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 № 2b**

по дисциплине: «Системное программирование»

на тему: «Командный язык и скрипты Shell Часть 2 – Скрипты, создание проекта, диалоги.»

Выполнила: ст. гр. 10702221

Баяманова А.К.

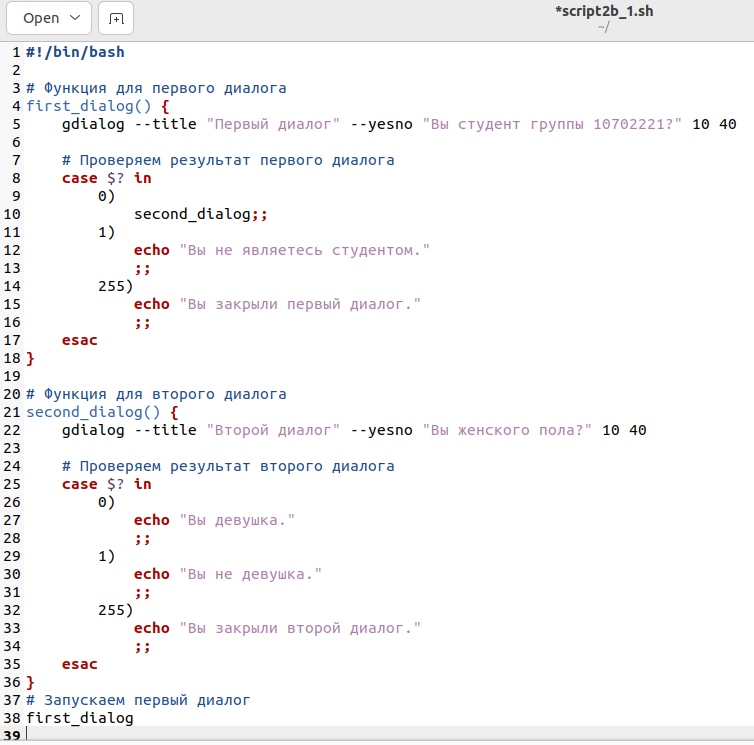
Проверил: Давыденко Н.В.

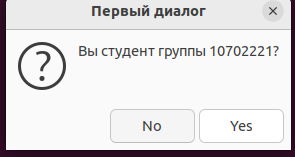
Минск 2023

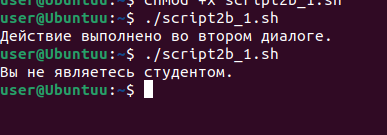
**Цель работы**: Закрепить на практике принципы создания проектов с помощью скриптов SHELL, освоить средства примитивного графического интерфейса в Linux-скриптах.

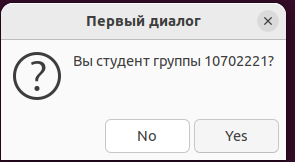
**Задание 1:**

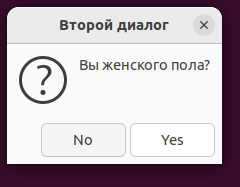
1. **Сделать пример с двумя вложенными диалогами типа YesNo.**

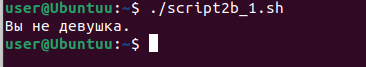


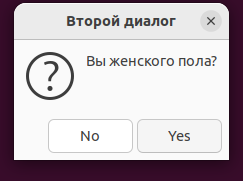


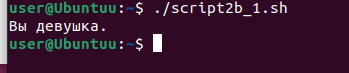




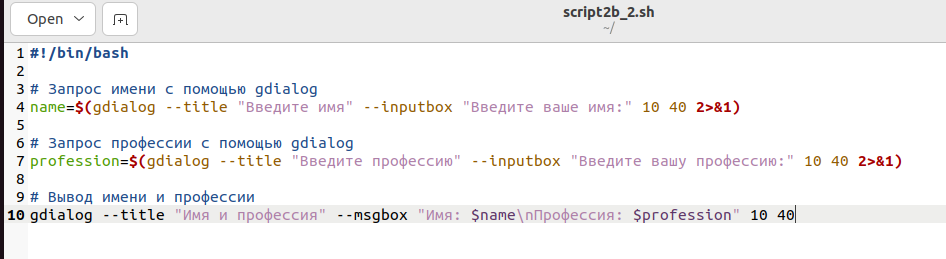


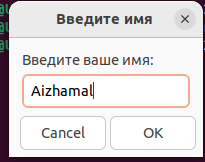


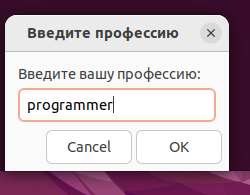


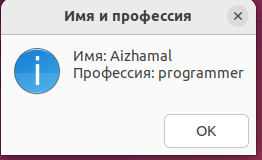


1. **Сделать пример, запрашивающий сначала имя человека, а потом профессию. Вывести имя + профессию, прочитанные в диалоге.**

****

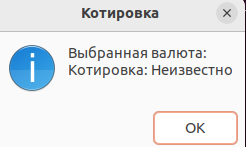
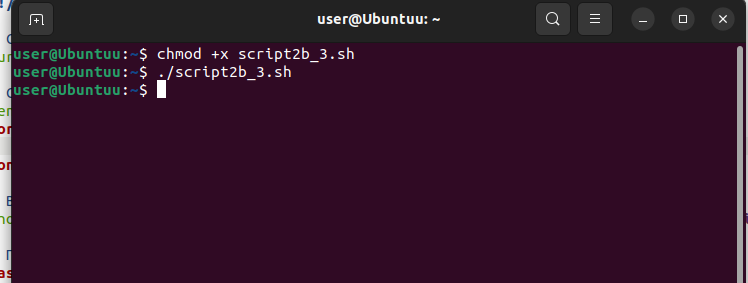
****

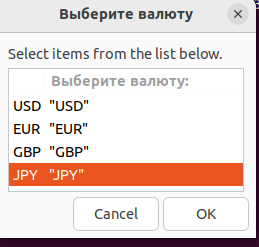
****

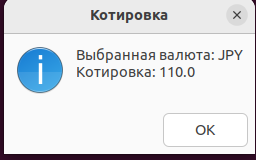
****

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.

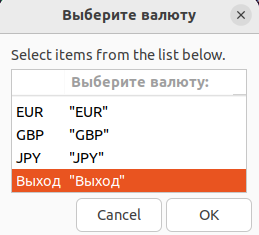






4. Измените предыдущую программу так, вместо списка валют предлагалось оконное меню валют. Чтобы программа работала в цикле. Для выхода из цикла нужно вместо названия валюты вводить exit. 5. Измените предыдущую программу так, для выхода из программы в меню была кнопка закрытия программы «exit».





**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

report\_home\_space () {

local format="%8s%10s%10s\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 "<H2>Home Space Utilization ($user\_name)</H2>"

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 "<H3>$i</H3>"

echo "<PRE>"

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

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

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

echo "</PRE>"

done

return

}

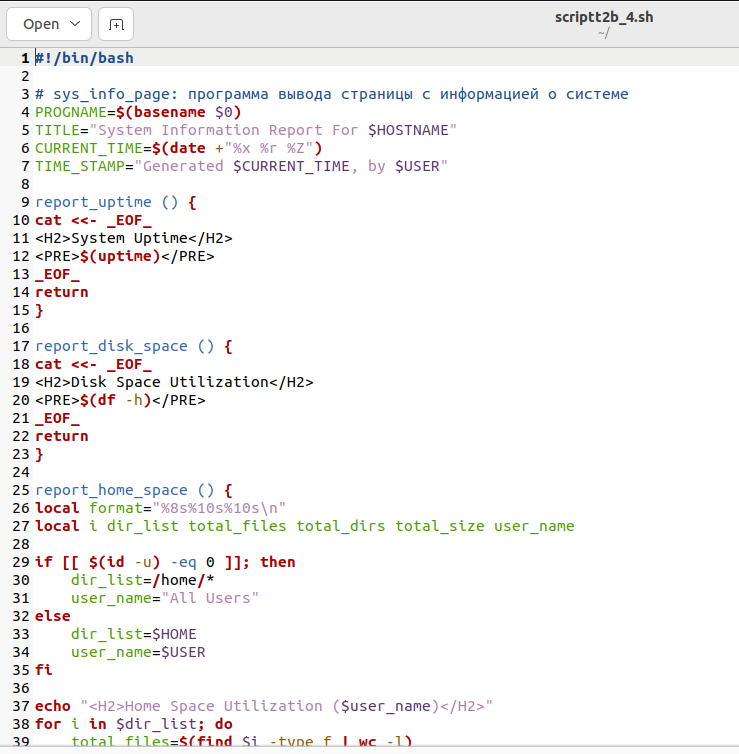
В этой новой версии проверяется наличие привилегий суперпользователя, но

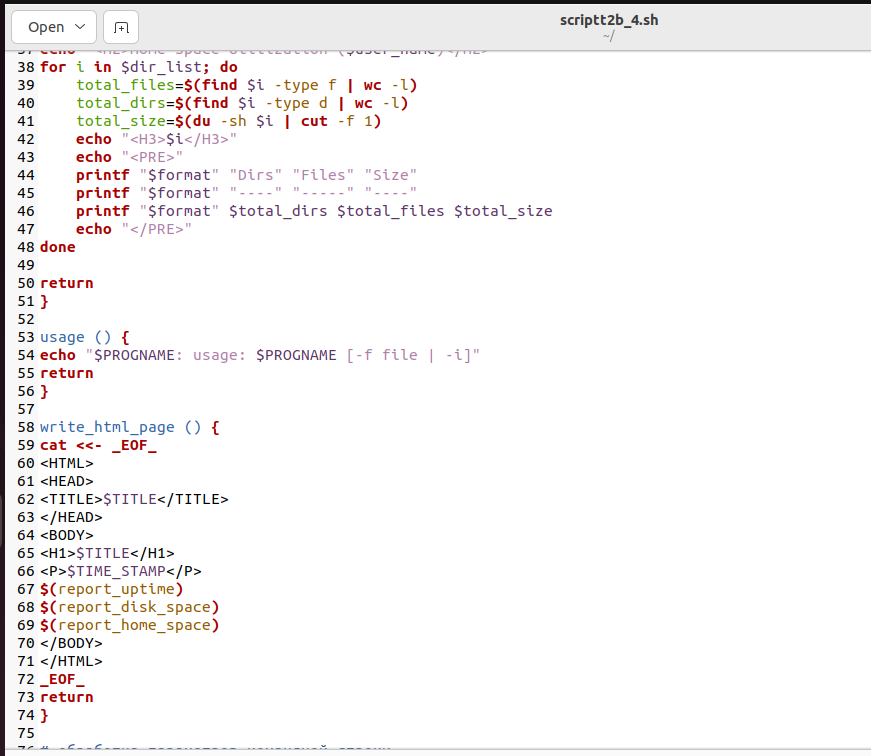
вместо того, чтобы выполнить полный набор операций в каждой из ветвей if,

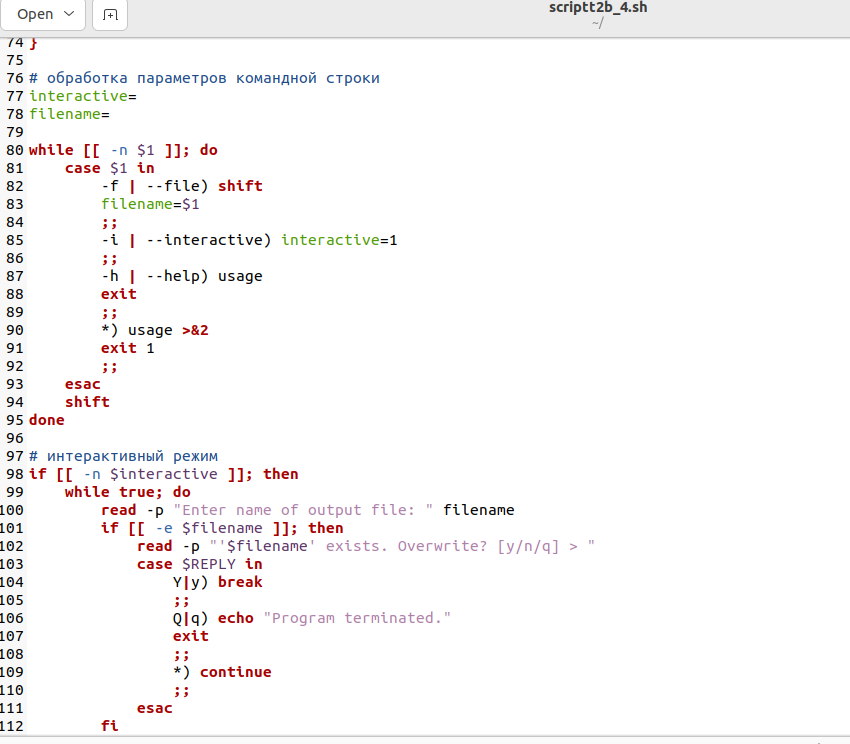
здесь устанавливаются некоторые переменные, которые затем используются в

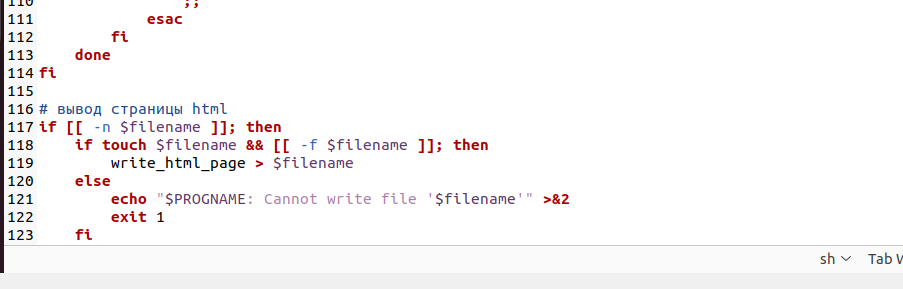
цикле for. В функции использованы несколько локальных переменных и

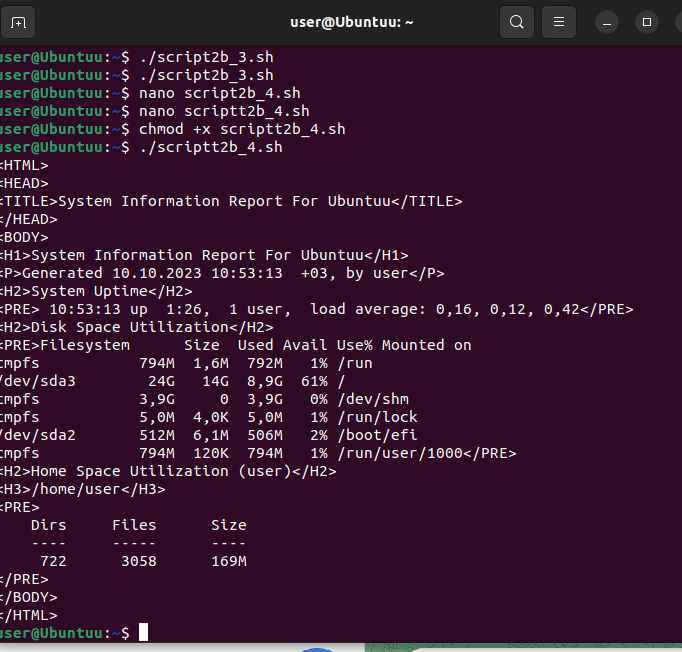
команда printf для форматирования части вывода.









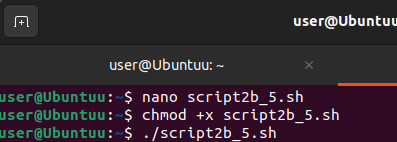


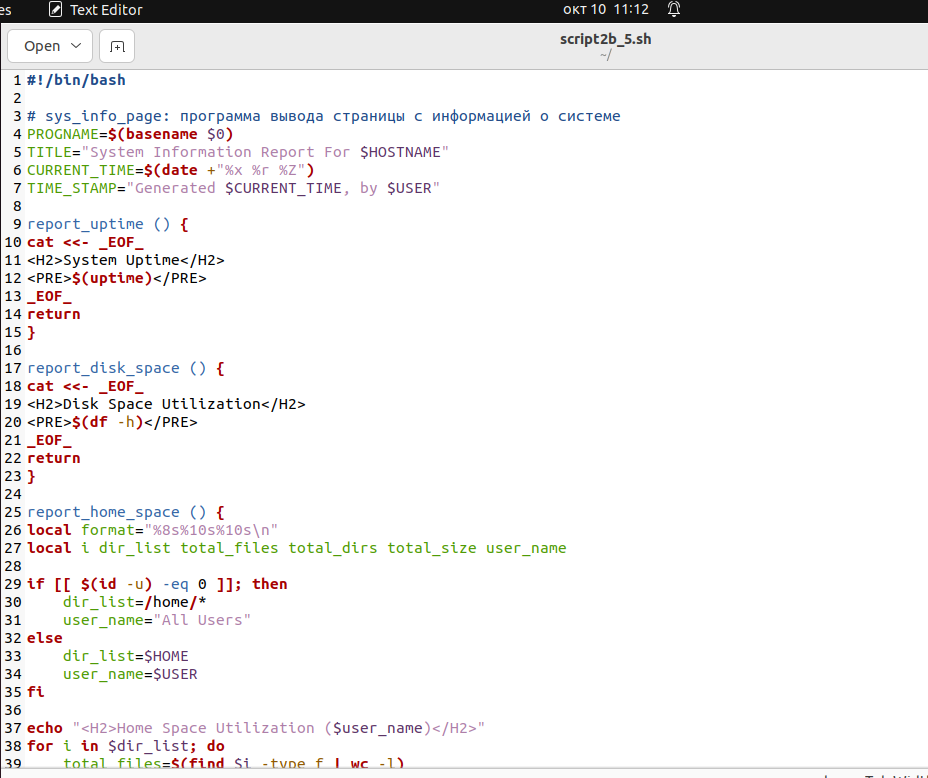
В этой версии скрипта добавлен вывод информации о домашнем каталоге каждого пользователя и общем количестве файлов и подкаталогов. Вся информация о домашних каталогах собирается в цикле, и каждый каталог отображается в отдельной секции.

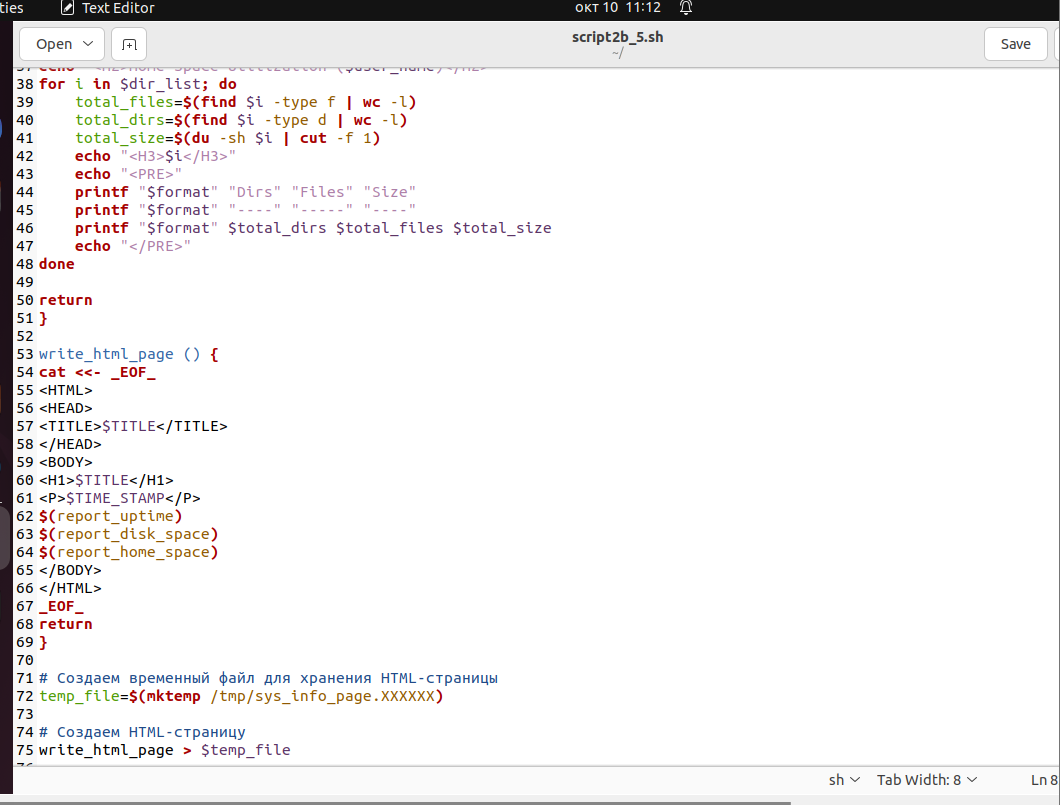
**Задание 3:**

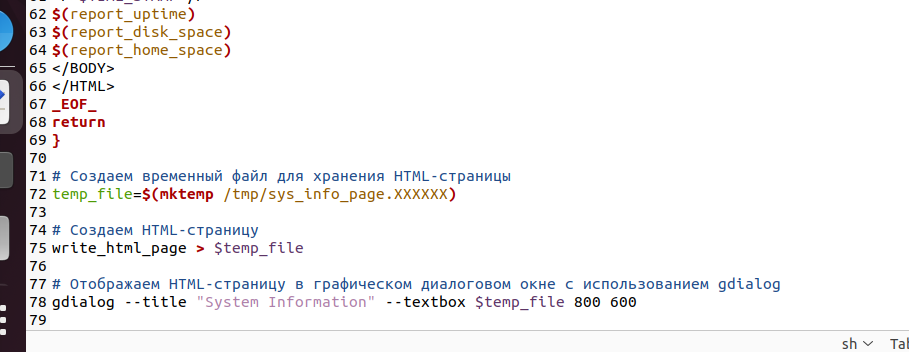
Для сценария выполненного в задании №2 создайте графическое диалоговое

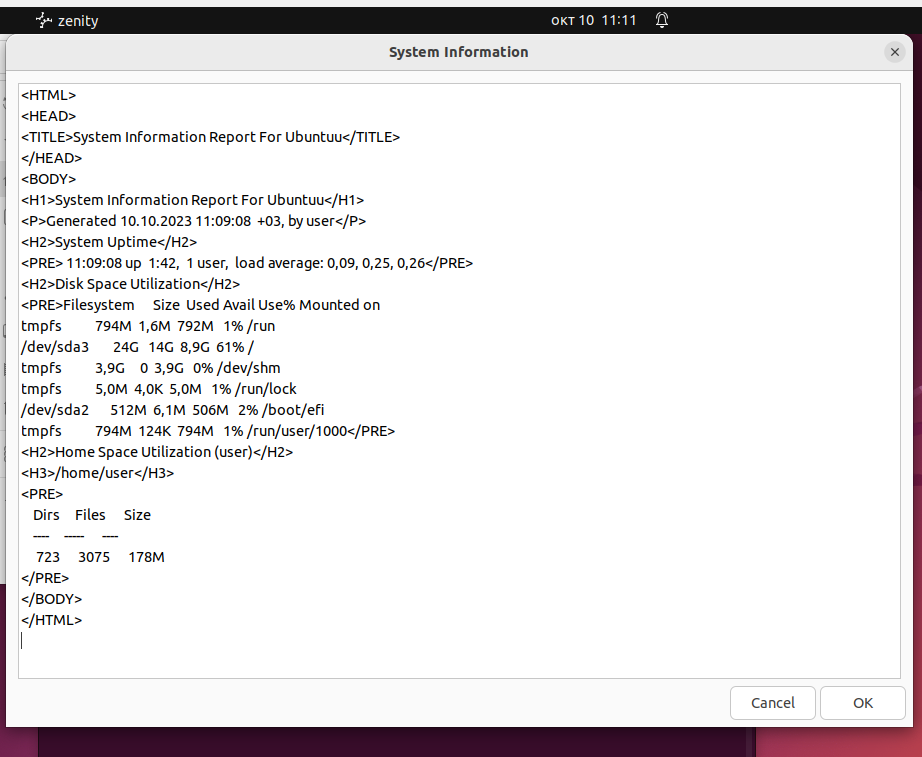
окно.











**Контрольные вопросы:**

**Возможности командной оболочки и скриптов Shell:**

**2) Как командная оболочка определяет тип значения переменных?**

Командная оболочка в Linux (например, Bash) определяет тип значения переменных на основе их содержимого и контекста использования. В Bash и других оболочках типы переменных могут быть следующими:

1. Строковые переменные (Strings): Переменные могут содержать строки текста. Bash не делает различия между одинарными и двойными кавычками, поэтому значения в переменных могут быть заключены в одинарные или двойные кавычки, или даже без кавычек.

Примеры:

```bash

my\_string="Hello, World!"

another\_string='This is a string'

``

2. Числовые переменные (Numbers): Bash поддерживает целые числа (integer) и числа с плавающей запятой (floating-point). Важно отметить, что в Bash отсутствует явное объявление типов переменных, поэтому они могут содержать числа разных типов.

Примеры:

```bash

integer\_var=42

float\_var=3.14

```

3. Массивы (Arrays): Bash поддерживает массивы, в которых можно хранить множество значений. Значения в массиве могут быть строками или числами.

Примеры:

```bash

my\_array=("apple" "banana" "cherry")

numbers=(1 2 3 4 5)

`

4. Булевы переменные (Booleans): В Bash нет встроенных булевых переменных, но обычно используют числовые значения 0 (ложь) и 1 (истина) для представления булевых значений.

Примеры:

```bash

is\_true=1

is\_false=0

```

5. Переменные с другими типами: Поскольку Bash является слабо типизированным языком, переменные могут содержать значения различных типов в зависимости от контекста. Это может быть удобно, но также требует осторожности при использовании переменных.

Примеры:

```bash

mixed\_var="Hello" # Строка

mixed\_var=42 # Целое число

mixed\_var=3.14 # Число с плавающей запятой

```

Тип переменной определяется автоматически во время выполнения в зависимости от того, какое значение присваивается переменной и какие операции выполняются с этим значением. Bash выполняет неявное приведение типов во многих случаях.

**Графический интерфейс диалога:**

**11) С помощью каких утилит реализуется средства примитивного**

**графического интерфейса в Linux-скриптах?**

В Linux-скриптах для создания примитивных графических интерфейсов можно использовать различные утилиты и библиотеки. Некоторые из наиболее распространенных утилит и библиотек для создания текстовых графических интерфейсов включают:

1. `dialog`: `dialog` является текстовым интерфейсом для создания диалоговых окон и меню в командной строке. Он позволяет создавать простые графические интерфейсы для взаимодействия с пользователем.

2. `whiptail`: `whiptail` - это улучшенная версия `dialog` с дополнительными возможностями для создания текстовых интерфейсов. Он предоставляет различные виджеты, такие как окна сообщений, списки, радиокнопки и флажки.

3. `ncurses`: `ncurses` - это библиотека для создания текстовых графических интерфейсов в терминале. Она предоставляет разработчикам более широкие возможности для создания сложных интерфейсов, таких как текстовые игры или консольные приложения.

4. `yad`: `yad` (Yet Another Dialog) - это инструмент, предоставляющий множество возможностей для создания графических интерфейсов в командной строке. Он позволяет создавать диалоговые окна, окна выбора файлов, и многое другое.

5. `zenity`: `zenity` - это инструмент для создания графических интерфейсов в скриптах с использованием GTK+ библиотеки. Он предоставляет диалоговые окна для ввода данных и взаимодействия с пользователем.

Эти утилиты позволяют создавать простые графические интерфейсы в командной строке, что может быть полезно для интерактивных скриптов и задач, где не требуется полноценный графический пользовательский интерфейс.