🐚 Manual Rápido - Shell Script

🎯 O que é Shell Script?

Imagine que você tem um **robô assistente** que executa comandos do terminal pra você. Shell Script é a "receita" que você escreve para esse robô! 🤖

Analogia: É como escrever uma lista de tarefas que o computador vai executar automaticamente.

🚀 Primeiro Script

📝 Hello World

```bash
#!/bin/bash
# Meu primeiro script

echo "Olá, mundo!"
echo "Hoje é: $(date)"
```

⚡ Como executar:

```bash
# 1. Criar arquivo
nano meu_script.sh

# 2. Dar permissão de execução
chmod +x meu_script.sh

# 3. Executar
./meu_script.sh
```

📦 Variáveis

🏷️ Criar e Usar

bash

```bash
#!/bin/bash

# Criar variáveis (SEM espaços ao redor do =)
nome="João"
idade=25
site="github.com"

# Usar variáveis (com $)
echo "Nome: $nome"
echo "Idade: $idade anos"
echo "Site: https://$site"

# Ou com chaves (mais seguro)
echo "Email: contato@${site}"
```

🎮 Variáveis Especiais

bash

```bash
$0   # Nome do script
$1   # Primeiro parâmetro
$2   # Segundo parâmetro
$#   # Número total de parâmetros
$@   # Todos os parâmetros
$$   # PID do script
$?   # Código de saída do último comando
```

💬 Entrada do Usuário

bash

```bash
#!/bin/bash

echo "Qual seu nome?"
read nome

echo "Qual sua idade?"
read idade

echo "Olá $nome, você tem $idade anos!"
```

🔍 Condicionais (if/else)

🎯 Estrutura Básica

bash

```bash
#!/bin/bash

idade=18

if [ $idade -ge 18 ]; then
   echo "Você é maior de idade"
elif [ $idade -ge 13 ]; then
   echo "Você é adolescente"
else
   echo "Você é criança"
fi
```

🔧 Operadores de Comparação

bash

```bash
# Números
-eq   # igual (equal)
-ne   # diferente (not equal)
-gt   # maior que (greater than)
-ge   # maior ou igual (greater or equal)
-lt   # menor que (less than)
-le   # menor ou igual (less or equal)

# Strings
=     # igual
!=    # diferente
-z    # string vazia
-n    # string não vazia

# Arquivos
-f    # é arquivo
-d    # é diretório
-e    # existe
-r    # é legível
-w    # é gravável
-x    # é executável
```

🧩 Exemplos Práticos

bash

```bash
#!/bin/bash

# Verificar se arquivo existe
if [ -f "dados.txt" ]; then
    echo "Arquivo existe!"
else
    echo "Arquivo não encontrado"
fi

# Verificar entrada do usuário
echo "Digite sua nota (0-10):"
read nota

if [ $nota -ge 7 ]; then
    echo "Aprovado! 🎉"
elif [ $nota -ge 5 ]; then
    echo "Recuperação 📚"
else
    echo "Reprovado 😞"
fi
```

🔄 Loops (Repetições)

🎪 For Loop

bash

```bash
#!/bin/bash

# Loop simples
for i in 1 2 3 4 5; do
    echo "Número: $i"
done

# Loop com range
for i in {1..10}; do
    echo "Contando: $i"
done

# Loop em arquivos
for arquivo in *.txt; do
    echo "Processando: $arquivo"
done

# Loop em lista
frutas="maçã banana laranja"
for fruta in $frutas; do
    echo "Fruta: $fruta"
done
```

🌀 While Loop

bash

```bash
#!/bin/bash

# Contador
contador=1
while [ $contador -le 5 ]; do
    echo "Volta $contador"
    contador=$((contador + 1))
done

# Loop infinito (cuidado!)
while true; do
    echo "Pressione Ctrl+C para parar"
    sleep 1
done

# Ler arquivo linha por linha
while read linha; do
    echo "Linha: $linha"
done < arquivo.txt
```

🎯 Until Loop

bash

```bash
#!/bin/bash

# Executa até a condição ser verdadeira
numero=1
until [ $numero -gt 5 ]; do
    echo "Número: $numero"
    numero=$((numero + 1))
done
```

🛠️ Funções

📋 Criar Funções

bash

```bash
#!/bin/bash

# Função simples
saudar() {
    echo "Olá! Bem-vindo!"
}

# Função com parâmetros
calcular_area() {
    local largura=$1
    local altura=$2
    local area=$((largura * altura))
    echo "Área: $area m²"
}

# Função que retorna valor
dobrar() {
    local numero=$1
    local resultado=$((numero * 2))
    return $resultado
}

# Usar as funções
saudar
calcular_area 5 3

dobrar 7
echo "Resultado: $?"
```

📁 Manipulação de Arquivos

📝 Operações Básicas

bash

```bash
#!/bin/bash

# Criar arquivo
echo "Conteúdo inicial" > arquivo.txt

# Adicionar ao arquivo
echo "Nova linha" >> arquivo.txt

# Ler arquivo
while read linha; do
    echo "Li: $linha"
done < arquivo.txt

# Verificar se existe
if [ -f "arquivo.txt" ]; then
    echo "Arquivo existe!"

    # Obter informações
    echo "Tamanho: $(wc -l < arquivo.txt) linhas"
    echo "Modificado: $(date -r arquivo.txt)"
fi

# Backup de arquivo
cp arquivo.txt "arquivo_backup_$(date +%Y%m%d).txt"
```

🗂️ Trabalhar com Diretórios

bash

```bash
#!/bin/bash

# Criar diretório
mkdir -p projetos/meu_app

# Navegar
cd projetos/meu_app

# Listar conteúdo
for item in *; do
    if [ -d "$item" ]; then
        echo "📁 $item"
    elif [ -f "$item" ]; then
        echo "📄 $item"
    fi
done

# Voltar ao diretório anterior
cd -
```

🎨 Exemplos Práticos

🧹 Script de Limpeza

bash

```bash
#!/bin/bash

echo "🧹 Iniciando limpeza do sistema..."

# Limpar cache
echo "Limpando cache..."
rm -rf ~/.cache/*

# Limpar lixeira
echo "Esvaziando lixeira..."
rm -rf ~/.local/share/Trash/*

# Limpar logs antigos
echo "Removendo logs antigos..."
find /var/log -name "*.log" -mtime +30 -delete 2>/dev/null

echo "✅ Limpeza concluída!"
```

📊 Monitoramento do Sistema

bash

```bash
#!/bin/bash

echo "📊 RELATÓRIO DO SISTEMA - $(date)"
echo "================================="

# CPU
echo "🖥️ CPU:"
top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d'%' -f1

# Memória
echo "💾 Memória:"
free -h | grep Mem | awk '{print "Usada: " $3 " / Total: " $2}'

# Disco
echo "💿 Disco:"
df -h | grep -E '^/dev/' | awk '{print $1 ": " $3 "/" $2 " (" $5 ")"}'

# Processos top 5
echo "⚡ Top 5 processos:"
ps aux --sort=-%cpu | head -6 | tail -5 | awk '{print $11 " - " $3 "%"}'
```

🔄 Backup Automático

bash

```bash
#!/bin/bash

# Configurações
origem="$HOME/Documentos"
destino="$HOME/Backups"
data=$(date +%Y%m%d_%H%M%S)
arquivo_backup="backup_$data.tar.gz"

echo "📦 Criando backup..."

# Criar diretório de backup se não existir
mkdir -p "$destino"

# Criar backup compactado
tar -czf "$destino/$arquivo_backup" "$origem"

if [ $? -eq 0 ]; then
    echo "✅ Backup criado: $arquivo_backup"

    # Manter apenas os 5 backups mais recentes
    cd "$destino"
    ls -t backup_*.tar.gz | tail -n +6 | xargs rm -f

    echo "🧹 Backups antigos removidos"
else
    echo "❌ Erro ao criar backup"
fi
```

🐛 Debugging e Testes

🔍 Debug Mode

bash

```bash
#!/bin/bash

# Modo debug (mostra cada comando executado)
set -x

# Ou no shebang
#!/bin/bash -x

# Modo strict (para no primeiro erro)
set -e

# Combinação útil
set -euo pipefail
```

🧪 Testes

bash

```bash
#!/bin/bash

# Função de teste
testar_funcao() {
    local resultado=$(minha_funcao 5)
    local esperado=10

    if [ "$resultado" -eq "$esperado" ]; then
        echo "✅ Teste passou"
    else
        echo "❌ Teste falhou: esperado $esperado, obtido $resultado"
    fi
}

# Executar testes
echo "🧪 Executando testes..."
testar_funcao
```

💡 Truques e Dicas

⚡ Comandos Úteis

bash

```bash
# Executar comando e capturar saída
resultado=$(comando)
resultado=`comando`  # forma antiga

# Aritmética
soma=$((5 + 3))
divisao=$((10 / 2))

# Substituição de strings
texto="Hello World"
echo ${texto/World/Universe}  # Hello Universe

# Verificar se comando existe
if command -v git &> /dev/null; then
    echo "Git está instalado"
fi

# Timeout para comandos
timeout 5s comando_lento
```

🎨 Cores no Terminal

bash

```bash
#!/bin/bash

# Definir cores
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

echo -e "${RED}Erro!${NC}"
echo -e "${GREEN}Sucesso!${NC}"
echo -e "${YELLOW}Aviso!${NC}"
```

📋 Arrays

bash

```bash
#!/bin/bash

# Criar array
frutas=("maçã" "banana" "laranja")

# Acessar elementos
echo ${frutas[0]}  # primeira fruta
echo ${frutas[@]}  # todas as frutas

# Tamanho do array
echo ${#frutas[@]}

# Loop em array
for fruta in "${frutas[@]}"; do
    echo "Fruta: $fruta"
done
```

📚 Boas Práticas

✅ Do's (Faça)

bash

```bash
#!/bin/bash

# 1. Sempre use shebang
#!/bin/bash

# 2. Comente seu código
# Este script faz backup dos documentos

# 3. Use aspas em variáveis
echo "$nome"  # ✅ Correto
echo $nome    # ❌ Pode dar problema

# 4. Valide entrada
if [ $# -eq 0 ]; then
    echo "Uso: $0 <arquivo>"
    exit 1
fi

# 5. Use local em funções
minha_funcao() {
    local var_local="valor"
}

# 6. Trate erros
comando || {
    echo "Erro ao executar comando"
    exit 1
}
```

❌ Don'ts (Não faça)

bash

```bash
# ❌ Não use espaços ao redor do =
var = "valor"  # ERRADO

# ❌ Não esqueça aspas com espaços
arquivo="meu arquivo.txt"
rm $arquivo    # PERIGO! Remove 'meu' e 'arquivo.txt'
rm "$arquivo"  # ✅ Correto

# ❌ Não use cat desnecessariamente
cat arquivo.txt | grep "texto"  # Lento
grep "texto" arquivo.txt        # ✅ Mais eficiente
```

🎯 Exemplo Final: Script Completo

bash

```bash
#!/bin/bash
set -euo pipefail

# Script de deploy automático
# Autor: Seu Nome
# Data: $(date)

# Configurações
readonly PROJETO="meu-app"
readonly SERVIDOR="usuario@servidor.com"
readonly DIRETORIO="/var/www/html"

# Cores
readonly RED='\033[0;31m'
readonly GREEN='\033[0;32m'
readonly YELLOW='\033[1;33m'
readonly NC='\033[0m'

# Função de log
log() {
    echo -e "${GREEN}[INFO]${NC} $1"
}

erro() {
    echo -e "${RED}[ERRO]${NC} $1" >&2
}

aviso() {
    echo -e "${YELLOW}[AVISO]${NC} $1"
}

# Validar argumentos
if [ $# -ne 1 ]; then
    erro "Uso: $0 <versao>"
    exit 1
fi

versao=$1

# Função principal
main() {
    log "Iniciando deploy da versão $versao"

    # Verificar se git está limpo
    if [ -n "$(git status --porcelain)" ]; then
        erro "Existem mudanças não commitadas"
```

```
    exit 1
  fi

  # Executar testes
  log "Executando testes..."
  npm test || {
    erro "Testes falharam"
    exit 1
  }

  # Build
  log "Fazendo build..."
  npm run build

  # Upload
  log "Enviando para servidor..."
  rsync -avz --delete dist/ "$SERVIDOR:$DIRETORIO"

  # Tag no git
  git tag "v$versao"
  git push origin "v$versao"

  log "Deploy concluído com sucesso! 🎉"
}

# Executar função principal
main
```

🏁 Para Começar

🗓️ Roteiro de Estudos

Dia 1-2: Variáveis e echo **Dia 3-4:** Condicionais (if/else)
Dia 5-6: Loops (for/while) **Dia 7-8:** Funções **Dia 9-10:** Manipulação de arquivos **Dia 11+:** Projetos práticos

🎯 Dicas de Ouro

1. 🧪 **Teste sempre** - Execute linha por linha antes de automatizar
2. 📝 **Comente tudo** - Você vai esquecer o que fez em 1 semana
3. 🔍 **Use ShellCheck** - Site que valida seu código
4. 📚 **Comece simples** - Automatize tarefas que você já faz manualmente
5. 🤝 **Peça ajuda** - `man comando` é seu amigo

🚀 Próximos Passos

- Aprenda **regex** para manipulação avançada de texto
- Estude **awk** e **sed** para processamento de dados
- Explore **cron** para automatizar execução
- Pratique com projetos reais do seu dia a dia

Lembre-se: Shell Script é sobre **automatizar o chato** para focar no que importa! 🤖✨