МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Кемеровский государственный университет»**

**Институт фундаментальных наук**

**Кафедра ЮНЕСКО по ИВТ**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**ПО ДИСЦИПЛИНЕ “Автоматизация процессов развертывания и инсталляции информационных систем”**

Направление 09.03.03 – Прикладная информатика в экономике

Студента 2 курса

Башкеева Степана Дмитриевича

Преподаватель:

И.Ю. Степанов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена:

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Кемерово 2022

Содержание

[Цель работы 3](#_Toc1257682675)

[Задачи 4](#_Toc349176931)

[Описание предметной области 4](#_Toc1102257847)

[Практическая часть 4](#_Toc1604501228)

[Заключение 6](#_Toc2103873596)

[Список литературы 7](#_Toc1997619830)

Цель работы

Изучить документацию и работу приложения для Android – эмулятора терминала Termux. А также с помощью данного терминала запустить бота для Telegram.

# Задачи

* Изучить документацию Termux.
* Запустить бота на Termux.

# Описание предметной области

Termux – это эмулятор терминала Android и приложение для среды Linux, которое работает напрямую без необходимости укоренения или настройки. Репозиторий Termux содержит более 500 программных пакетов, скомпилированных и готовых к установке для этого эмулятора терминала. Это, конечно не Debian/Linux, в репозитории которого около 30 000 пакетов.

# Практическая часть

1. Для начала работы необходимо создать репозиторий и перенести файлы с ботом на веб-сервисе для хостинга GitHub.

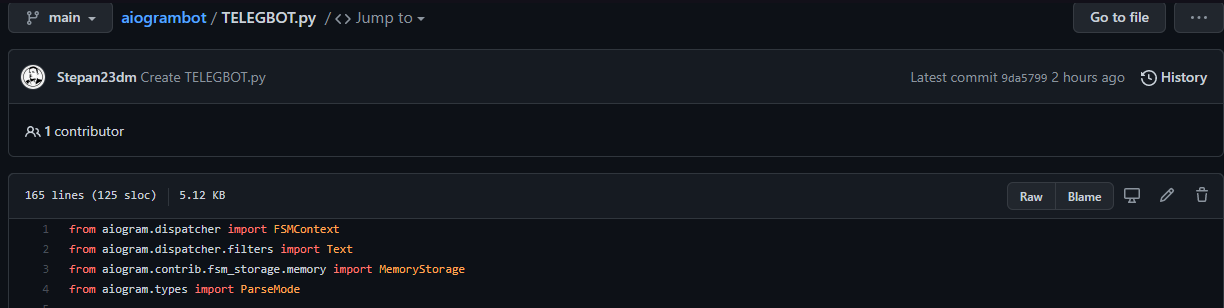


Рис. 1. Репозиторий aiogrambot.

1. Далее нужно установить приложение Termux на смартфон.

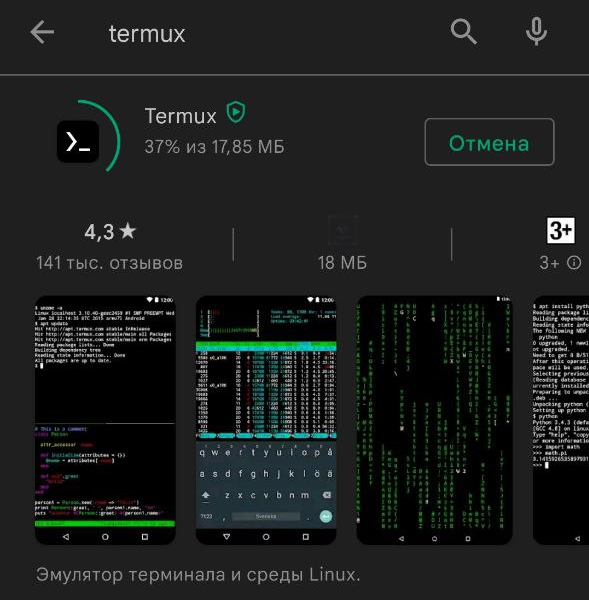


Рис. 2. Установка приложения Termux.

1. Открыв, нужно установить программы - Git и Python.

Чтобы установить git, необходимо в терминале вписать команду - pkg install git. (Рис. 3). [1]

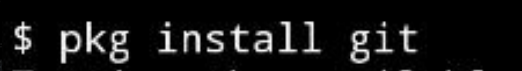


Рис. 3. Установка Git.

Для установки python, необходимо в терминале вписать команду - pkg install python. (Рис. 4). [1]

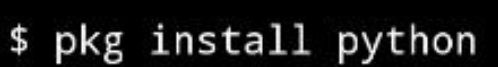


Рис. 4. Установка Python.

1. После установки программ вписываем команду - git clone [https://github.com/( Имя\_](https://github.com/( Имя)пользователя)/(Название\_репозитория).git.

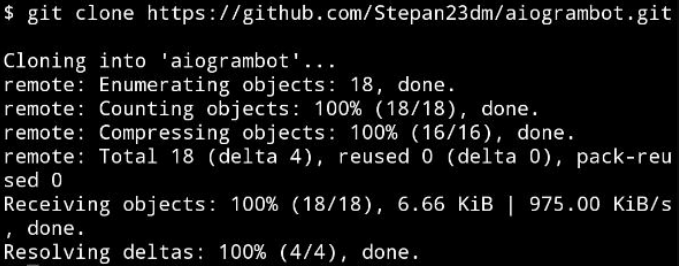


Рис. 5. Клонирование репозитория на смартфон.

1. Запуск бота.

После клонирования репозитория вписываем команду “ls”. В нашем случаи данная команда выведет наш клонированный репозиторий с ботом. Далее необходимо войти в репозиторий ‘aiogrambot’, для этого вписываем команду ‘cd aiogrambot’. После этого запускаем главный файл бота, пишем команду ‘python TELEGBOT.py'. (Рис. 6). [2]



Рис. 6. Запуск бота.

После запуска бота заходим в Telegram и проверяем бота. (Рис. 7).

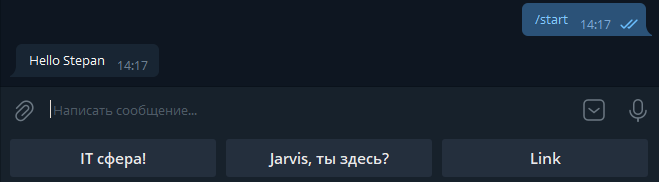


Рис. 7. Проверка работы бота.

# Заключение

В результате лабораторной работы была изучена документация Termux. На основе изученной документации был запущен бот для Telegram. Также был получен опыт работы с эмулятором терминала Termux.

# Список литературы

1. Termux шаг за шагом [Электрон. ресурс] / URL - <https://habr.com/ru/post/444950/>
2. Код доступа Termux [Электрон. ресурс] / URL - <https://habr.com/ru/post/652633/>
3. Termux [Электрон. ресурс] / URL - <https://play.google.com/store/ap>

# Приложение 1

Листинг 1. Код бота

from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters import Text  
from aiogram.contrib.fsm\_storage.memory import MemoryStorage  
from aiogram.types import ParseMode  
from aiogram.utils import executor  
  
import config  
import logging  
import buttons as btn  
import aiogram.utils.markdown as md  
import requests as req  
import aiohttp  
import asyncio  
  
from aiogram import Bot, Dispatcher, executor, types  
  
from aiogram.dispatcher.filters.state import State, StatesGroup  
  
from filters import IsAdminFilter  
#import infomations text  
from info import info\_lang, py\_link, js\_link, c\_link  
  
#log level  
logging.basicConfig(level=logging.INFO)  
  
#bot init  
bot = Bot(token=config.TOKEN)  
dp = Dispatcher(bot, storage=MemoryStorage())  
storage = MemoryStorage()  
  
#activate filters  
dp.filters\_factory.bind(IsAdminFilter)  
  
#class State  
class Mudialog(StatesGroup):  
 res\_user = State()  
  
#start bot message  
@dp.message\_handler(commands=['start'])  
async def command\_start(message: types.Message):  
 await bot.send\_message(message.from\_user.id, 'Hello {0.first\_name}'.format(message.from\_user), reply\_markup=btn.mainMenu)  
  
#remove keyboard  
@dp.message\_handler(commands=['close'])  
async def command\_start(message: types.Message):  
 await bot.send\_message(message.from\_user.id, 'Клавиатура закрыта!', reply\_markup=types.ReplyKeyboardRemove())  
  
  
class Form(StatesGroup):  
 name = State()  
 link = State()  
  
  
  
#buttons  
@dp.message\_handler()  
async def bot\_message(message: types.Message):  
 if message.text == 'IT сфера!':  
 await bot.send\_message(message.from\_user.id, 'IT сфера!', reply\_markup=btn.otherMenu)  
  
 elif message.text == 'Главное меню':  
 await bot.send\_message(message.from\_user.id, 'Главное меню', reply\_markup=btn.mainMenu)  
  
 elif message.text == 'Python':  
 await bot.send\_message(message.from\_user.id, py\_link)  
  
 elif message.text == 'JavaScript':  
 await bot.send\_message(message.from\_user.id, js\_link)  
  
 elif message.text == 'C#':  
 await bot.send\_message(message.from\_user.id, c\_link)  
  
 elif message.text == 'Информация':  
 await bot.send\_message(message.from\_user.id, info\_lang)  
  
 elif message.text == 'Jarvis, ты здесь?':  
 await Form.name.set()  
 await message.reply("К вашим услугам, сэр! Как я могу к вам обращатся?")  
  
 elif message.text == 'Link':  
 await Form.link.set()  
 await message.reply('Введите ссылку.')  
  
  
@dp.message\_handler(state='\*', commands='cancel')  
@dp.message\_handler(Text(equals='отмена', ignore\_case=True), state='\*')  
async def cancel\_handler(message: types.Message, state: FSMContext):  
 current\_state = await state.get\_state()  
 if current\_state is None:  
 return  
  
 await state.finish()  
 await message.reply('ОК')  
  
#input name  
@dp.message\_handler(state=Form.name)  
async def process\_name(message: types.Message, state: FSMContext):  
 async with state.proxy() as data:  
 data['name'] = message.text  
  
 async with state.proxy() as data:  
  
 await bot.send\_message(  
 message.chat.id,  
 md.text(  
 md.text('Приятно видеть вас за работай, ', md.bold(data['name'])),  
 sep='\n',  
 ),  
 parse\_mode=ParseMode.MARKDOWN,  
 )  
  
 await state.finish()  
  
#outlink  
@dp.message\_handler(state=Form.link)  
async def process\_link(message: types.Message, state: FSMContext):  
 async with state.proxy() as data:  
 try:  
 lk = req.get(message.text)  
 data['link'] = lk.json()['data']['email']  
 except:  
 data['link'] = 'Ошибка'  
  
 async with state.proxy() as data:  
  
 await bot.send\_message(  
 message.chat.id,  
 md.text(  
 md.text('Ответ сервера: ', md.bold(data['link'])),  
 sep='\n',  
 ),  
 parse\_mode=ParseMode.MARKDOWN,  
 )  
  
 await state.finish()  
#voice message  
@dp.message\_handler(content\_types=['voice'])  
async def message\_voice\_handler(message: types.Message):  
 await message.reply("Вы отправили голосовое сообщение.")  
  
#run long-polling  
if \_\_name\_\_ == "\_\_main\_\_":  
 executor.start\_polling(dp, skip\_updates=True)