

# Atividade 09

Valor: 3,0 pontos na Segunda Nota

Primeira Chamada: 18/11/2025

Segunda Chamada: 19/11/2025

## Questão 01: Gerenciamento de Arquivo de Hosts com Funções e Opções (1,5 pontos)

O objetivo desta questão é criar um *script* para gerenciar um arquivo simples de mapeamento de nomes de máquinas (*hostnames*) e endereços IP, demonstrando o uso obrigatório do `getopts` para processar múltiplas opções e argumentos.

Nome do Script: `hosts.sh`

### Estrutura de Dados:

O *script* deve usar um arquivo chamado `hosts.db` para persistir os dados. Cada entrada neste arquivo deve ser um par `hostname, IP` por linha.

### Requisito de Sintaxe (Obrigatório):

Você deve utilizar o comando `getopts` em um laço `while` para identificar qual operação o usuário deseja realizar. As opções que exigem um argumento devem ser seguidas por dois pontos na cadeia de opções do `getopts`.

### Funções de Manipulação (Obrigatório):

Você deve criar **funções** 4 separadas no *script* para cada operação a seguir. O `getopts` deve chamar a função apropriada.

Operação	Chamada Esperada	Ação da Função
Adicionar	<code>./hosts.sh -a &lt;hostname&gt; -i &lt;IP&gt;</code>	Adiciona o par <code>hostname, IP</code> ao arquivo <code>hosts.db</code> .
Remover	<code>./hosts.sh -d &lt;hostname&gt;</code>	Remove a linha que contém o <code>&lt;hostname&gt;</code> do arquivo <code>hosts.db</code> . Dica: Use o comando <code>sed</code> com a função de deletar ( <code>d</code> ).
Listar	<code>./hosts.sh -l</code>	Exibe todo o conteúdo do arquivo <code>hosts.db</code> .
Procurar	<code>./hosts.sh &lt;hostname&gt; (padrão)</code>	Procura pelo <code>hostname</code> fornecido e imprime o IP correspondente.
Procurar Reversa	<code>./hosts.sh -r &lt;IP&gt;</code>	Procura pelo <code>IP</code> fornecido e imprime o <code>hostname</code> correspondente. Dica: Use <code>grep</code> com expressão regular para procurar e <code>cut</code> para isolar o resultado.

### Exemplo de Execução:

```
# Adiciona entradas
$ ./hosts.sh -a routerlab -i 192.168.0.1
$ ./hosts.sh -a lab01 -i 192.168.0.100

# Lista todas as entradas (Saída pode ser formatada com printf)
$ ./hosts.sh -l
routerlab 192.168.0.1
lab01     192.168.0.100

# Remove uma entrada
$ ./hosts.sh -d routerlab

# Procura pelo nome (Saída: Apenas o IP)
$ ./hosts.sh lab01
192.168.0.100

# Procura reversa (Saída: Apenas o hostname)
$ ./hosts.sh -r 192.168.0.100
lab01
```

---

## Questão 02: Monitor de Desempenho Interativo (1,5 pontos)

O objetivo é criar um *script* interativo para monitorar o desempenho básico de um servidor Linux, utilizando funções e comandos de **leitura/escrita** ( `read` , `tput` ) para construir um menu funcional.

**Nome do Script:** `sistema.sh`

### Requisitos de Interação e Formato:

1. **Limpeza e Menu:** O *script* deve começar limpando a tela ( `tput clear` ) e exibindo o menu de opções.
2. **Função Menu (Obrigatório):** Crie uma função chamada `menu` para exibir o título do sistema e todas as opções disponíveis.
3. **Processamento da Escolha:** Use o comando `read` para capturar a escolha do usuário.
4. **Estrutura de Controle:** Use a estrutura de controle `case` para executar o comando correspondente à opção escolhida.
5. **Loop Principal:** O *script* deve ser executado em um **laço de repetição** ( `while` ou `until` ) que só termina quando a opção **Sair** for selecionada.

### Opções do Menu:

Ao selecionar uma opção, o *script* deve:

- Limpar a tela novamente ( `tput clear` ).
- Executar o comando correspondente.
- Exibir o resultado do comando.
- Usar o comando `read -n 1 -p "Pressione ENTER para voltar ao menu..."` para aguardar o usuário, e então retornar ao menu inicial.

Opção	Comando a ser Executado	Descrição
1	<code>uptime</code>	Tempo ligado do sistema.
2	<code>dmesg \n tail -n 10</code>	As 10 últimas mensagens do Kernel.
3	<code>vmstat 1 10</code>	Estatísticas de memória virtual (10 relatórios de 1 segundo).
4	<code>mpstat -P ALL 1 5</code>	Uso da CPU por núcleo (5 relatórios de 1 segundo).
5	<code>pidstat 1 5</code>	Uso da CPU por processo (5 relatórios de 1 segundo).
6	<code>free -m</code>	Uso da memória física em Megabytes.
7	Sair	Termina a execução do <i>script</i> ( <code>exit</code> ).

## Dicas de Implementação:

- Use um laço `while true` e o comando `case` para controlar o fluxo.
- Use `tput setaf X` e `tput sgr0` para colorir o menu ou o título.

## CrITÉrios de Correção:

O professor irá verificar a execução, navegando pelas opções (1 a 6) para garantir que o comando correto é executado, a tela é limpa e o retorno ao menu ocorre, e que a opção 7 finaliza o *script*. A presença da função `menu` e o uso de `case` são obrigatórios.