INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE ALAGOAS
CURSO TÉCNICO INTEGRADO EM INFORMÁTICA

MARCOS ANTHONY RODRIGUES CARDOSO
PAULO HENRIQUE ALVES DA SILVA
ROBERT RICARDO CALHEIRO DA SILVA
THIAGO ROGÉRIO DE MELO

PROJETO FINAL SHELL SCRIPT

Banco Caixa

RIO LARGO
2023

MARCOS ANTHONY RODRIGUES CARDOSO
PAULO HENRIQUE ALVES DA SILVA
ROBERT RICARDO CALHEIRO DA SILVA
THIAGO ROGÉRIO DE MELO

PROJETO FINAL SHELL SCRIPT

Banco Caixa

Trabalho avaliativo, apresentado à disciplina de Sistemas Operacionais, como parte das exigências para a obtenção da nota do 4° bimestre sob orientação do prof. Me. Pablo Tiburcio.

RIO LARGO
2023

SUMÁRIO

INTRODUÇÃO

Este relatório descreve o desenvolvimento de um script em Shell para simular operações bancárias comuns, como saque, depósito, transferência, entre outras funcionalidades, de forma prática e intuitiva. O script foi criado para replicar as operações disponíveis em um sistema bancário, especificamente o da Caixa, utilizando if/else/elif, case, for, while, until, funções, e a ferramenta ‘Dialog‘ para uma interface de linha de comando.

FUNCIONALIDADES

O script desenvolvido apresenta as seguintes funcionalidades:

- Saque: Permite ao usuário simular um saque, dividindo o valor inserido em notas de diferentes valores.
- Depósito: Simula a operação de depósito de um valor especificado pelo usuário.
- Transferência: Simula a transferência de dinheiro para outra conta, solicitando o valor e o número da conta de destino.
- Menu de Opções: Oferece um menu interativo para o usuário escolher a operação desejada.
- Tratamento de Erros: Verifica se os valores inseridos são numéricos e trata cancelamentos do usuário.

ESTRUTURAS DO CÓDIGO

Estruturas condicionais/controle: if/else/elif

As estruturas if, else, e elif são usadas para tomar decisões com base nas condições. Por exemplo, no código:

if [$? -eq 0]

Aqui, verifica-se se o código de retorno do comando anterior é zero (indicando sucesso). Se for o caso, o bloco de código dentro do if é executado. Caso contrário, o bloco dentro do else seria executado. Utiliza-se para verificar se o usuário pressionou "OK" ao fornecer uma entrada.

Estruturas condicionais/controle: case

A estrutura case é usada para realizar ações diferentes com base no valor de uma variável. No código, temos:

```
case $escolha in
    1)
        realizar_saque
        ;;
    2)
        realizar_deposito
        ;;
    3)
        realizar_transferencia
        ;;
    0)
        echo "Saindo do programa."
        exit 0
        ;;
    *)
        dialog --msgbox "Opção inválida. Por favor, escolha uma opção válida." 8 40
        ;;
esac
```

Aqui, a variável $escolha determina qual opção foi escolhida pelo usuário, e o código executa a função correspondente.

Estruturas de repetição: for

A estrutura de repetição for é utilizada para percorrer os elementos de um array que representa os valores das notas de dinheiro. Aqui está a parte específica do código onde o for é empregado:

```
# Calcula a quantidade de cada nota
for nota in "${notas[@]}"; do
    qtd=$((valor / nota)) # Calcula a quantidade de notas necessárias
    valor=$((valor % nota)) # Calcula o valor restante para as próximas notas

    notas_quantidade+=("$qtd")
done
```

Aqui estão os elementos principais do bloco for:

- for nota in "${notas[@]}"; do: Isso itera sobre cada elemento do array notas. O valor atual do elemento é armazenado na variável nota em cada iteração.
- qtd=$((valor / nota)): Calcula a quantidade de notas necessárias para o valor atual. Esta linha usa a divisão inteira para determinar quantas vezes a nota pode ser usada no valor total.
- valor=$((valor % nota)): Calcula o valor restante após usar as notas calculadas. O operador % é o operador módulo, que retorna o resto da divisão.
- notas_quantidade+=("$qtd"): Adiciona a quantidade de notas calculadas ao array notas_quantidade.

Estruturas de repetição: while

A estrutura while é usada para criar um loop que continua enquanto a condição especificada for verdadeira. No código:

```
# Loop principal usando while para manter o programa em execução
while true; do
    escolha=$(dialog --clear --backtitle "Opções da Caixa" --title "Escolha uma opção" \
        --menu "Selecione uma opção:" 15 60 9 \
        1 "Saque" \
        2 "Depósito" \
        3 "Transferência" \
        0 "Sair" \
        3>&1 1>&2 2>&3)

    case $escolha in
        1)
            realizar_saque
            ;;
        2)
            realizar_deposito
            ;;
        3)
            realizar_transferencia
            ;;
        0)
            echo "Saindo do programa."
            exit 0
            ;;
        *)
            dialog --msgbox "Opção inválida. Por favor, escolha uma opção válida." 8 40
            ;;
    esac
done
```

Este loop mantém o programa em execução continuamente até que seja interrompido explicitamente (por exemplo, ao escolher a opção "Sair").

Estruturas de repetição: until

A estrutura until é usada para criar um loop que continua até que a condição especificada seja verdadeira. No código:

```
        until [[ $valor =~ ^[0-9]+$ ]]; do
            if [[ ! $valor ]]; then
                echo "Operação de transferência cancelada."
                return
            fi
            dialog --msgbox "Valor inválido. Digite um valor numérico." 8 40
            valor=$(dialog --inputbox "Digite o valor da transferência:" 8 40 --stdout)
        done
```

Este loop continua até que o valor digitado seja um número inteiro.

Estruturas com funções

O código define três funções: realizar_saque, realizar_deposito e realizar_transferencia. Cada função realiza operações específicas, sendo simulação de um saque, depósito ou transferência, respectivamente.

- realizar_saque:

```
# Função para realizar um saque (mantida igual)
realizar_saque() {
    valor=$(dialog --inputbox "Digite o valor do saque:" 8 40 --stdout)
    if [ $? -eq 0 ]; then # Verifica se o usuário pressionou OK
        if [[ $valor =~ ^[0-9]+$ ]]; then
            dialog --msgbox "Simulando operação de saque de R\$$valor." 8 40

            notas=(200 100 50 20 10 5 2 1) # Array com os valores das notas
            notas_quantidade=() # Array para armazenar a quantidade de cada nota

            # Calcula a quantidade de cada nota
            for nota in "${notas[@]}"; do
                qtd=$((valor / nota)) # Calcula a quantidade de notas necessárias
                valor=$((valor % nota)) # Calcula o valor restante para as próximas notas

                notas_quantidade+=("$qtd")
            done

            # Exibe a quantidade de cada nota
            dialog --msgbox "Notas:\n\nR\$200: ${notas_quantidade[0]}x\nR\$100: ${notas_quantidade[1]}x\nR\$50:
        else
            dialog --msgbox "Valor inválido. Digite um valor numérico." 8 40
        fi
    else
        echo "Operação de saque cancelada."
    fi
}
```

- realizar_deposito:

```
# Função para realizar um depósito (mantida igual)
realizar_deposito() {
    valor=$(dialog --inputbox "Digite o valor do depósito:" 8 40 --stdout)
    if [ $? -eq 0 ]; then # Verifica se o usuário pressionou OK
        if [[ $valor =~ ^[0-9]+$ ]]; then
            dialog --msgbox "Simulando operação de depósito de R\$$valor." 8 40
            # Adicionar código para operação de depósito
        else
            dialog --msgbox "Valor inválido. Digite um valor numérico." 8 40
        fi
    else
        echo "Operação de depósito cancelada."
    fi
}
```

- realizar_transferencia:

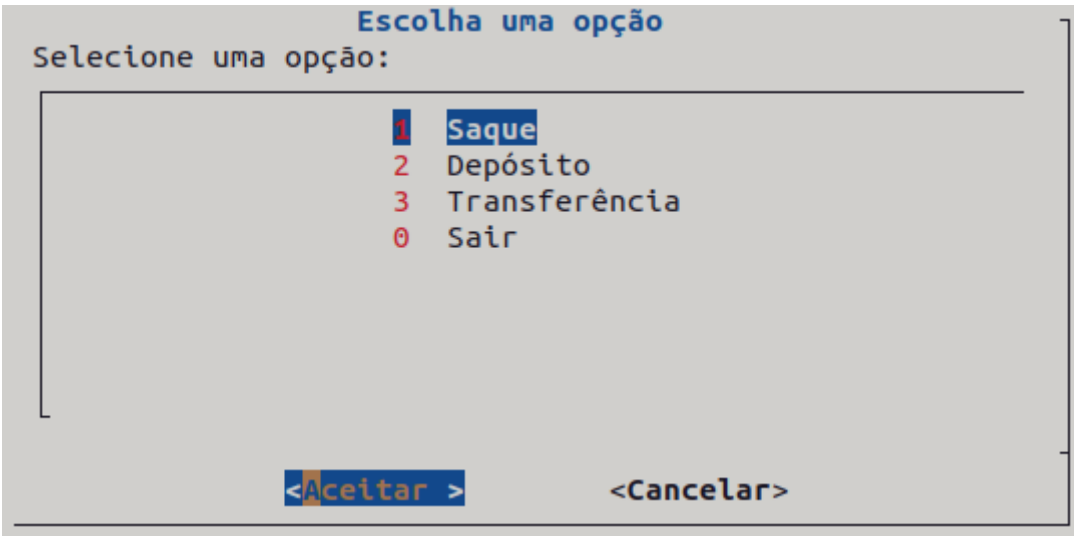
```
# Função para realizar uma transferência
realizar_transferencia() {
    local valor
    valor=$(dialog --inputbox "Digite o valor da transferência:" 8 40 --stdout)
    if [ $? -eq 0 ]; then # Verifica se o usuário pressionou OK
        until [[ $valor =~ ^[0-9]+$ ]]; do
            if [[ ! $valor ]]; then
                echo "Operação de transferência cancelada."
                return
            fi
            dialog --msgbox "Valor inválido. Digite um valor numérico." 8 40
            valor=$(dialog --inputbox "Digite o valor da transferência:" 8 40 --stdout)
        done

        destino=$(dialog --inputbox "Digite a conta de destino:" 8 40 --stdout)
        if [ $? -eq 0 ]; then # Verifica se o usuário pressionou OK
            dialog --msgbox "Simulando operação de transferência de R\$$valor para a conta $destino." 8 60
            # Adicionar código para operação de transferência
        else
            echo "Operação de transferência cancelada."
        fi
    else
        echo "Operação de transferência cancelada."
    fi
}
```

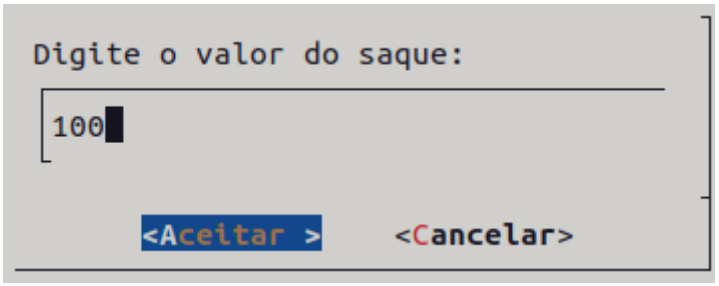
Dialog

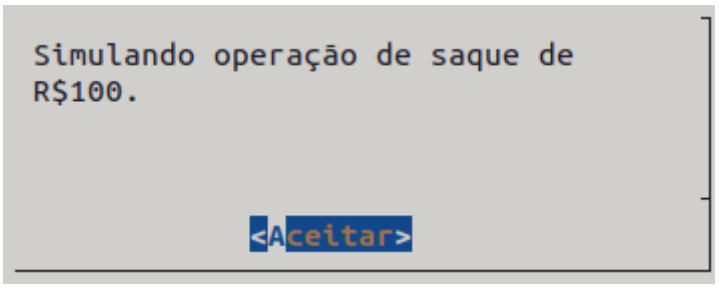
O comando dialog é utilizado para exibir caixas de diálogo interativas no terminal. Ele é usado para obter entradas do usuário e exibir mensagens informativas.

- Escolher opção:

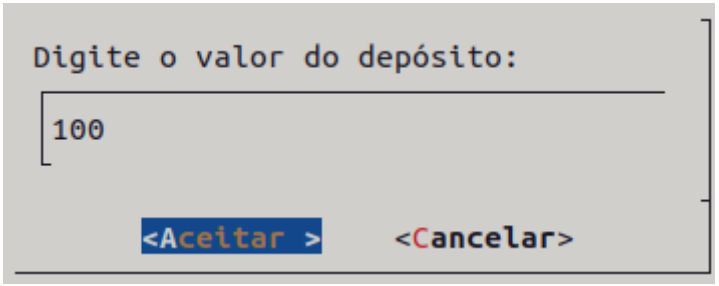


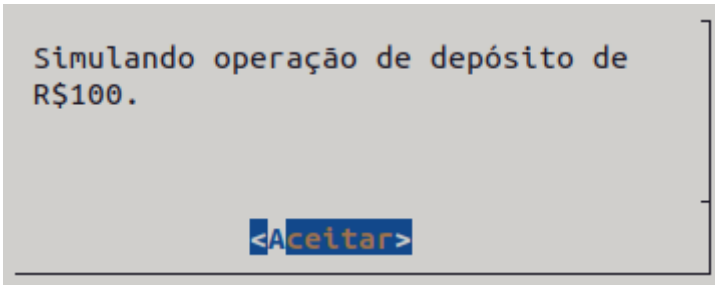
- Saque:



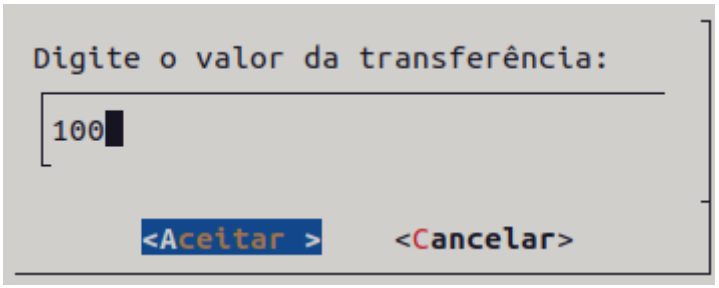


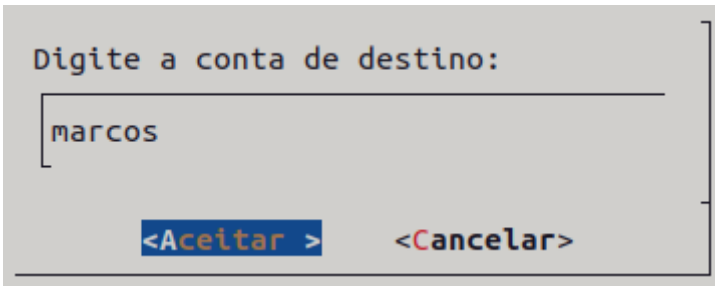
- Depósito:

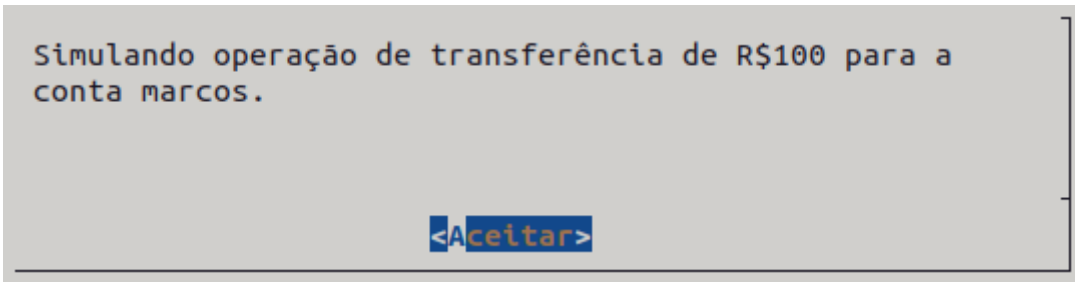




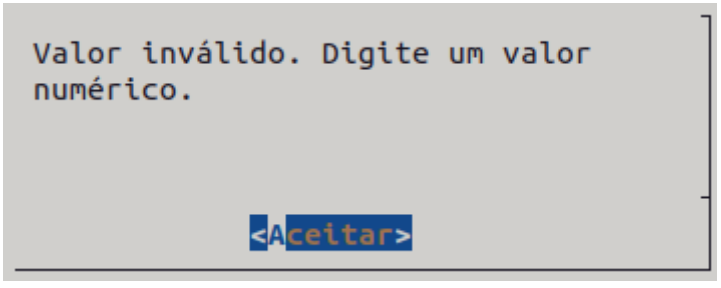
- Transferência:







- Validação: aparece quando o valor digitado em uma das opções é invalido, ou seja, não é numérico.



CONCLUSÃO

Portanto, o script foi desenvolvido através da escolha da empresa de banco Caixa, para a responder a sua demanda com uma estrutura inicial que simula operações bancárias básicas, utilizando if/else/elif, case, for, while, until, funções, e dialog. No entanto, para ser utilizado em um ambiente real, seria necessário expandir a lógica para incluir funcionalidades bancárias autênticas e garantir a segurança e precisão das operações.