Cap 02
Estruturas dos Sistemas Operacionais

1. O que você entende por ***User Interface*** (UI)? Quais são os diferentes tipos de interação usuário-máquina geralmente fornecidas pelos diversos Sistemas Operacionais? Explique:
   R.: A User Interface (UI) refere-se ao meio pelo qual um usuário interage com um dispositivo ou software.

   Quase todos SOs possuem uma UI (User Interface). Essa interface pode assumir várias formas:
   - Command-line Interface (CLI): Interface de linha de comando
   - Batch Interface: Comandos e diretivas inseridas em arquivos que podem ser executados automaticamente pelo SO. Ex: arquivos **.bat** e **.sh;**
   - Graphical User Interface (GUI): sistemas de janelas

2. Sobre os vídeos da evolução das interfaces gráficas do **Microsoft Windows** e **Apple MacOS**, considerando a versão mais recentes de ambos, quais interfaces são melhores na sua opinião e por que?
   R.: Cada um tem sua vantagem, eu prefiro o Windows por já estar mais acostumado e da pra jogar, diferente do macOS que não tem suporte pra maioria dos jogos.
3. O que são ***System Calls***? Como são implementadas e como os programas podem acessá-las de forma mais simplificada?
   R.: System calls (chamadas de sistema) são mecanismos utilizados pelos programas para solicitar serviços ou recursos do sistema operacional. As system calls são implementadas internamente pelo sistema operacional e são acessíveis aos programas por meio de bibliotecas ou wrappers fornecidos pelo sistema operacional. Para acessá-las de forma mais simplificada, os programas utilizam essas bibliotecas que encapsulam os detalhes complexos das chamadas de sistema.
4. Para cada categoria de ***System Calls***, cite um exemplo para os SO's Linux e Windows;
   R.: Para o gerenciamento de arquivos, exemplos no Linux incluem open(), no Windows é CreateFile(). Para processos, no Linux temos fork() e no Windows, CreateProcess(). Para memória, no Linux é mmap(), no Windows é VirtualAlloc(). Para rede, em ambos os sistemas é socket(). E para gerenciamento de tempo, no Linux é gettimeofday(), enquanto no Windows é GetSystemTime().
5. Antes de iniciar a construção ou adaptação de um novo SO, geralmente quais objetivos devem ser verificados? Explique;
   R.: Antes de iniciar a construção ou adaptação de um novo sistema operacional (SO), é crucial estabelecer uma série de objetivos e considerações para garantir que o sistema atenda às necessidades e expectativas dos usuários.
6. Ao instalar um novo SO, o que deve ser feito para adaptá-lo melhor às vontades e hardware do usuário? Explique;
   R.: Ao instalar um novo sistema operacional (SO), a adaptação ideal às preferências e ao hardware do usuário envolve selecionar opções de instalação personalizadas, configurar idioma e região, particionar o disco, instalar e atualizar drivers de hardware, manter software atualizado, instalar aplicativos necessários, personalizar a interface, ajustar configurações de segurança e privacidade,

otimizar configurações de energia, criar contas de usuário, configurar backups e executar testes para garantir a funcionalidade e satisfação do usuário.

7. O que você entende por ***Debugging***? Que medidas geralmente os SO's adotam a fim de verificar/corrigir possíveis erros no Sistema?

R.: Debugging, em termos de programação e desenvolvimento de software, refere-se ao processo de identificação, análise e correção de erros (bugs) em um programa ou sistema. Esses erros podem causar mau funcionamento, comportamento inesperado ou falhas no sistema. Debugging é uma parte crítica do ciclo de desenvolvimento de software para garantir que o software funcione conforme o esperado e seja livre de erros. Os sistemas operacionais (SOs) também precisam passar por processos de debugging para garantir sua estabilidade, segurança e eficiência.

8. No contexto de SO's, o que são Programas de Sistema? Explique e cite dois exemplos.

R.: Programas de sistema são softwares essenciais para o funcionamento e gerenciamento de sistemas operacionais (SOs) e outros softwares. Eles não são interativos para usuários finais, mas sim suportam o sistema operacional e suas funções. Dois exemplos são compiladores, que traduzem código-fonte em código de máquina, e carregadores, que carregam programas executáveis na memória. Esses programas otimizam a eficiência e a estabilidade do sistema ao realizar tarefas de baixo nível e fornecer serviços cruciais.

9. Quais são os diferentes Modelos Arquiteturais geralmente utilizados para projetar SO's? Faça um breve comparativo entre eles;

R.: Existem três principais modelos arquiteturais de sistemas operacionais: monolítico, micronúcleo e máquina virtual. O modelo monolítico concentra todos os componentes em um único programa, oferecendo simplicidade e eficiência, mas pode ser difícil de manter. O modelo de micronúcleo divide o sistema em partes mínimas, melhorando a modularidade e permitindo atualizações mais fáceis. O modelo de máquina virtual cria uma camada intermediária para executar vários sistemas operacionais em um único hardware. A escolha entre eles depende das necessidades específicas do sistema, com o monolítico sendo simples e eficiente, o micronúcleo sendo modular e o de máquina virtual possibilitando a virtualização de múltiplos sistemas.

10. O que são ***Virtual Machines***? Quais são os diferentes modos de criar ambientes virtualizados?

R.: Virtual Machines (VMs) são ambientes isolados que emulam hardware e permitem executar sistemas operacionais completos dentro de um sistema hospedeiro.

- Virtualização de Hardware (Hypervisor Type 1)
- Virtualização de Hardware (Hypervisor Type 2)
- Containerização
- Emulação