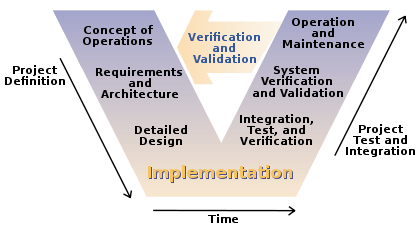
Fluxo de projeto embarcado

# Cross-compiler

É um compilador que roda em uma plataforma e gera um executável para outra plataforma, como por exemplo, se usássemos o gcc para compilar um executável para uma cpu de um microcontrolador.

# Embarcados

1.



O modelo em V é uma metodologia de desenvolvimento. Diferente da metodologia ágil, esse modelo é linear, sem apresentar iterações cíclicas. O gráfico acima representa a ordem dos passos dele.

Os passos começam com o design dos vários aspectos do projeto antes de chegarem a implementação de fato do código. Com o projeto implementado, começam as iterações de testes do projeto. Pelo fato de seguir uma forma linear, é muito importante gastar mais tempo na parte de design, pois só se faz ela uma vez.

Fonte: https://airbrake.io/blog/sdlc/v-model

2. Um RTOS é sigla para real-time operating system. Esse OS é caracterizado por ser utilizado em aplicações em tempo real, sem ter um buffer entre entrada e a saída. Por isso, é um aspecto muito importante de um RTOS a consistência de tempo de resposta, também conhecida como jitter.

O que difere um RTOS de algum outro OS é como age o componente scheduler. Ele é responsável por decidir quando um programa e roda e outro não. Em um RTOS, o padrão que o scheduler segue é chamado de determinístico, por seguir um nível de prioridade para cada tarefa definido pelo usuário.

Um exemplo de aplicação real-time seria um sistema embarcado para semáforo, em que o sistema iria decidir automaticamente o tempo que o semáforo ficaria verde baseado na densidade do tráfego.

# C

Em geral, C++ é uma linguagem mais fácil de se usar, por apresentar vários recursos que ajudam na vida do programador. Um deles é o bloco try and catch que não está presente em C. Em C, o programador deve inicializar todas as structs, enquanto em C++ o compilador que faz a inicialização. C++ também permite o uso de orientação a objetos, que C não permite.

# Paralelismo vs. Concorrência

O conceito de paralelismo e concorrência em um processador se torna claro quando pensamos em como de fato é feito o processamento de informações. Em uma CPU de apenas um core, a cada ciclo de clock é possível ser realizada uma instrução. Assim, se tivermos esta CPU rodando algum programa e ela receber uma entrada de algum periférico, o programa deixará de ser executado para que a CPU processe a informação do periférico, e depois será resumido. Isso é chamado de concorrência, e diz respeito à disputa pelos recursos do processador pelas diversas tarefas do OS.

Já em processadores com dois ou mais cores, é possível realizar mais instruções simultaneamente. Como no exemplo anterior, se esta CPU recebesse uma informação de um periférico ao mesmo que tempo em que ela rodasse um programa, ela seria capaz de realizar ambas tarefas ao mesmo tempo, sem ter que interromper nada. Isso é chamado de paralelismo, e diz respeito ao tratamento de informações em paralelo, sem interferência no resto do processador.