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 № 2.2**

по дисциплине:” Системное программирование”

на тему: “Командный язык и скрипты Shell”

Выполнил**:** студент группы 10701321

Черепковский М.В.

Принял**:** Давыденко Н.В.

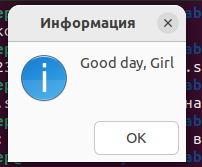
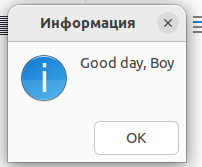
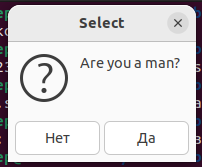
Минск 2023

# Лабораторная работа № 2.2. Командный язык и скрипты Shell

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

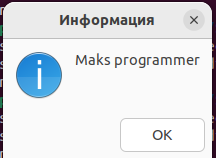
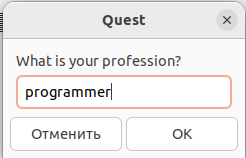
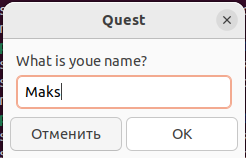
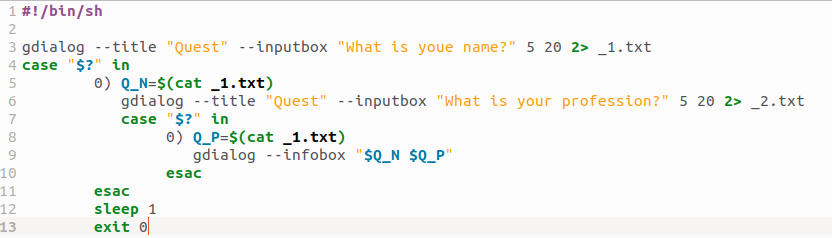
Задание 1

1. Сделать пример с двумя вложенными диалогами типа YesNo.



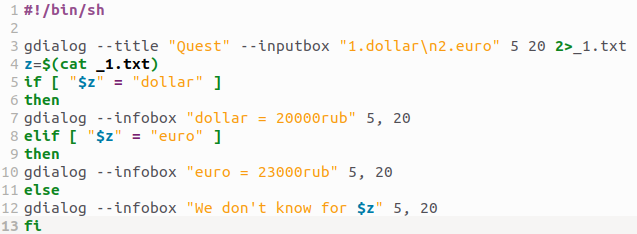
2. Сделать пример, запрашивающий сначала имя человека, а потом

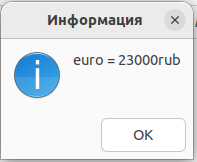
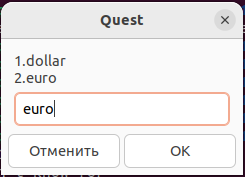
профессию. Вывести имя + профессию, прочитанные в диалоге.



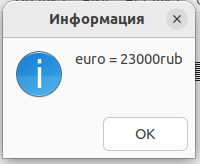
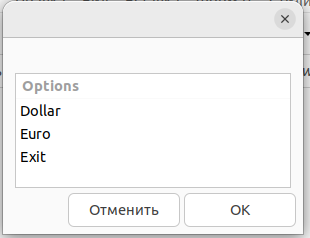
3. Вывести список с названиями валют. После выбора валюты система

должна вывести ее котировку



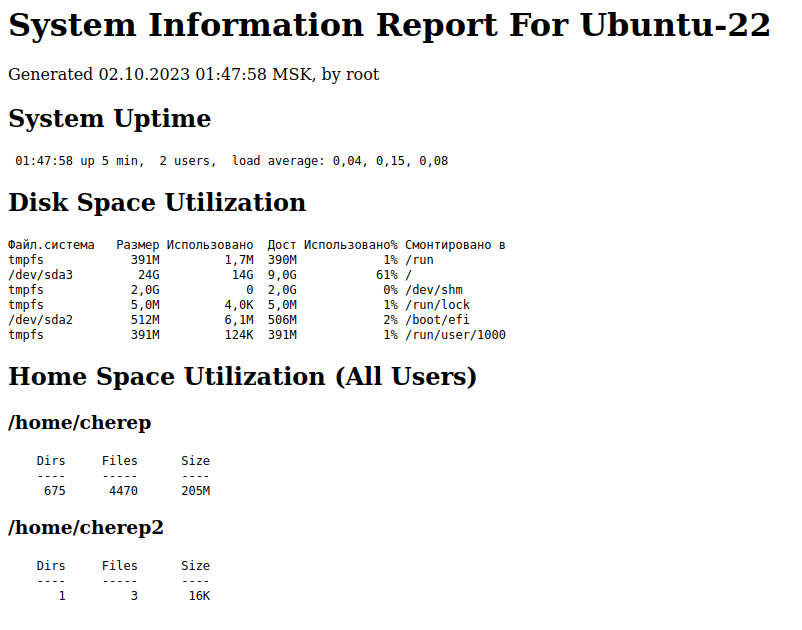


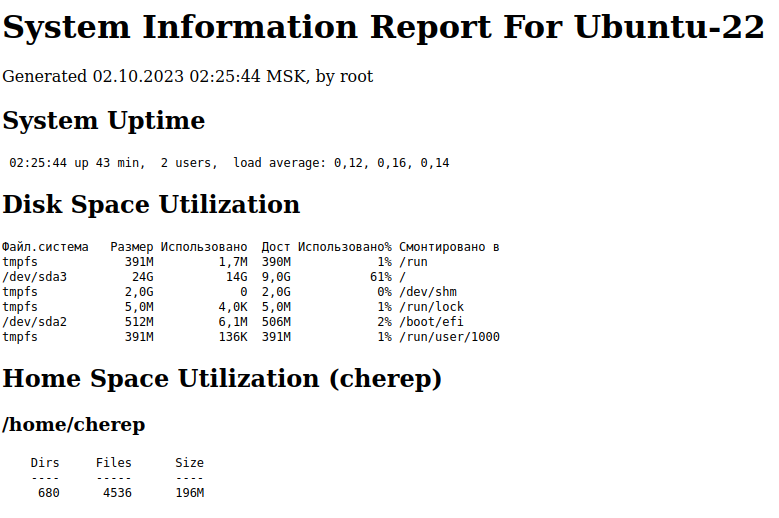
4,5 Измените предыдущую программу так, вместо списка валют  
предлагалось оконное меню валют. Чтобы программа работала в цикле.  
Для выхода из цикла нужно вместо названия валюты вводить exit.  
  
Измените предыдущую программу так, для выхода из программы в  
меню была кнопка закрытия программы «exit».



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.  



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

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Вывод*:* В ходе выполнения лабораторной работы изучили основные команды,   
для написания скриптов.  
  
Контрольные вопросы:  
 **1) Что такое *конфликт имен переменных* и как решается эта проблемма?**

Конфликт имен переменных в командной оболочке (shell) возникает, когда в одной и той же области видимости (например, внутри одного скрипта) используются две или более переменные с одинаковыми именами. Это может привести к непредсказуемому поведению программы или скрипта, так как shell будет затирать значение переменной одной другой, и это может вызвать ошибки в выполнении кода.

Чтобы решить эту проблему, можно использовать следующие методы:

1. \*\*Локальные переменные\*\*: В большинстве оболочек, таких как Bash, переменные могут быть объявлены как локальные. Это ограничивает область видимости переменной только для текущего блока кода или функции, избегая конфликта имен с переменными извне. Например:

```bash

# Локальная переменная x внутри функции

my\_function() {

local x=42

echo $x

}

```

2. \*\*Использование префиксов\*\*: Добавление префикса или суффикса к именам переменных может помочь избежать конфликта. Например, добавление имени скрипта или функции к переменной:

```bash

script\_name\_variable="Это переменная в скрипте"

function\_name\_variable="Это переменная в функции"

```

3. \*\*Использование массивов\*\*: В Bash можно использовать массивы для хранения переменных с разными именами:

```bash

my\_array[0]="Переменная1"

my\_array[1]="Переменная2"

```

4. \*\*Избегание глобальных переменных\*\*: Старайтесь избегать создания глобальных переменных, которые могут быть доступны из разных частей скрипта. Лучше ограничивайте область видимости переменных.

5. \*\*Использование функций\*\*: Функции в Bash создают свою собственную область видимости, поэтому переменные, объявленные внутри функции, не конфликтуют с глобальными переменными.

Решение конфликта имен переменных в shell зависит от конкретной ситуации и требований скрипта, но обычно описанные методы помогают избежать проблем с конфликтами имен.

**2) С помощью каких утилит реализуется средства примитивного графического интерфейса в Linux-скриптах?**

В Linux-скриптах для создания примитивного графического интерфейса можно использовать несколько утилит и библиотек. Некоторые из них включают:

1. \*\*Dialog\*\*: Dialog - это утилита командной строки, которая позволяет создавать диалоговые окна с различными элементами управления, такими как кнопки, текстовые поля, списки и т. д. Она предоставляет простой способ создания текстовых интерфейсов.

2. \*\*Zenity\*\*: Zenity - это еще одна утилита командной строки, которая предоставляет графические диалоговые окна, используя библиотеки GTK+. Она позволяет создавать окна с кнопками, текстовыми полями, списками и так далее.

3. \*\*Whiptail\*\*: Whiptail - это другая текстовая утилита для создания диалоговых окон в командной строке. Она может использоваться для создания текстовых интерфейсов с различными элементами управления.

4. \*\*Tkinter\*\*: Tkinter - это библиотека для Python, которая позволяет создавать графические интерфейсы на основе библиотеки Tk. Хотя это не утилита командной строки, вы можете использовать ее в скриптах Python для создания более сложных графических интерфейсов.

5. \*\*ncurses\*\*: ncurses - это библиотека, которая предоставляет функциональность для создания текстовых пользовательских интерфейсов в командной строке. Она позволяет создавать сложные текстовые интерфейсы с поддержкой цветов, окон и клавиш управления.

Эти утилиты и библиотеки могут быть полезными для создания простых графических интерфейсов в Linux-скриптах, особенно если вы хотите предоставить пользователям более удобные способы взаимодействия с вашими скриптами через текстовые окна и диалоги.