# Aula 5: Projeto e Implementação do Sistema

## Objetivos de Projeto

O primeiro problema que enfrentamos ao projetar um sistema é a definição dos seu **objetivos** e **especificações**. No nível mais alto, o projeto de sistema será afetado pela escolha do hardware e do tipo do sistema: *batch*, tempo compartilhado, monousuário, multiusuário, distribuído, tempo real, ou de uso geral.

Além do nível de projeto mais alto, os requisitos podem ser muito mais difíceis de especificar. Eles podem ser divididos em dois grupos básicos: objetivos de *usuário* e objetivos do *sistema*. Os **usuários** desejam certas prioridades óbvias em um sistema: ele deve ser eficaz e fácil de usar, confiável, seguro e rápido. Já para o **projetista**, as prioridades em um sistema são: ele deve ser fácil de projetar, implementar e manter, deve ser flexível, confiável, livre de erros e eficiente. Portanto, não existe uma **solução** única para o problema de definir os requisitos para um sistema operacional.

## Mecanismos e Políticas

A especificação e o projeto de um sistema operacional é uma tarefa altamente criativa. Embora nenhum livro possa dizer a você como fazê-la, existem princípios gerais de **engenharia de software** especialmente aplicáveis aos sistemas operacionais.

Um importante princípio é a separação entre **política** e **mecanismo**. Os mecanismos determinam *como* fazer algo; as políticas determinam o *que* será feito. Por exemplo, o *timer* é um mecanismo para assegurar a proteção da CPU, mas a decisão de por quanto tempo o *timer* deve ficar ativo, para um determinado usuário, é uma decisão de política.

A separação entre política e mecanismo é importante para a **flexibilidade**. As políticas se modificam no espaço ou no tempo. Um mudança na política iria, então, exigir a redefinição de apenas alguns parâmetros do sistema.

## Implementação

Uma vez que o sistema operacional tenha sido projetado, ele deve ser **implementado**. Atualmente os sistemas operacionais têm sido escritos com frequência através de linguagens de mais alto nível, como C ou C++.

As vantagens de se usar uma linguagem de mais alto nível para implementar um sistema operacional são as mesmas levadas em conta para programas aplicativos: o código pode ser escrito mais rápido, é mais compacto e mais fácil de compreender e depurar. Como desvantagens, está na redução de velocidade e o aumento dos requisitos de armazenamento.

# Geração do Sistema

Podemos projetar, codificar e implementar um sistema operacional para uma máquina em um determinado local. É mais comum, entretanto, que os sistemas operacionais sejam projetados para operar em qualquer uma de uma classe de máquinas, em uma variedade de locais, com uma variedade de configurações periféricas.

O sistema deve ser configurado e gerado para cada instalação específica, um processa algumas vezes denominado **geração do sistema (SYSGEN)**. Este programa lê a partir de um determinado arquivo ou solicita ao operador informação relacionada a configuração específica do hardware, ou aciona o hardware, diretamente, para determinar que componentes estão instalados.

Após gerado, um sistema operacional deve se tornar disponível para uso pelo hardware. Mas como o hardware sabe onde o kernel está ou como carregar este kernel? O procedimento de inicializar um computador carregando um kernel é conhecido para inicialização (**booting**). A maioria dos sistemas de computadores possuem um pequeno trecho de código, armazenado em memória ROM, conhecido como **programa bootstrap**. Este código é capaz de endereçar o kernel, carregá-lo na memória principal e iniciar sua execução.

# Exercícios

1. Quais os problemas enfrentados em um projeto de sistema?

2. Por que a separação entre mecanismo e política é um princípio desejável?

3. Na sua opinião, é melhor ou não utilizar uma linguagem de alto nível na implementação de um sistema?

4. Qual a importância do SYSGEN?

5. Qual a diferença entre memória ROM e CMOS? Qual a utilidade dessas memórias?