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: ”Командный язык и скрипты Shell.

Часть 2 — Скрипты, создание проекта, диалоги***”***

Выполнил**:** студент группы 10702221

Будчанин В.А.

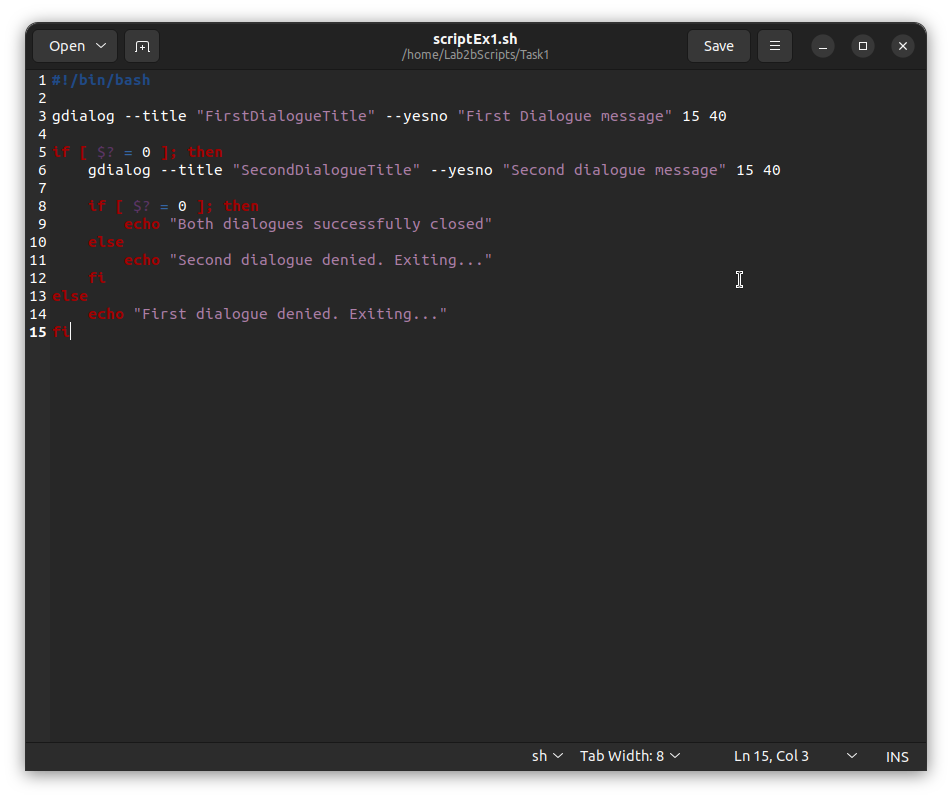
Принял**:** Давыденко Н.В.

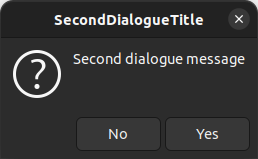
Минск, 2023

# Лабораторная работа № 4. Командный язык и скрипты Shell. Часть 2.

***Задание 1:***

*1.1. Сделать пример с двумя вложенными диалогами типа YesNo.*

*Рис. 1.1 - Окно gedit скрипта*

*Рис. 1.2 - Диалоговые окна*

**Скрипт:**

**#!/bin/bash**

**gdialog --title "FirstDialogueTitle" --yesno "First Dialogue message" 15 40**

**if [ $? = 0 ]; then**

**gdialog --title "SecondDialogueTitle" --yesno "Second dialogue message" 15 40**

**if [ $? = 0 ]; then**

**echo "Both dialogues successfully closed"**

**else**

**echo "Second dialogue denied. Exiting..."**

**fi**

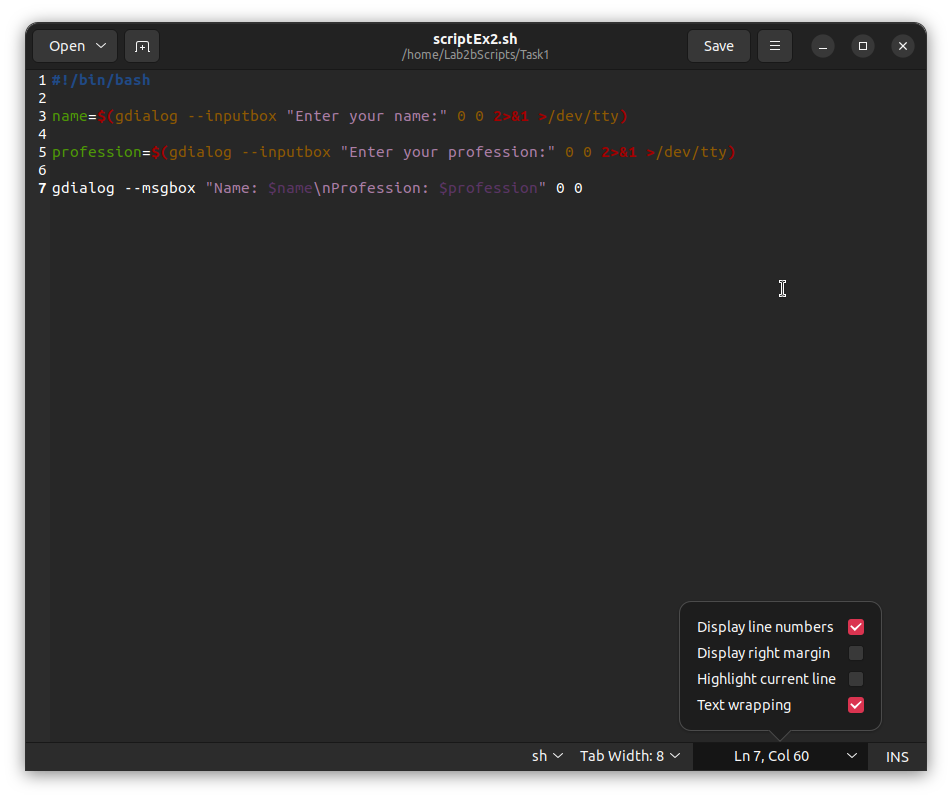
**else**

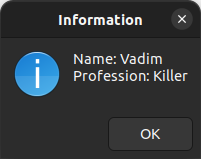
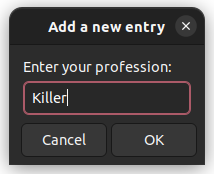
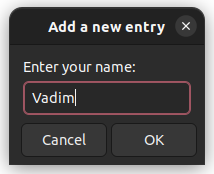
**echo "First dialogue denied. Exiting..."**

**fi**

*1.2. Сделать пример, запрашивающий сначала имя человека, а потом*

*профессию. Вывести имя + профессию, прочитанные в диалоге.*

*Рис. 2 - Окно gedit скрипта*

*Рис. 2.2 - Диалоговые окна*

**Скрипт:**

**#!/bin/bash**

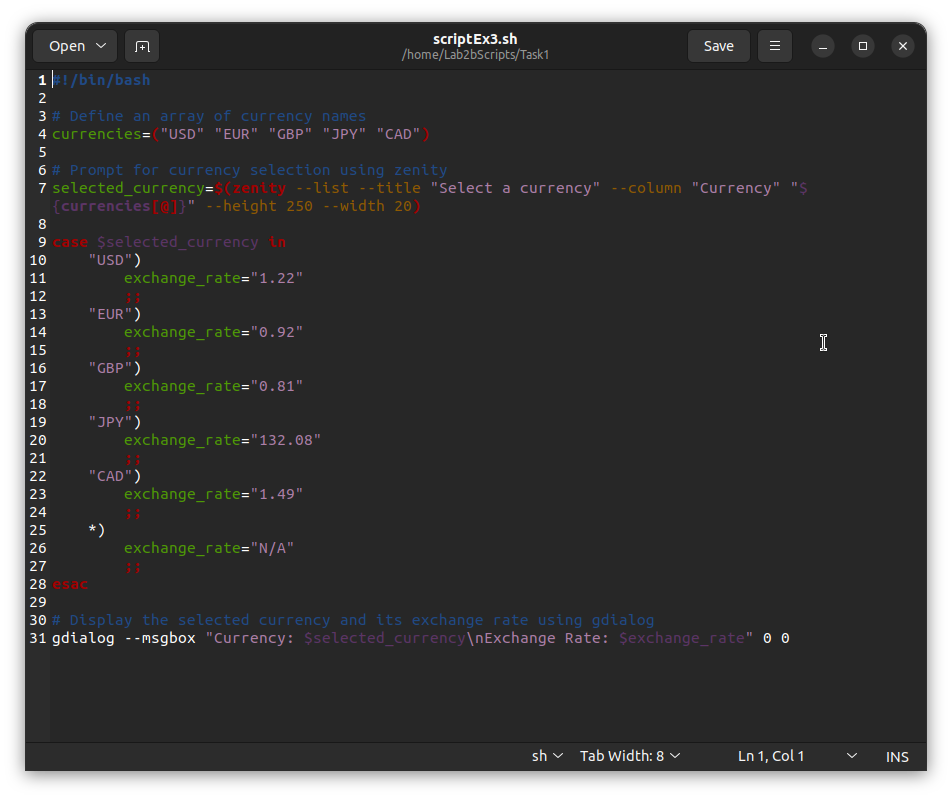
**name=$(gdialog --inputbox "Enter your name:" 0 0 2>&1 >/dev/tty)**

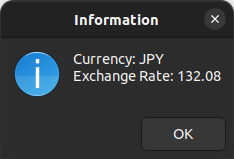
**profession=$(gdialog --inputbox "Enter your profession:" 0 0 2>&1 >/dev/tty)**

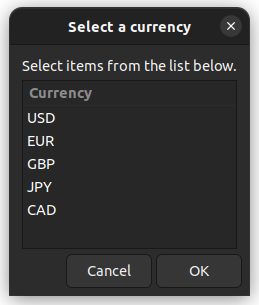
**gdialog --msgbox "Name: $name\nProfession: $profession" 0 0**

*1.3. Вывести список с названиями валют. После выбора валюты система*

*должна вывести ее котировку.*

*Рис. 3.1 - Окно gedit скрипта*

**

*Рис. 3.2 - Диалоговые окна*

**Скрипт:**

**#!/bin/bash**

**currencies=("USD" "EUR" "GBP" "JPY" "CAD")**

**selected\_currency=$(zenity --list --title "Select a currency" --column "Currency" "${currencies[@]}" --height 250 --width 20)**

**case $selected\_currency in**

**"USD")**

**exchange\_rate="1.22"**

**;;**

**"EUR")**

**exchange\_rate="0.92"**

**;;**

**"GBP")**

**exchange\_rate="0.81"**

**;;**

**"JPY")**

**exchange\_rate="132.08"**

**;;**

**"CAD")**

**exchange\_rate="1.49"**

**;;**

**\*)**

**exchange\_rate="N/A"**

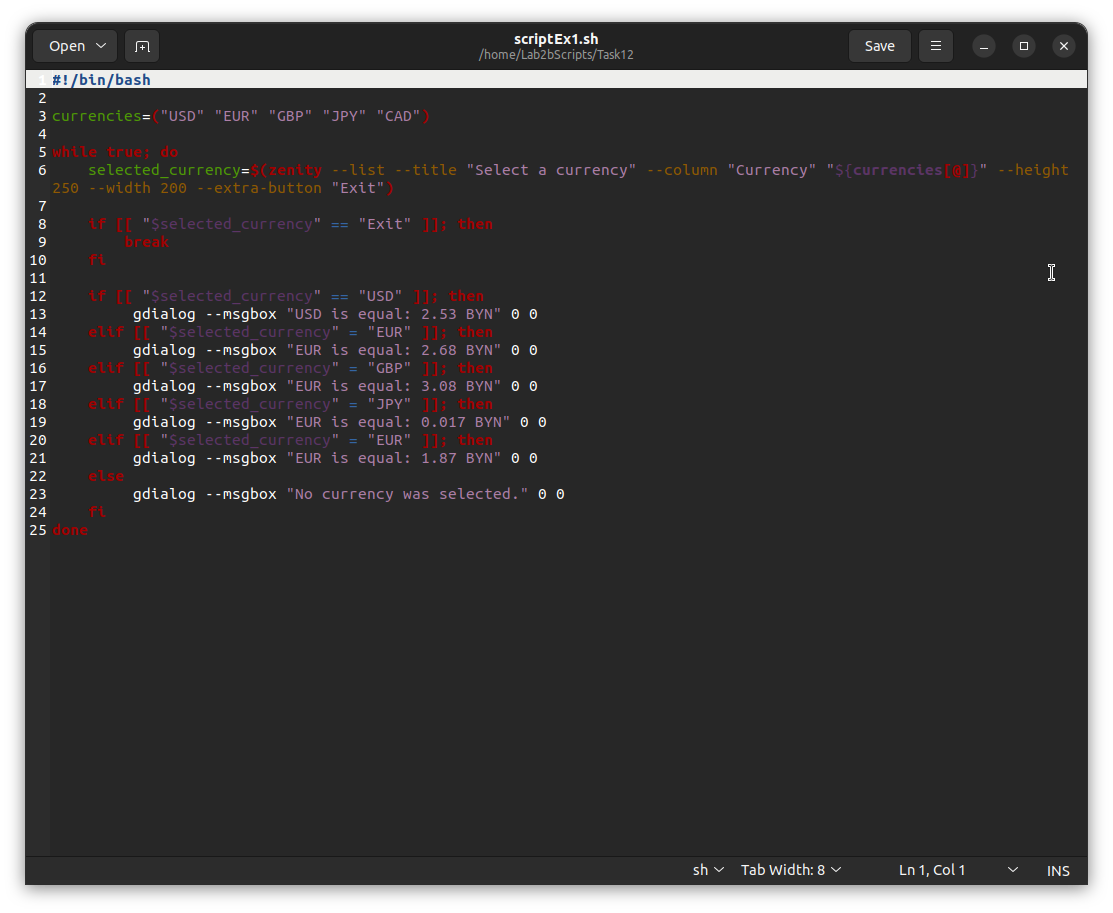
**;;**

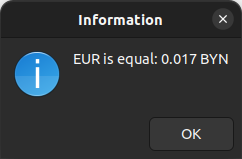
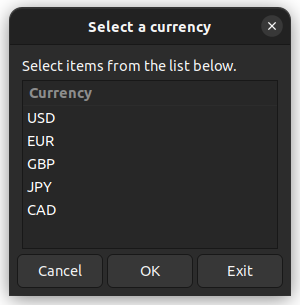
**esac**

**gdialog --msgbox "Currency: $selected\_currency\nExchange Rate: $exchange\_rate" 0 0**

*1.4-5 Вывести список с названиями валют. После выбора валюты система*

*должна вывести ее котировку.*

*Рис. 4.1 - Окно gedit скрипта*

*Рис. 4.2 - Диалоговые окна*

**Скрипт:**

**#!/bin/bash**

**currencies=("USD" "EUR" "GBP" "JPY" "CAD")**

**while true; do**

**selected\_currency=$(zenity --list --title "Select a currency" --column "Currency" "${currencies[@]}" --height 250 --width 200 --extra-button "Exit")**

**if [[ "$selected\_currency" == "Exit" ]]; then**

**break**

**fi**

**if [[ "$selected\_currency" == "USD" ]]; then**

**gdialog --msgbox "USD is equal: 2.53 BYN" 0 0**

**elif [[ "$selected\_currency" = "EUR" ]]; then**

**gdialog --msgbox "EUR is equal: 2.68 BYN" 0 0**

**elif [[ "$selected\_currency" = "GBP" ]]; then**

**gdialog --msgbox "EUR is equal: 3.08 BYN" 0 0**

**elif [[ "$selected\_currency" = "JPY" ]]; then**

**gdialog --msgbox "EUR is equal: 0.017 BYN" 0 0**

**elif [[ "$selected\_currency" = "EUR" ]]; then**

**gdialog --msgbox "EUR is equal: 1.87 BYN" 0 0**

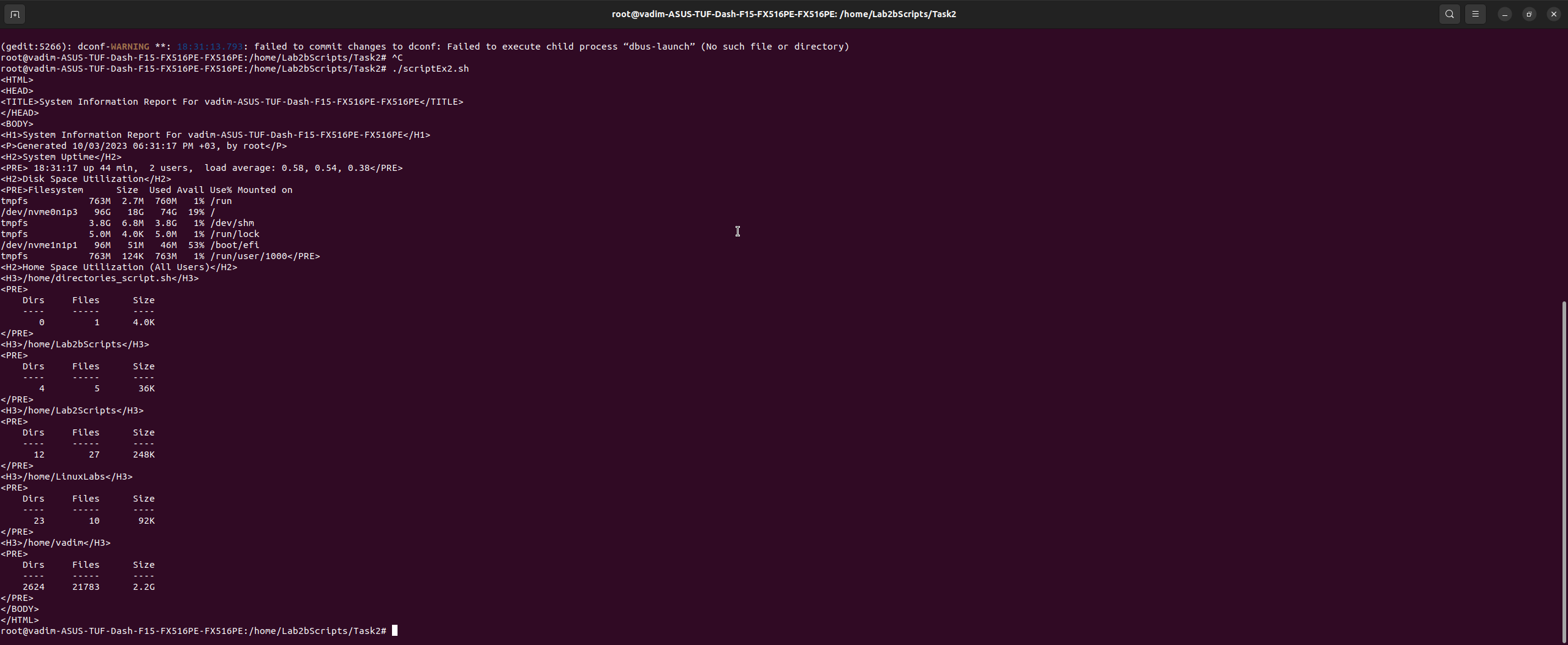
**else**

**gdialog --msgbox "No currency was selected." 0 0**

**fi**

**done**

***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.*

*Рис. 1 - Окно gedit первого скрипта второго задания*

**Скрипт:**

**#!/bin/bash**

**# sys\_info\_page: программа вывода страницы с информацией о системе**

**PROGNAME=$(basename $0)**

**TITLE="System Information Report For $HOSTNAME"**

**CURRENT\_TIME=$(date +"%x %r %Z")**

**TIME\_STAMP="Generated $CURRENT\_TIME, by $USER"**

**report\_uptime () {**

**cat <<- \_EOF\_**

**<H2>System Uptime</H2>**

**<PRE>$(uptime)</PRE>**

**\_EOF\_**

**return**

**}**

**report\_disk\_space () {**

**cat <<- \_EOF\_**

**<H2>Disk Space Utilization</H2>**

**<PRE>$(df -h)</PRE>**

**\_EOF\_**

**return**

**}**

**report\_home\_space () {**

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

**cat <<- \_EOF\_**

**<H2>Home Space Utilization (All Users)</H2>**

**<PRE>$(du -sh /home/\*)</PRE>**

**\_EOF\_**

**else**

**cat <<- \_EOF\_**

**<H2>Home Space Utilization ($USER)</H2>**

**<PRE>$(du -sh $HOME)</PRE>**

**\_EOF\_**

**fi**

**return**

**}**

**usage () {**

**echo "$PROGNAME: usage: $PROGNAME [-f file | -i]"**

**return**

**}**

**write\_html\_page () {**

**cat <<- \_EOF\_**

**<HTML>**

**<HEAD>**

**<TITLE>$TITLE</TITLE>**

**</HEAD>**

**<BODY>**

**<H1>$TITLE</H1>**

**<P>$TIME\_STAMP</P>**

**$(report\_uptime)**

**$(report\_disk\_space)**

**$(report\_home\_space)**

**</BODY>**

**</HTML>**

**\_EOF\_**

**return**

**}**

**# обработка параметров командной строки**

**interactive=**

**filename=**

**while [[ -n $1 ]]; do**

**case $1 in**

**-f | --file) shift**

**filename=$1**

**;;**

**-i | --interactive) interactive=1**

**;;**

**-h | --help)**

**usage**

**exit**

**;;**

**\*) usage >&2**

**exit 1**

**;;**

**esac**

**shift**

**done**

**# интерактивный режим**

**if [[ -n $interactive ]]; then**

**while true; do**

**read -p "Enter name of output file: " filename**

**if [[ -e $filename ]]; then**

**read -p "'$filename' exists. Overwrite? [y/n/q] > "**

**case $REPLY in**

**Y|y) break**

**;;**

**Q|q) echo "Program terminated."**

**exit**

**;;**

**\*) continue**

**;;**

**esac**

**fi**

**done**

**fi**

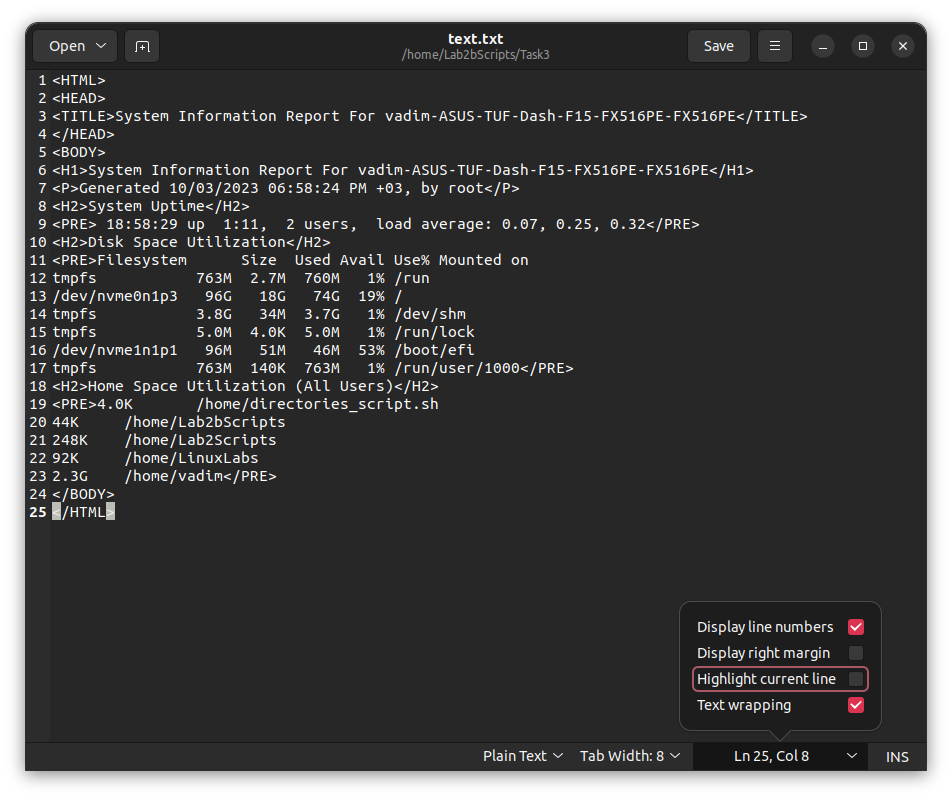
**# вывод страницы html**

**if [[ -n $filename ]]; then**

**if touch $filename && [[ -f $filename ]]; then**

**write\_html\_page > $filename**

***Задание 3****. Для сценария выполненного в задании №2 создайте графическое диалоговое окно.*

*Рис. 1 - Окно gedit файла вывода с помощью диалога*

**Скрипт:**

**#!/bin/bash**

**# sys\_info\_page: программа вывода страницы с информацией о системе**

**# Define gdialog dialog box options**

**DIALOG=${DIALOG=dialog}**

**TITLE="System Information Report For $HOSTNAME"**

**CURRENT\_TIME=$(date +"%x %r %Z")**

**TIME\_STAMP="Generated $CURRENT\_TIME, by $USER"**

**HEIGHT=15**

**WIDTH=60**

**CHOICE\_HEIGHT=4**

**BACKTITLE="System Information Report"**

**# Define functions for each section of the report**

**report\_uptime () {**

**cat <<- \_EOF\_**

**<H2>System Uptime</H2>**

**<PRE>$(uptime)</PRE>**

**\_EOF\_**

**return**

**}**

**report\_disk\_space () {**

**cat <<- \_EOF\_**

**<H2>Disk Space Utilization</H2>**

**<PRE>$(df -h)</PRE>**

**\_EOF\_**

**return**

**}**

**report\_home\_space () {**

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

**cat <<- \_EOF\_**

**<H2>Home Space Utilization (All Users)</H2>**

**<PRE>$(du -sh /home/\*)</PRE>**

**\_EOF\_**

**else**

**cat <<- \_EOF\_**

**<H2>Home Space Utilization ($USER)</H2>**

**<PRE>$(du -sh $HOME)</PRE>**

**\_EOF\_**

**fi**

**return**

**}**

**# Define usage function**

**usage () {**

**echo "$PROGNAME: usage: $PROGNAME [-f file | -i]"**

**return**

**}**

**# Define function to write HTML page**

**write\_html\_page () {**

**cat <<- \_EOF\_**

**<HTML>**

**<HEAD>**

**<TITLE>$TITLE</TITLE>**

**</HEAD>**

**<BODY>**

**<H1>$TITLE</H1>**

**<P>$TIME\_STAMP</P>**

**$(report\_uptime)**

**$(report\_disk\_space)**

**$(report\_home\_space)**

**</BODY>**

**</HTML>**

**\_EOF\_**

**return**

**}**

**# Process command line options**

**interactive=**

**filename=**

**while [[ -n $1 ]]; do**

**case $1 in**

**-f | --file) shift**

**filename=$1**

**;;**

**-i | --interactive) interactive=1**

**;;**

**-h | --help)**

**usage**

**exit**

**;;**

**\*) usage >&2**

**exit 1**

**;;**

**esac**

**shift**

**done**

**# Check if interactive mode is enabled**

**if [[ -n $interactive ]]; then**

**while true; do**

**filename=$($DIALOG --title "$BACKTITLE" --inputbox "Enter name of output file:" $HEIGHT $WIDTH 3>&1 1>&2 2>&3)**

**if [[ -e $filename ]]; then**

**choice=$($DIALOG --title "$BACKTITLE" --backtitle "$BACKTITLE" --yesno "'$filename' exists. Overwrite?" $HEIGHT $WIDTH 3>&1 1>&2 2>&3)**

**case $choice in**

**0) break;;**

**1) continue;;**

**255) echo "Program terminated.";;**

**esac**

**else**

**break**

**fi**

**done**

**fi**

**# Generate HTML page and write to file**

**if [[ -n $filename ]]; then**

**if touch $filename && [[ -f $filename ]]; then**

**write\_html\_page > $filename**

**fi**

**fi**

Вывод*:* в ходе выполнения лабораторной работы были изучены принципы написания и особенности создания bash-скриптов c использованием диалогов zenity/gdialog, а также усовершенствованы знания синтаксических и семантических особенностей кода, интерпретируемых башем.

**Контрольные вопросы:**

1. Какое типичное соглашение о написании имен переменных констант принято программистами? Для чего это делается?

Типичное соглашение о написании имен переменных и констант, которое принято программистами, включает использование стиля "camelCase" или "snake\_case". В стиле "camelCase" первая буква имени переменной начинается с маленькой буквы, а каждое последующее слово в имени начинается с заглавной буквы. Например: myVariableNameили myConstantName.

В стиле "snake\_case" все буквы написаны строчными, а слова разделяются символом подчеркивания. Например: my\_variable\_nameили my\_constant\_name.

Цель таких соглашений заключается в том, чтобы сделать код более читаемым и понятным для других программистов. Согласованное и описательное именование переменных и констант помогает лучше понимать код, уменьшает вероятность ошибок и облегчает поддержку кодовой базы. Следуя этим соглашениям, становится проще сотрудничать с другими разработчиками и поддерживать единый стиль кодирования в рамках проекта или организации.

1. Какие существуют типы окон примитивного графического интерфейса Linux-скриптов?

Существует несколько типов окон примитивного графического интерфейса для Linux-скриптов. Некоторые из них включают:

1. dialog: Это утилита командной строки, которая позволяет создавать диалоговые окна с различными элементами, такими как кнопки, списки, текстовые поля и т. д.

2. zenity: Это инструмент командной строки, который предоставляет возможность создания графических диалоговых окон с помощью GTK+.

3. whiptail: Это еще одна утилита командной строки, которая позволяет создавать диалоговые окна с различными элементами, подобными dialog, но с более простым интерфейсом.

4. curses: Это библиотека, которая позволяет создавать текстовые пользовательские интерфейсы (TUI) с использованием символьной графики. Она предоставляет функции для создания окон, кнопок, полей ввода и других элементов интерфейса. Это лишь несколько примеров типов окон примитивного графического интерфейса для Linux-скриптов. В зависимости от ваших потребностей и предпочтений, вы можете выбрать подходящий инструмент для создания пользовательского интерфейса в своих скриптах.