МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## ***Факультет информационных технологий и робототехники***

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 4**

по дисциплине:” Системное программирование ”

на тему: ”Скрипты, создание проекта, диалоги ***”***

Выполнил**:** студент группы 10701321 Кругликовский М.А.

Принял**:** ст.пр. Станкевич C.П.

Минск 2023

# **Лабораторная работа № 4**

**Цель работы**

Закрепить на практике принципы создания проектов с помощью скриптов SHELL, освоить средства примитивного графического интерфейса в Linux-скриптах.

**Задание**

**Задание 1**

1. Сделать пример с двумя вложенными диалогами типа YesNo.

2. Сделать пример, запрашивающий сначала имя человека, а потом профессию. Вывести имя + профессию, прочитанные в диалоге.

3. Вывести список с названиями валют. После выбора валюты система должна вывести ее котировку.

Здесь следует обратить внимание на проверку условия. Запомните, если выполняем сравнение строк, то в квадратных скобках строки записываем в двоичных кавычках. При этом (!) нужно разделять все слова пробелами как слева, так и справа. Наконец, слово then договоримся писать на новой строке. Заметим, что диалог отображается списком, поскольку указаны символы перехода на новую строку в команде dialog.

4. Измените предыдущую программу так, вместо списка валют предлагалось оконное меню валют. Чтобы программа работала в цикле. Для выхода из цикла нужно вместо названия валюты вводить exit.

5. Измените предыдущую программу так, для выхода из программы в меню была кнопка закрытия программы «exit».

Работа диалогов должна быть подтверждена скриншотами в отчетах.

**Задание 2**

Создайте копию проекта представелнного в упражнении №3. Внесем изменеия в новом проекте, добавив вывод информации о домашнем каталоге каждого пользователя и включив в вывод общее число файлов и подкаталогов в каждом из них:

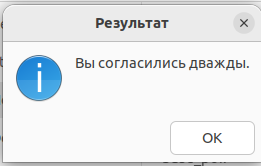
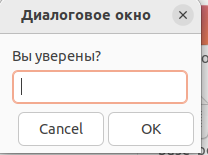
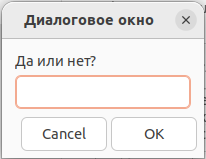
В этой новой версии проверяется наличие привилегий суперпользователя, но вместо того, чтобы выполнить полный набор операций в каждой из ветвей if, здесь устанавливаются некоторые переменные, которые затем используются в цикле for. В функции использованы несколько локальных переменных и команда printf для форматирования части вывода.

**Задание 3**

Для сценария выполненного в задании №2 создайте графическое диалоговое окно.

**Результаты выполнения**

**Задание 1**

****

Рисунки 1-3. Пример с двумя вложенными диалогами типа YesNo

#!/bin/bash

answer1=$(zenity --entry --text "Да или нет?" --title "Диалоговое окно")

if [ "$answer1" == "да" ]; then

answer2=$(zenity --entry --text "Вы уверены?" --title "Диалоговое окно")

if [ "$answer2" == "да" ]; then

zenity --info --text "Вы согласились дважды." --title "Результат"

else

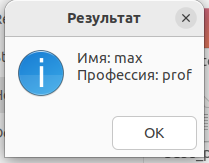
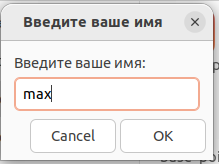
zenity --info --text "Вы изменили решение." --title "Результат"

fi

else

zenity --info --text "Вы отказались." --title "Результат"

fi



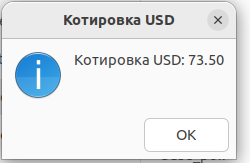
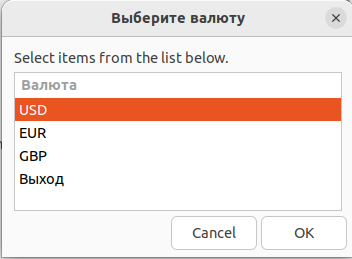
Рисунки 4-5. Скрипт для запроса имени и профессии

#!/bin/bash

name=$(zenity --entry --title "Введите ваше имя" --text "Введите ваше имя:")

profession=$(zenity --entry --title "Введите вашу профессию" --text "Введите вашу профессию:")

zenity --info --title "Результат" --text "Имя: $name\nПрофессия: $profession"



Рисунки 6-7. Скрипт для вывода списка валют и их котировок

**Без цикла**

#!/bin/bash

currencies=("USD" "EUR" "GBP")

choice=$(zenity --list --title "Выберите валюту" --column "Валюта" "${currencies[@]}" "Выход")

case $choice in

"USD")

zenity --info --title "Котировка USD" --text "Котировка USD: 73.50"

;;

"EUR")

zenity --info --title "Котировка EUR" --text "Котировка EUR: 86.20"

;;

"GBP")

zenity --info --title "Котировка GBP" --text "Котировка GBP: 101.80"

;;

"Выход")

exit 0

;;

\*)

zenity --error --title "Ошибка" --text "Выбранной валюты нет в списке."

;;

Esac

**С циклом**

#!/bin/bash

while true; do

choice=$(zenity --list --column="Валюта" "USD" "EUR" "GBP" "Выход")

case $choice in

"Выход")

exit 0

;;

"USD")

zenity --info --text="Котировка USD: 73.50"

;;

"EUR")

zenity --info --text="Котировка EUR: 86.20"

;;

"GBP")

zenity --info --text="Котировка GBP: 101.80"

;;

esac

done

**Задание 2**

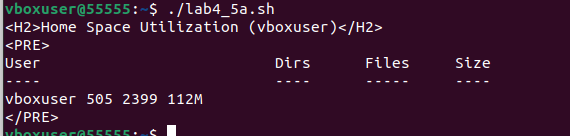


Рисунок 6. Копия проекта представелнного в упражнении №3.

#!/bin/bash

report\_home\_space () {

local format="%-30s%-10s%-10s%-10s\n"

local dir\_list total\_files total\_dirs total\_size

if [[ $(id -u) -eq 0 ]]; then

dir\_list="/home/\*"

echo "<H2>Home Space Utilization (All Users)</H2>"

else

dir\_list="$HOME"

echo "<H2>Home Space Utilization ($USER)</H2>"

fi

echo "<PRE>"

printf "$format" "User" "Dirs" "Files" "Size"

printf "$format" "----" "----" "-----" "----"

for i in $dir\_list; do

total\_files=$(find "$i" -type f | wc -l)

total\_dirs=$(find "$i" -type d | wc -l)

total\_size=$(du -sh "$i" | cut -f 1)

echo "$(basename $i) $total\_dirs $total\_files $total\_size"

done

echo "</PRE>"

}

report\_home\_space

**Задание 3**

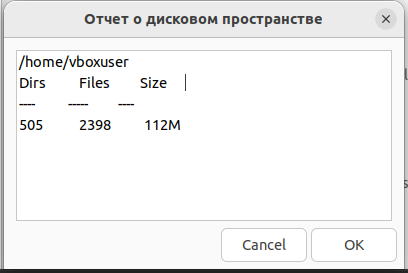


Рисунок 15. Сценарий выполненного в задании №2 создан графическое диалоговое окно.

#!/bin/bash

report\_home\_space () {

local format="%-15s%-15s%-15s\n"

local i dir\_list total\_files total\_dirs total\_size user\_name

if [[ $(id -u) -eq 0 ]]; then

dir\_list="/home/\*"

user\_name="All Users"

else

dir\_list="$HOME"

user\_name="$USER"

fi

echo "Home Space Utilization ($user\_name)"

for i in $dir\_list; do

total\_files=$(find "$i" -type f | wc -l)

total\_dirs=$(find "$i" -type d | wc -l)

total\_size=$(du -sh "$i" | cut -f 1)

zenity --text-info --width=400 --title="Отчет о дисковом пространстве" --filename=/dev/stdin <<EOF

$i

$(printf "$format" "Dirs" "Files" "Size")

$(printf "$format" "----" "-----" "----")

$(printf "$format" "$total\_dirs" "$total\_files" "$total\_size")

EOF

done

}

report\_home\_space

**Контрольные вопросы**

**Что такое область видимости переменных? Какие типы области видимости переменных существуют?**

Область видимости переменных - это концепция в программировании, которая определяет, где в коде переменная может быть доступна или использована. Существует несколько типов области видимости переменных:

**Локальная область видимости (Local Scope):**

Переменные, объявленные внутри функции или блока кода, имеют локальную область видимости.

Эти переменные видимы только в пределах функции или блока, в котором они объявлены.

Они недоступны за пределами этой функции или блока.

**Глобальная область видимости (Global Scope):**

Переменные, объявленные вне всех функций и блоков кода, имеют глобальную область видимости.

Они видимы и доступны для всех частей кода в программе, включая функции и блоки.

Глобальные переменные обычно объявляются в начале программы и используются в разных частях кода.

**Область видимости вложенных функций (Nested Function Scope):**

Если функция объявлена внутри другой функции, она может иметь доступ к переменным своей родительской функции.

Переменные внешней (родительской) функции могут быть доступны внутри внутренней (дочерней) функции, но не наоборот.

**Блочная область видимости (Block Scope):**

Начиная с языка программирования, таких как C++, C#, JavaScript (последние версии), переменные могут иметь область видимости ограниченную блоками кода, например, блоками if, for, while, и так далее.

Переменные, объявленные внутри таких блоков, видимы только в пределах этого блока и не доступны за его пределами.

**Область видимости модулей (Module Scope):**

В некоторых языках программирования, таких как Python, модули могут иметь свою собственную область видимости.

Переменные, объявленные внутри модуля, видимы только внутри этого модуля.

**Область видимости анонимных функций (Closure Scope):**

В некоторых языках, например, JavaScript, анонимные функции (замыкания) могут иметь доступ к переменным из внешних областей видимости.

Эта область видимости позволяет сохранять состояние между вызовами анонимных функций.

**Какие существуют типы окон примитивного графического**

**интерфейса Linux-скриптов?**

В Linux существует несколько типов окон примитивного графического интерфейса, которые могут быть использованы в скриптах. Ниже приведены некоторые из них:

**Xdialog:**

Xdialog предоставляет набор диалоговых окон и позволяет создавать графические пользовательские интерфейсы в скриптах с помощью диалогов, таких как сообщения, ввод текста, списки и другие.

Этот инструмент основан на библиотеке диалогов (Dialog) и предоставляет более широкие возможности для создания интерфейсов.

**Zenity:**

Zenity — это утилита командной строки для создания диалоговых окон GTK+ в скриптах.

Она позволяет создавать простые окна для ввода текста, выбора файлов, списков и других элементов интерфейса.

**YAD (Yet Another Dialog):**

YAD — это форк Zenity, предоставляющий дополнительные возможности и опции для создания окон с более сложными элементами интерфейса, такими как вкладки, значки и т. д.

**Xmessage:**

Xmessage — это стандартный инструмент X Window System, который позволяет создавать простые диалоговые окна с текстовым содержимым.

**Xterm:**

Xterm — это стандартный эмулятор терминала в среде X Window System, который может использоваться для создания текстовых диалоговых окон в командной строке.

**Whiptail:**

Whiptail — это инструмент командной строки, который предоставляет текстовые диалоги в стиле ncurses. Он часто используется в скриптах для создания интерактивных текстовых интерфейсов.

**Tkinter:**

Tkinter — это библиотека для создания графических пользовательских интерфейсов на языке программирования Python. Она позволяет создавать оконные приложения с различными виджетами и элементами интерфейса.