nº18 - Nov/04)

Como atender as requisições de resposta em tempo real das aplicações industriais se a Ethernet não é determinística? Este é o grande desafio da Ethernet Industrial.

*“A Ethernet Industrial é puro marketing, não é?”. Perguntou-me, durante um almoço, um engenheiro de vendas de uma grande multinacional de Automação, depois de ter lido o primeiro artigo desta série. “Como assim?”. Respondi, meio espantado, para logo ouvir em seguida: “Não é somente uma moda para vender uma Ethernet mais ‘robusta’ para o pessoal de fábrica?”. Pensei comigo mesmo: “Ou ele leu o artigo e não entendeu, ou meu artigo não foi claro”. E, em seguida, expliquei-lhe que a utilização da Ethernet para o chão de fábrica visa substituir as redes industriais utilizadas em sistemas de controle que automatizam processos produtivos. Comentei que, além disso, a grande vantagem da Ethernet Industrial são as tecnologias de TI como, por exemplo, os protocolos da Internet, que também migrarão ao chão de fábrica. Dessa forma, a mesma estrutura de rede e de protocolos poderá ser utilizada desde o ambiente de escritórios até o ambiente industrial permitindo a integração total da corporação empresarial. “Mas a Ethernet Industrial ainda é uma tecnologia emergente e algumas dúvidas persistem. Muito trabalho ainda resta, mas é inegável seu potencial”, finalizei minha explicação para nos dedicarmos às suculentas picanhas.*

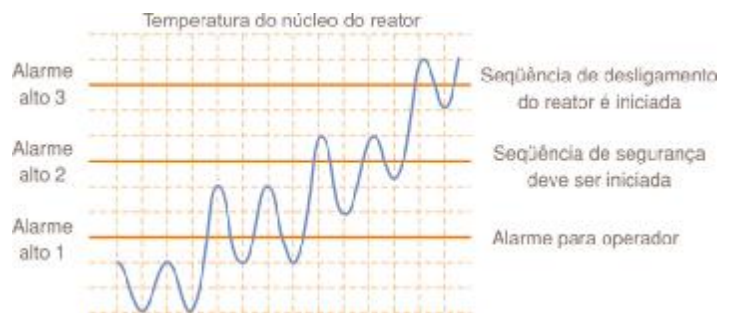
### Requisito de Tempo Real no Ambiente Industrial

Os próximos parágrafos vão parecer tediosos devido à teoria envolvida. Contudo, se o leitor perseverar até o final, com certeza, compreenderá bem qual é a principal restrição da Ethernet no chão de fábrica: a falta de determinismo!

A resposta em tempo real é a principal característica dos sistemas de controle do ambiente industrial. Sistemas de controle são empregados em diversos segmentos industriais como, por exemplo, geração de energia, química, manufatura, automobilística, indústria de bebidas, etc. para coordenar, monitorar e registrar as condições de máquinas, produtos e processos. Vamos tomar um exemplo de um sistema de controle de um reator nuclear. A temperatura do núcleo do reator deve ser constantemente monitorada para evitar sobreaquecimento e conseqüente derretimento e escape de radiação. Para tanto, três níveis de alarme são especificados (**figura 1**) e,

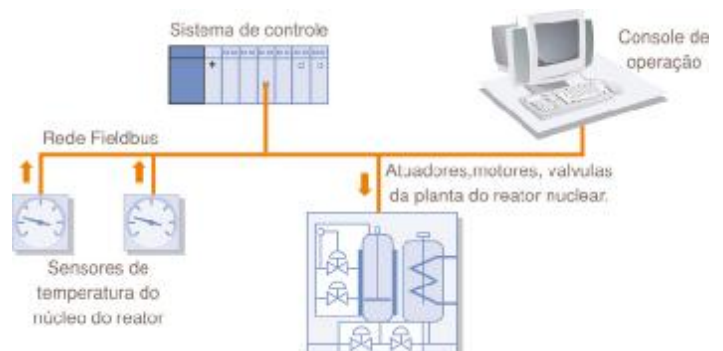
para cada um deles, as seguintes ações devem ocorrer:

- Se a temperatura exceder o “Alarme Alto 1” por 2 segundos, uma mensagem de alarme deve ser enviada pelo Sistema de Controle ao console do operador para notificá-lo da situação;
- Se a temperatura exceder o “Alarme Alto 2” por 700 ms, o Sistema de Controle deve iniciar uma seqüência de segurança para controlar a situação;
- Se a temperatura exceder o “Alarme Alto 3” por 200 ms, o Sistema de Controle deve iniciar uma seqüência de desligamento do Núcleo do Reator.



F.1 - Níveis de alarmes da temperatura do núcleo do reator.

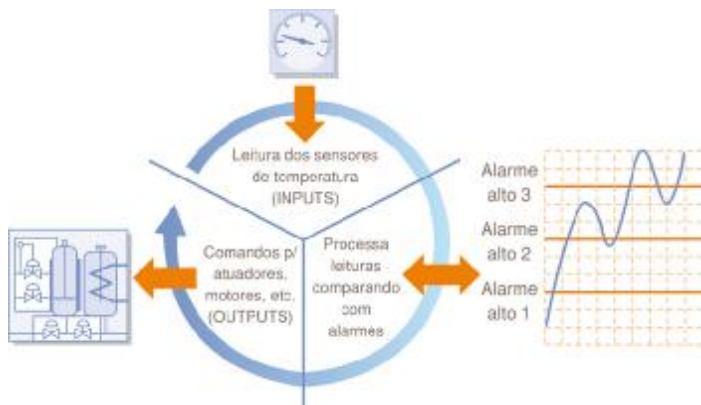
Na **figura 2**, temos uma representação do sistema de controle para este exemplo. Os equipamentos, sensores e demais dispositivos conectados ao sistema de controle, diretamente ou através de uma rede Fieldbus (lembre-se que, nesta série de artigos, Fieldbus é o termo genérico que utilizamos para rede industrial e não o nome de um produto), são chamados de I/O (input/output). Aqueles que transmitem sinais ao sistema de controle são inputs como, por exemplo, os sensores de temperatura do núcleo do reator. Aqueles que recebem comandos do sistema de controle para executarem ações na planta, tais como, atuadores, motores, válvulas, robôs, etc., são outputs.



F.2: Representação de Sistema de Controle da Usina Nuclear.

Em intervalos periódicos de tempo, chamados de scan time, o sistema de controle deve realizar uma varredura e

ler os sinais dos inputs, no caso, os sensores de temperatura instalados no núcleo do reator. Em seguida, o sistema de controle deve processar as leituras, comparando-as com os valores de “Alarme Alto 1”, “Alarme Alto 2” e “Alarme Alto 3”. Caso uma das leituras ultrapasse qualquer um destes três alarmes, nos tempos descritos, devem-se iniciar as ações previstas, enviando comandos para os outputs, ou seja, os equipamentos instalados na planta, tais como, atuadores, motores, válvulas, etc. Na **figura 3**, temos uma representação do ciclo do scan time do sistema de controle.



F.3 - Representação do scan time do sistema de controle.

Se, por hipótese, o scan time do sistema de controle fosse maior que 200 ms, o “Alarme Alto 3” não seria detectado em tempo e provocaria o superaquecimento do núcleo do reator. Dessa forma, requisitos de alta performance, especificados em termos de tempo de resposta para eventos, devem ser suportados pelo sistema de controle para garantir a segurança da planta nuclear. Como todo o sistema está envolvido, a velocidade da CPU, software e rede Fieldbus são todos fatores que afetam o tempo de resposta.

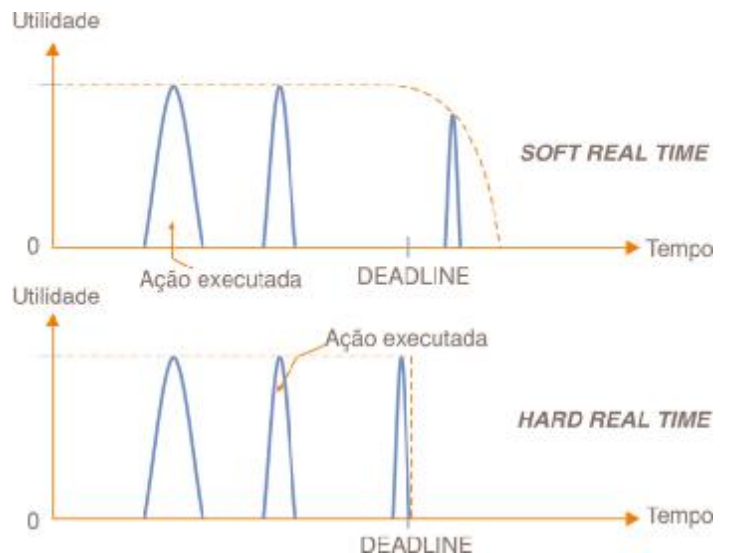
A rede Fieldbus deve transmitir pequenos e freqüentes pacotes de dados entre um grande número de nós e atendendo às restrições de tempo. É mais importante que a maior parte dos dados sejam transmitidos e entregues, respeitando a restrições de tempo, do que todos os dados sejam entregues da melhor maneira possível, porém, sem garantia de um tempo limite. Em contrapartida, a rede de uso geral encontrada, normalmente, no ambiente de escritórios, deve transmitir pacotes de dados grandes,

mas, não freqüentes, utilizando altas taxas de transmissão e sem restrições de tempo. Portanto, a principal característica que diferencia uma rede Fieldbus de uma rede de uso geral do ambiente de escritórios é o suporte a aplicações em tempo real.

## O Conceito de Tempo Real

Através dos parágrafos anteriores, afirmou-se que o principal requisito de uma rede Fieldbus, que faz parte de um sistema de controle, é a resposta em tempo real. Apesar do conceito de tempo real ser, freqüentemente, mencionado em vários trabalhos, é um conceito pouco compreendido.

A principal característica da comunicação em tempo real é o tempo em que a mensagem é entregue. Para cada mensagem é associado um deadline, ou seja, um atraso máximo que limita o tempo de entrega das mensagens através da rede. Se uma a mensagem é entregue no destino após o deadline, seu valor para a aplicação pode ser enormemente reduzido ou, até em algumas circunstâncias, considerado sem utilidade, sendo descartada pela aplicação [1]. As aplicações em tempo real se classificam em soft real time e hard real time.



F.4 - Função tempo/utilidade para hard e soft real time.

Desenvolva um programa em linguagem C para o sistema de controle de temperatura indicado no artigo. O programa desenvolvido por você deve ser determinístico para eventos.

**Observações para o programa:** O tipo de dado `time_t` é definido pela biblioteca ISO da linguagem C para armazenar valores de tempo. Estes valores são obtidos através da função `time()` que é definida no arquivo de cabeçalho de nome `<time.h>`. Os sistemas de padrão Unix e POSIX implementam o `time_t` como um inteiro com sinal (tipicamente com 32 ou 64 bits).

# Departamento de Engenharia e Arquitetura

Disciplina: Sistemas de Tempo Real

Prof. Rafael G. B. de Araújo

Aluno:



```
#include <stdio.h>
#include <time.h>

int main(void){
time_t    now;
// Obtem o tempo corrente
now = time(NULL);
return 0;
}
```

A seqüência de segurança é ativada chamando-se a função void ReatorSafe (void); e a seqüência de desligamento chamando-se a função void ReatorOff (void); A leitura do sensor de temperatura é feita chamando-se a função float TempRead (void); Os níveis de Alarme Alto 1, Alarme Alto 2 e Alarme Alto 3 serão definidos pelo programador. Ative essas funções através de comentários e/ou impressões na tela.

**Exercício 02:** Utilizando o kit didático da porta paralela simule a execução de um sumô de robô. Os motores devem ser invertidos quando o sensor for acionado. Suponha que o robô utiliza dois motores de passo. De acordo com a regra do campeonato de sumô de robôs cada round possui 90 segundos. Desta forma os robôs devem parar após cada round. Utilize threads para desenvolver a aplicação. Adicionar sensores de luz para limites da arena (+2 sensores final de curso). Um botão deve servir para ligar o robô e acionar o programa que controlará o tempo de 90 segundos.

**Exercício 03:** Desenvolva uma calculadora que receba os parâmetros (operação e valores) a ser efetuada no momento de sua ativação. Nenhum questionamento deve ser feito ao usuário do programa. A funcionalidade e operações a serem realizadas pela calculadora ficam a critério do programador (detalhar em cabeçalho comentado no programa).

**Exercício 04:** Spawn1 programa 1 ativa programa 2. Simular um motor de passo sendo acionado durante 10 ciclos. Variar com WAIT e NOWAIT.

**Exercício 05:** Spawn1 programa 1 ativa programa 2, espera e ativa programa 3. Motor 1 acionado, após 5 ciclos acionar motor 2 que após 5 ciclos esperará o acionamento de sensor (representado por qualquer botão do kit) para encerramento. Utilizar WAIT.

## The Spawnl( ) function

DOS is a single tasking operating system, thus only one program runs at a time. The Spawnl( ) function provides us with the capability of starting the execution of one program from within another program. The first program is called the parent process and the second program that gets called from within the first program is called a child process. Once the second program starts execution, the first is put on hold until the second program completes execution. The first program is then restarted. The following program demonstrates use of spawnl( ) function.

```
/* Mult.c */
int main ( int argc, char* argv[ ] ){
    int a[3], i, ret ;
    if ( argc < 3 || argc > 3 ){
        printf ( "Too many or Too few arguments..." ) ;
        exit ( 0 ) ;
    }

    for ( i = 1 ; i < argc ; i++ )
        a[i] = atoi ( argv[i] ) ;
    ret = a[1] * a[2] ;
    return ret ;
}
```

# Departamento de Engenharia e Arquitetura

Disciplina: Sistemas de Tempo Real

Prof. Rafael G. B. de Araújo

Aluno:



```
/* Spawn.c */
#include <process.h>
#include <stdio.h>
main( ){
    int val ;
    val = spawnl ( P_WAIT, "C:\\\\Mult.exe", "3", "10", "20", NULL);
    printf ( "\\nReturned value is: %d", val ) ;
}
```

Here, there are two programs. The program 'Mult.exe' works as a child process whereas 'Spawn.exe' works as a parent process. On execution of 'Spawn.exe' it invokes 'Mult.exe' and passes the command-line arguments to it. 'Mult.exe' in turn on execution, calculates the product of 10 and 20 and returns the value to val in 'Spawn.exe'. In our call to spawnl( ) function, we have passed 6 parameters, P\_WAIT as the mode of execution, path of '.exe' file to run as child process, total number of arguments to be passed to the child process, list of command line arguments and NULL. P\_WAIT will cause our application to freeze execution until the child process has completed its execution. This parameter needs to be passed as the default parameter if you are working under DOS. Under other operating systems that support multitasking, this parameter can be P\_NOWAIT or P\_OVERLAY. P\_NOWAIT will cause the parent process to execute along with the child process, P\_OVERLAY will load the child process on top of the parent process in the memory.