Jossana Kelly Tavares Marques - tavares.marquesss@gmail.com

Resumo - Módulo 2

● Sistemas Operacionais

A **parte física** do computador é o hardware, seus componentes e os demais periféricos, enquanto a **parte lógica** é o software operando juntamente com os sistemas operacionais.

O Sistema Operacional em si é um **intermediador** entre o software e o hardware. Ele serve como uma parte de **interface** que efetua o controle de entrada e saída de dados e rotinas do hardware e permite que nós (os usuários) **controlemos o hardware.** E o **driver** é o que conduz a ligação entre os equipamentos que compõem o hardware e o sistema operacional.

Existem dois tipos de interação possíveis para com ele, sendo a **interação textual** (que é aquela que acontece através de comandos no terminal) e a **interação gráfica** (que é aquela que acontece por meio de ícones visuais na tela).

● Máquina Multinível

O funcionamento de um computador acontece em camadas que reúnem o funcionamento e a interação do software e hardware em níveis, e a isso damos o nome de **máquina multinível**.

● Histórico de computadores

- **primeira geração:** O primeiro computador, o ENIAC, era operado por válvulas, sem sistema operacional, ou seja, ele **operava diretamente com o hardware**;
- **segunda geração:** Já tinha uma linguagem de programação de baixo nível presente (Assembly), onde trabalhavam com o conceito de filas de execução (lote) e **transistores**;
- **terceira geração:** Surgimento do sistema **OS/360** da IBM, e eram utilizados **circuitos integrados**.
- **quarta geração:** Já havia Sistema operacional, a máquina era operada por uma pessoa e eram utilizados **circuitos integrados em grande escala**;
- **quinta geração:** É a geração atual,se tem a difusão da Internet em níveis absurdos, IOT, computação oblíqua e grandes empresas de tecnologia ativas e dominantes no mercado (sigla GAFAM).

● Tipos de Sistemas operacionais

- **monotarefa:** permite a execução de 1 programa por vez (na memória RAM) e as tarefas eram organizadas em fila e realizadas alternadamente;

- **multitarefa ou multiprogramado:** permite a execução de mais de um programa por vez na memória, e a execução acontece de maneira simultânea.
- **sistema em Batch:** o próprio sistema cria as filas de execução, sem haver uma interação com o usuário durante a espera.

● Sistemas operacionais mobile

- **Android**: está na maioria das coisas, é de **código aberto e flexível**, pois cada fabricante pode realizar alterações. É **formado por 5 camadas básicas**, sendo elas: Linux Kernel, HAL (hardware abstract layer), bibliotecas, Java API e System apps.
- **IOS:** foi **pioneiro no touchscreen** (2007), **não possui código aberto**, possui **apps próprios**, não é flexível e **só funciona em hardware da Apple**.

● Filas

- São estruturas de organização utilizadas pelos sistemas operacionais visando otimização, desempenho e velocidade.

● Comandos no terminal

Toda unidade de armazenamento é representada por uma letra, e o Windows geralmente fica no **C** (se padronizou).
→ **C:\users**

Comandos aprendidos:

- **dir**: ver os diretórios da pasta do caminho;
- **cd**: entrar e sair das pastas;
- **rmdir** ou **rd**: apagar diretório (apenas vazios);
- **dir/?**: lista opções de comando;
- **del:** deletar;
- **copy:** copiar arquivo/diretório;
- **move:** mover arquivo/diretório.

*alguns dos comandos podem ser utilizados com **.** e ***** no final.

● Git e GitHub:

- **Git:** versionador de arquivo (minimiza/elimina a multiplicação de arquivos);
- **GitHub:** repositório remoto e rede social de desenvolvedores.

Comandos aprendidos:

- **git init:** para inicializar um diretório como repositório;
- **git config –global user.name “ ”**: para configurar/definir o nome de usuário utilizador do Git que irá efetuar os commits;

- **git config –global user.email “ ”** para configurar/definir o e-mail de usuário utilizador do Git que irá efetuar os commits;
- **git config -l:** para visualizar as configurações de usuário
- **git commit -m “”:** para commitar;
- **git add *:** para adicionar os arquivos no stage (na área de espera);
- **git push:** empurra os arquivos para o repositório remoto.