BÁSICO DE UNIX E SUAS FERRAMENTAS

UNIX:

É/foi um:
→ Sistema Operacional
→ Conjunto de ferramentas
→ Espirito da época

Mesmo existindo a mais de 50 anos, ainda é um sistema/formato/ferramenta **extremamente** útil. Hoje vem em forma de **LINUX**, **BSD**, **MacOS**, etc.

O padrão UNIX foi feito em volta de criar várias ferramentas mínimas, com funções bem estabelecidas, que facilmente possam se comunicar. **O coração do sistema é o SHELL.**

A interface padrão para o sistema é o **terminal.** Ele é um conjunto de teclado-saída que pode enviar e receber texto.

TTY → TELETYPES → Terminais adaptados em máquinas de escrever

GLASS TTY → Dois monitores que recebiam texto e escreviam na tela.

Hoje, terminais são **emuladores** dentro dos SO para servidor X. Os mais comuns Gnome Terminal, Alacritty, Xtern, Terminator e ST.

SHELL:

O SHELL é uma interface para facilitar a comunicação com um computador.

Unix é dividido em **três partes:**
→ Os programas
→ O SHELL
→ O Kernel

KERNEL:

Toma conta de fazer as funções básicas do sistema, como se comunicar com o sistema de arquivos, se comunicar com Hardware e Software, etc

PROGRAMAS:

São ferramentas mínimas que realizam funções desejadas pelo usuário.

SHELL:

Interliga todas as partes. Ele consegue comunicar o **Kernel**, assim como os programas, interligar os programas, controlar fluxo, etc.

O **SHELL** roda em cima de um diretório, conhecido como **Working Directory.** Pode ser acessado por **$PWD** ou **pwd**.

SISTEMA DE ARQUIVOS:

O **Sistema de arquivos** UNIX é extremamente simples: Não existe flags para indicar se uma seção de disco é reservada para programas, ou se é para arquivos, ou código externos, etc.

No sistema de arquivos, **todos os arquivos são tratados igualmente,** seja um arquivo de texto, binário, etc.

O **Sistema de arquivos** inclui entrada para Hardware, processos, arquivos de sistema, memória, etc.

PROGRAMAS UNIX:

São ferramentas que são feiras para executar uma ação ou um grupo de ações similares, de uma maneira eficiente, e de uma maneira legível por outros programas.

A comunicação dos programas tem em comum três arquivos: **0, 1 e 2,** mais conhecido como **STDIN** (entrada padrão 0), **STOUT** (saída padrão 1) e **STDERR** (erro padrão 2).

Entrada do usuário	→	**STDIN**
Saída do programa	→	**STDOUT**
Erro	→	**STDERR**

Por padrão, o **SHELL** tem algumas ferramentas para alterar o que os programas fazem, o fluxo dos programas, como se comunicam com o sistema.

São elas:

- → Redireção
- → Expansão de argumentos
- → Regex
- → Substituição de comandos
- → Variáveis
- → Controle de fluxo, etc

>

Escreve a saída do programa para dentro do arquivo

programa > arquivo

»

Concatena a saída padrão do comando para dentro do arquivo. Não sobrescreve o arquivo, mas adiciona a saída do programa ao final dele.

programa » arquivo

<

Executa o programa, mas ao invés de utilizar o teclado como entrada, usa o conteúdo do arquivo.

programa < arquivo

| (PIPE)

Escreve a saída padrão do programa1 na entrada padrão do programa 2

programa1 | programa2

programa1 < arquivo1 | programa2 | programa3 » arquivo2

ARGUMENTOS:

Os argumentos passados para os programas podem ser, e frequentemente são mais complexos do que o usuário escreve.

Usuário escreve : programa arquivo
Programa recebe : programa/usr/ucb/arquivo

Na maioria dos casos, essa expansão de argumentos se dá a partir de caminhos, regex, substituição de comandos ou variáveis.

REGEX:

É uma maneira para se referir as **strings** de uma maneira geral.
Regex utilizado pelo POSIX:

ALGUNS COMANDOS BÁSICOS NO SHELL:

pwm → Mostra em qual diretório está
cd → Pasta Abre uma pasta
./ → Pasta atual
../ → Pasta anterior
ls → Lista um diretório
ll → Lista um diretório por completo
touch nome.extensão → Cria um arquivo
mkdir nomePasta → Cria uma pasta
rm -rf nomeArquivo → Exclui um arquivo
echo → Imprimir um arquivo
cat → Concatena um arquivo
mv → Move arquivos
grep 'letra/palavra' < programa.extensão → Procura uma palavra determinada em um arquivo

> Maior que → Escreve a saída do programa para dentro do arquivo
cat > texto.txt

>> Maior Maior que → Não sobrescreve o arquivo, mas adiciona a saída do programa ao final dele.
cat >> texto.txt

< Menor que → Executa o programa utilizando o conteúdo de um arquivo.
grep 'teste' < texto.txt

| PIPE → Inicia um programa ao termino de outro
programa1 | programa2

; → Lê o próximo comando na mesma linha
& → Não espera o último comando finalizar para iniciar o próximo
&& → Só executa o segundo comando caso o primeiro tenha funcionado
|| → Só executa o segundo comando caso o primeiro tenha falhado
if → "Se"
while → "Até"
for → "Enquanto"
case → "Caso X", senão X

PARA QUANDO ABRIR UM ARQUIVO NO VSCODE, ELE NÃO FECHAR O QUE ESTÁ ABERTO:

WORKBENCH.EDITOR.ENABLEPREVIEW para **FALSE**