

**Cap.2 – Estruturas do Sistema Operacional**

1. O que você entende por User Interface (UI)? Quais são os diferentes tipos de interação usuário-máquina geralmente fornecidas pelos diversos Sistemas Operacionais (SO’s)? Explique;

**R-** O User Interface é um termo comumente utilizado para explicar como são feitas as interações entre pessoas e softwares ou aplicativos. O UI tem como objetivo facilitar a navegação dos usuários, permitindo que eles consigam ser orientados através de recursos visuais para concluírem suas ações com praticidade.

**Command-line Interface (CLI)** - é um meio de interagir com um programa de computador, onde o utilizador emite comandos para o programa sob a forma de sucessivas linhas de texto (linhas de comando). Cada sistema operacional traz um intérprete padrão (o shell) para aqueles comandos os quais executam tarefas distintas e resolvem diferentes tipos de problemas.

**Graphical User Interface (GUI)** - é um tipo de interface do usuário que permite a interação com dispositivos digitais por meio de elementos gráficos como ícones e outros indicadores visuais. A interação é, geralmente, com um teclado, mouse ou no próprio monitor (touchscreen), com os quais o usuário é capaz de selecionar símbolos e manipulá-los de forma a obter algum resultado prático.

1. O que são System Calls? Como são implementadas e como os programas podem acessá-las de forma mais simplificada?

**R-** System Calls são mecanismos programáticos que fornecem uma interface para acessar os serviços oferecidos pelo SO. Isto pode incluir serviços relacionados ao hardware (por exemplo, acessar uma unidade de disco rígido), criação e execução de novos processos e comunicação com serviços do núcleo de maneira integral como escalonamento do processador. Geralmente são rotinas escritas em Assembly ou linguagem de alto nível como C ou C++. Um programa aplicativo pode chamar diretamente as System Calls ou chamá-las por intermédio de uma API.

1. Para cada categoria de System Calls, cite um exemplo para os SO’s Linux e Windows;

**R- Controle de Processos**

Windows: CreateProcess( ) Linux: fork( )

**Manipulação de Arquivos**

Windows: CreateFile( ) Linux: open( )

**Manipulação de Dispositivos**

Windows: ReadConsole( ) Linux: read( )

**Manutenção de Informações**

Windows: Sleep( ) Linux: sleep( )

**Comunicação**

Windows: CreatePipe( ) Linux: pipe( )

**Proteção**

Windows: SetFileSecurity( ) Linux: chmod( )

1. 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 SO, alguns objetivos devem ser definidos e verificados, como, por exemplo: Se o SO será monousuário, multiusuário, multitarefa, tempo-real, se será para PC ou dispositivos móveis, etc.

É importante também verificar políticas (o que fazer) e Mecanismos (como fazer) e definir a linguagem de Programação.

1. 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 SO, um programa especial SYSGEN verifica o hardware ou pergunta ao usuário informações para definir os parâmetros e os componentes da máquina. O SYSGEN copia do DVD ou ISO de instalação somente os módulos necessários do SO, de acordo com o hardware, e os instala no disco.

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

**R-** Debugging é a atividade de descobrir e corrigir erros (bugs) em um Sistema. A maioria dos Sistemas Operacionais gravam as informações de erros (tanto de processos quanto de Kernel) em arquivos de Logs e guardam Capturas de Memória (Memory Dumps) para alertar aos operadores ou usuários que um problema ocorreu.

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

**R-** Os Programas de Sistema, também conhecidos como utilitários de sistema, fornecem um ambiente amigável para o desenvolvimento e a execução de programas.

Utilitários – Ex: Browsers, planilhas, jogos, etc.

Suporte a linguagens de Programação – Ex: Compiladores, montadores, depuradores...

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

**R- Arquitetura Simples:** As interfaces e níveis de funcionalidades oferecidas pelo SO não estão bem separadas (tudo colocado em um grande programa executável). Trabalha como se fosse um só módulo.

**Arquitetura em Camadas:** Possui várias camadas com funções especificas. Cada camada usa somente as funções e serviços disponibilizados pelas camadas inferiores.

**Arquitetura de Microkernel:** O Kernel do sistema é enxuto, possuindo somente os componentes essenciais (Primitivas do núcleo). Os componentes essenciais normalmente são os códigos de baixo nível necessários para interagir com o hardware e abstrações fundamentais. Todos os outros aspectos de alto nível como políticas de gerenciamento de recursos são implementados fora do núcleo.

**Arquitetura em Módulos:** É o melhor tipo de arquitetura. O Kernel é enxuto, possuindo somente os componentes essenciais. Os demais módulos que compõem o SO são carregados conforme a necessidade, durante o boot ou dinamicamente durante a execução do SO. Cada módulo pode chamar serviços oferecidos por outros módulos independentemente, pois todos os módulos estão no mesmo nível hierárquico. Cada módulo pode se comunicar diretamente com os demais, não mais através de passagem de mensagens no Kernel.

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

**R-** Virtual Machines é um modelo de virtualização que cria vários ambientes convidados sobre um mesmo hardware, ou seja, cria várias Máquinas Virtuais convidadas com seus respectivos Sistemas Operacionais, possibilitando que o usuário acesse uma simulação de computador dentro de outro. Nesse sentido, a partir de um gerenciador que funciona como uma interface, o hardware principal é dividido em vários pools de recursos, o que permite uso deles para objetivos diferentes e específicos.

Os diferentes modos de criar ambientes virtualizados são: Virtualização total, Emulação e Paravirtualização.