Nome:	WESLEY MARCOS BORGES					Curso:	GEC
Matrícula:	1651	Período:	P8	Matéria:	C012	Turma:	

Cap. 2 – Estruturas do Sistema Operacional

1) Ao utilizar um sistema operacional, o usuário necessita ter uma comunicação com a máquina. Essas comunicações podem ser feitas, principalmente, com duas ferramentas: com a Interface de Linha de Comando (CLI) e com as Interface Gráficas de Usuário (GUI). Elas têm a mesma ideia, porém com abordagens diferentes. Enquanto a CLI depende que o usuário tenha um conhecimento mais apurado, por usar linhas de comando, a GUI tem uma liberdade quanto a isso, pois possui elementos gráficos para ajudar os usuários, dispensando um conhecimento profundo, além de periféricos que ajudam nessa produtividade, como o mouse. Uma curiosidade é que os PCs se popularizaram quando a Apple introduziu em um de seus primeiros modelos uma GUI.

2) As ***system calls***, como o próprio nome sugere, são chamadas do sistema, responsáveis por todas as operações de execução como alocação de recursos, arquivos de sistema, comunicação, proteção, operações de I/O, entre outras operações realizadas dentro de um SO. As ***system calls*** são implementadas por classes, que são conectadas por APIs, que facilitam o processo.

3) A tabela à seguir mostra as categorias de systrem calls no Windows e no Linux:

System Calls	Linux	Windows
Controle de Processos	fork()	CreateProcess()
Comunicação	pipe()	CreatePipe()
Proteção	chmod()	SetFileSecurity()
Manipulação de Dispositivos	loctl()	SetConsoleMode()
Manipulação de Arquivos	open()	CreateFile()
Manipulação de Informações	getpid()	GetCurrentProcessID()
Alarme/Tempo de processo	alarm()	SetTimer()

4) Ao construir/adaptar um novo SO, demos nos preocupar com algumas características. Primeiro definimos qual será o tipo do SO (mono usuário, multiusuário, tempo-real, multitarefa), para consolidar os objetivos do SO. Segundo, devemos verificar as políticas do SO (o que fazer). Terceiro, devemos verificar suas metodologias (como fazer). Por fim, definiremos qual a linguagem o qual o sistema será desenvolvido.

5) Ao instalarmos um novo SO, será rodado o System Generation (SysGen), que tem como principal função ajudar o sistema a entender quais características o usuário quer no seu SO, como a quantidade de partições de memória (HD/SSD), quais os dispositivos de I/O disponíveis, entre outros.

6) Na perspectiva de um SO, o processo de ***debugging*** é uma sequência de ações que o SO realiza à fim de encontrar o erro. Primeiro é feito uma pesquisa, onde, se for encontrada uma falha, o sistema a analisará e tomara uma medida cabível para resolver esse problema. Para evitar que o sistema não saiba qual o último processo feito por determinado software quando um erro ocorre, como a tela azul, é gerado logs do sistema, que ficam armazenadas na memória secundária. Esses logs armazenam as ***memory dumps***, que são "fotografias" da memória RAM no momento do erro. Dessa forma, o SO conseguirá continuar de onde ele parou no momento do erro.

7) Os programas de sistema são programas que ficam na camada acima do kernel, que tem como função facilitar e possibilitar determinadas atividades por parte do usuário, como desenvolver softwares, gerenciar os arquivos, abrir multimídias, etc. Alguns exemplos de programas de sistema são: Windows Explorer, bloco de notas, GUI, WMP, entre outros.

8) Existem quatro tipos de arquiteturas de SOs, sendo elas:

Arquitetura simples = trabalha de forma estruturada, onde os processos são centralizados, sendo simples e ágil. Mas como é centralizado, é mais suscetível a softwares maliciosos, além de que quando uma aplicação trava, o sistema todo para. Manutenção difícil. Exemplos: MS-DOS e Unix.

Arquitetura em camadas = trabalha em forma de camadas hierárquicas, onde são divididas as tarefas, sendo as camadas dependentes entre si. Há modularidade, facilitando a depuração e modificação. Mas é mais lenta, sendo que cada camada adiciona um overhead à camada de sistema. Exemplos: THE OS e MULTICS.

Arquitetura microkernel = trabalha com o kernel enxuto, ou seja, apenas com as variáveis primitivas. Dessa forma, o kernel tem que autorizar cada system call, o que deixa o kernel mais seguro e confiável, porém lento. Exemplos: MacOS X e Windows NT (Tradicional).

Arquitetura em módulos = trabalha com kernel enxuto. Os módulos são chamados assim que há necessidade, sendo eles operados de forma independentes, sem hierarquia (diferente da arquitetura em camadas). A comunicação entre módulos é feita de forma direta, sem dependência do kernel. Sua desvantagem está na complexidade de desenvolvimento. Exemplos: Linux e MacOS.

9) As Virtual Machines (VM) são ambientes de software (SOs) que rodam em cima do sistema operacional da máquina, ficando acima de uma camada de software, o SO instalado, que faz a comunicação com o hardware. A VM é chamada de guest (convidado). Esse processo de virtualização pode ocorrer de três formas:

Virtualização total = todo o hardware fala com o SO instalado e a VM roda em cima de uma camada de software (SO instalado).

Paravirtualização = máquinas que conseguem rodar mais de um SO ao mesmo tempo, iniciando todos juntos e independentes um do outro. Todos se comunicam com o hardware, que precisa ser muito potente para aguentar esse processamento.

Emulação = processo de simular um hardware específico em um SO. Um clássico exemplo são os consoles antigos, que possuem emuladores que permitem o usuário jogar os jogos antigos como se estivessem rodando no console original. Um exemplo atual é o Android Studio, que emula um celular android para rodar as aplicações.