**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**СЕРГИЕНКО ЛЕВ ЭДУАРДОВИЧ**

**ДАНИК КОХАН**

**АЛЕКСЕЙ РОМАНОВЕЦ**

**Знакомство с командной строкой Linux и bash-скриптами**

|  |  | **Преподаватель** |
| --- | --- | --- |
|  | **Андрушкевич И.Е.** |
|  | | |

**2024**

### **Тема: Знакомство с командной строкой Linux и bash-скриптами**

### **Цель работы:**

Ознакомиться с основными командами ОС Linux, научиться писать скрипты и запускать их с командной строки. Исследовать утилиту make.

### **Участники:**

1. **Go**: Даник Кохан
2. **Bash**: Леша Романовец
3. **Python**: Лев Сергиенко

### **Введение:**

В ходе лабораторной работы были выполнены задачи с использованием трех разных языков программированияДАНИК: Go, Bash и Python. Каждый язык был выбран по определенным причинам, которые связаны с их характеристиками и применимостью для решения задач.

### **Описание используемых языков программирования:**

#### **Go (выполнил Даник Кохан):**

Go был выбран за его эффективность и высокую производительность при работе с многозадачными операциями. Он обладает строгой статической типизацией, что позволяет избежать многих ошибок на этапе компиляции. Также, Go имеет встроенные библиотеки для работы с операционной системой и файловой системой, что упрощает взаимодействие с командной строкой Linux.

* **Применимость**: Go хорошо подходит для создания более сложных и масштабируемых приложений, особенно если требуется оптимизация производительности и использования системных ресурсов.

#### **Bash (выполнил Леша):**

Bash – это стандартная оболочка в большинстве дистрибутивов Linux. Он был выбран благодаря своей простоте и непосредственной связи с командной строкой Linux. Bash-скрипты позволяют автоматизировать задачи и легко работать с командами и файлами, что делает его естественным выбором для работы с командной строкой и системными процессами.

* **Применимость**: Bash является естественным выбором для задач, связанных с автоматизацией и скриптингом в Linux. Он прост в использовании и требует минимальных ресурсов.

#### **Python (выполнил Лев Сергиенко):**

Python – это высокоуровневый язык программирования, известный своей простотой и читаемостью. Он был выбран благодаря своей гибкости и поддержке многочисленных библиотек, что облегчает работу с файлами, строками, и взаимодействие с системными вызовами.

* **Применимость**: Python идеален для быстрого прототипирования и решения задач, связанных с обработкой данных и интеграцией различных систем.

### **Описание задач и их решение**

#### **Задача 1**

**Описание задачи и как она была выполнена**

Нужно вычислить **выигрышное число** для каждого участника на основе его фамилии и заданного веса. Затем отсортировать участников по выигрышным числам (если числа одинаковы, то по фамилии в алфавитном порядке). В итоге нужно вернуть фамилию участника с заданным рангом.

**Алгоритм:**

1. **Парсинг строки** фамилий и их разделение по запятой.
2. **Создание словаря алфавита**: каждому символу присваивается его ранг (A и a = 1, B и b = 2 и т.д.).
3. Для каждой фамилии:
   * Вычисляется сумма рангов букв фамилии.
   * К сумме добавляется длина фамилии.
   * Результат умножается на вес.
4. **Сортировка фамилий** по выигрышным числам в порядке убывания. Если числа совпадают, сортировка по алфавиту.
5. **Проверка на ошибки** (если строка пуста или ранг n больше количества участников).
6. Возврат фамилии с рангом n.

##### **Код на GoLang**

package main

import (

"fmt"

"sort"

"strings"

"unicode"

)

type pair struct {

val int

name string

}

func main() {

st := "COLIN,AMANDBA,AMANDAB,CAROL,PauL,JOSEPH"

we := []int{1, 4, 4, 5, 2, 1}

n := 4

if len(st) == 0 {

fmt.Println("Нет участников")

return

}

if len(st) < n {

fmt.Println("Недотстаночно участников ")

}

surnames := strings.Split(st, ",")

alphabetMap := make(map[byte]int)

for i := 'a'; i <= 'z'; i++ {

alphabetMap[byte(i)] = int(i - 'a' + 1)

}

weigths := make([]pair, len(surnames))

for i := 0; i < len(surnames); i++ {

sum := 0

for j := 0; j < len(surnames[i]); j++ {

lowerByte := byte(unicode.ToLower(rune(surnames[i][j])))

sum += alphabetMap[lowerByte]

}

sum \*= we[i]

sum += len(surnames[i])

weigths[i].val = sum

weigths[i].name = surnames[i]

}

sort.Slice(weigths, func(i, j int) bool {

return weigths[i].val > weigths[j].val

})

fmt.Println(weigths[n-1])

}

##### **Код на Bash**

#!/bin/bash

# Input data

names="COLIN,AMANDBA,AMANDAB,CAROL,PauL,JOSEPH"

weights="1,4,4,5,2,1"

n=4

# Convert input strings to arrays

IFS=',' read -r -a names\_array <<< "$names"

IFS=',' read -r -a weights\_array <<< "$weights"

echo "Names array: ${names\_array[@]}"

echo "Weights array: ${weights\_array[@]}"

# Check if there are no participants

if [ ${#names\_array[@]} -eq 0 ]; then

echo "No participants"

exit 0

fi

# Check if there are not enough participants

if [ $n -gt ${#names\_array[@]} ]; then

echo "Not enough participants"

exit 0

fi

# Initialize results array

results=()

# Calculate the winning number for each participant

for (( i=0; i<${#names\_array[@]}; i++ )); do

name=${names\_array[$i]}

weight=${weights\_array[$i]}

echo "Processing name: $name, weight: $weight"

# Calculate the sum of alphabetical positions

sum=0

for (( j=0; j<${#name}; j++ )); do

char=${name:$j:1}

ascii=$(printf '%d' "'$char")

if [[ $char =~ [A-Z] ]]; then

position=$((ascii - 64))

elif [[ $char =~ [a-z] ]]; then

position=$((ascii - 96))

else

position=0

fi

sum=$((sum + position))

done

som=$((sum + ${#name}))

echo "Sum of positions: $som"

# Calculate the winning number

winning\_number=$((som \* weight))

results+=("$winning\_number $name")

done

echo "Results before sorting: ${results[@]}"

# Sort the results by winning number (descending) and then by name (ascending)

sorted\_results=$(printf '%s\n' "${results[@]}" | sort -rn -k1,1 -k2,2)

# Print sorted results on new lines

echo "Sorted results:"

while IFS= read -r line; do

echo "$line"

done <<< "$sorted\_results"

# Print the n-th participant

echo "$sorted\_results" | awk -v n=$n 'NR==n{print $2}'

##### **Код на Python**

def calc\_som(name):

som = 0

name = name.upper()

for ch in name:

if "A" <= ch <= "Z":

som += ord(ch) - ord("A") + 1

return som + len(name)

def prize\_draw(st, we, n):

if not st:

return "Нет участников"

names = st.split(",")

if n > len(names):

return "Недостаточно участников"

participants = []

for i, name in enumerate(names):

som = calc\_som(name)

win\_number = som \* we[i]

participants.append((name, win\_number))

participants.sort(key=lambda x: (-x[1], x[0]))

return participants[n - 1][0]

names = "COLIN,AMANDBA,AMANDAB,CAROL,PauL,JOSEPH"

weights = [1, 4, 4, 5, 2, 1]

rank = 4

result = prize\_draw(names, weights, rank)

print(f"Результат: {result}")

#### **Задача 2**

**Описание задачи и как она была выполнена**

Взять целое число n и цифру d. Нужно возвести в квадрат все числа от 0 до n, и подсчитать, сколько раз цифра d встречается в записи всех этих квадратов.

**Алгоритм**

1. В цикле от 0 до n возводим каждое число в квадрат.
2. Преобразуем квадрат в строку.
3. Считаем, сколько раз цифра d встречается в строковом представлении квадрата.
4. Возвращаем итоговый подсчет.

##### **Код на Golang**

package main

import (

"fmt"

"strconv"

)

func nbDig(n, d int) int {

count := 0

digit := strconv.Itoa(d)

for k := 0; k <= n; k++ {

square := k \* k

squareStr := strconv.Itoa(square)

for \_, ch := range squareStr {

if string(ch) == digit {

count++

}

}

}

return count

}

func main() {

n := 10

d := 1

result := nbDig(n, d)

fmt.Println("Result:", result)

}

##### **Код на Bash**

#!/bin/bash

# Function to count the occurrences of digit d in the squares of numbers from 0 to n

nb\_dig() {

local n=$1

local d=$2

local count=0

# Iterate through all numbers from 0 to n

for (( k=0; k<=n; k++ )); do

square=$((k \* k))

# Convert the square to a string

square\_str="$square"

# Count occurrences of digit d in the square

count=$((count + $(echo "$square\_str" | grep -o "$d" | wc -l)))

done

echo "$count"

}

# Example usage

n=25

d=1

result=$(nb\_dig $n $d)

echo "The digit $d appears $result times in the squares of numbers from 0 to $n."

##### **Код на Python**

def nb\_dig(n, d):

count = 0

for k in range(n + 1):

*# Возводим число k в квадрат*

square = k\*\*2

*# Преобразуем квадрат в строку и подсчитываем, сколько раз цифра d встречается в строке*

count += str(square).count(str(d))

return count

print(nb\_dig(25, 1))

#### **Makefile**

**run2\_py**:

python3 task2.py

**run2\_c**:

gcc task2.c -o task2

./task2

**run2\_sh**:

./task2.sh

**run1\_py**:

python3 task1.py

**run1\_c**:

gcc task1.c -o task1

./task1

**run1\_sh**:

./task1.sh

### **Сравнение языков программирования: Go, Bash и Python**

#### **Go и Bash:**

**Go** и **Bash** сильно различаются по своей архитектуре, применимости и подходу к программированию.

1. **Go (Golang)** – это компилируемый язык, разработанный для высокопроизводительных системных и сетевых приложений. Go предназначен для обеспечения максимальной производительности и эффективности. Его основное преимущество – это поддержка параллелизма через «горутины» (lightweight threads), которые позволяют запускать тысячи задач одновременно с минимальными накладными расходами.
   * **Сильные стороны Go**:
     + **Производительность**: Go компилируется в машинный код, что делает его существенно быстрее интерпретируемых языков. Его производительность особенно важна при работе с большими объемами данных, вычислениями или сетью.
     + **Параллелизм**: Go предлагает встроенные средства для конкурентности и многозадачности, что упрощает разработку высоконагруженных приложений.
     + **Система типов**: Строгая типизация позволяет избежать множества ошибок на этапе компиляции.
2. **Bash** – это скриптовый язык командной оболочки, который используется в Unix-подобных системах для автоматизации выполнения команд. Он прост в использовании и отлично подходит для взаимодействия с командной строкой, работы с файлами и системными процессами.
   * **Сильные стороны Bash**:
     + **Простота написания и отладки**: Bash идеален для написания небольших сценариев, автоматизирующих рутинные задачи (например, управление файлами, запуск программ).
     + **Прямая интеграция с командной строкой**: Bash работает непосредственно с командной оболочкой, что делает его очень удобным для системного администрирования и написания утилит.

**Сравнение:**

* **Производительность**: Go значительно быстрее Bash, особенно на больших данных или в задачах, требующих высокой степени параллелизма. Bash, будучи интерпретируемым, подходит только для простых и менее ресурсоемких задач.
* **Удобство использования**: Bash проще для создания простых скриптов для командной строки и работы с файлами, тогда как Go требует большего количества кода, но предлагает больше возможностей для сложных и высокопроизводительных задач.
* **Масштабируемость**: Go подходит для создания масштабируемых решений и серверных приложений, тогда как Bash теряет эффективность при увеличении сложности задач.

#### **Python и Bash:**

**Python** и **Bash** часто используются для автоматизации, но их возможности сильно различаются.

1. **Python** – это высокоуровневый интерпретируемый язык общего назначения с огромной экосистемой библиотек, которые упрощают работу с данными, сетью и файловыми операциями.
   * **Сильные стороны Python**:
     + **Читаемость кода**: Python известен своей лаконичностью и удобочитаемостью, что упрощает сопровождение и разработку.
     + **Богатая стандартная библиотека**: Python предлагает встроенные инструменты для работы с файлами, текстом, сетью, а также мощные модули для обработки данных и выполнения сложных математических операций.
     + **Кросс-платформенность**: Python легко переносится между операционными системами и используется для автоматизации, работы с веб-приложениями, машинного обучения и анализа данных.
2. **Bash** – как уже упоминалось, это скриптовый язык для работы с командной строкой, который ориентирован на выполнение команд и управления файлами и процессами.
   * **Сильные стороны Bash**:
     + **Прямой доступ к системным вызовам**: Bash позволяет напрямую работать с командами операционной системы, что делает его особенно удобным для управления процессами и файловой системой.
     + **Автоматизация задач**: Bash быстро пишет и исполняет простые сценарии автоматизации.

**Сравнение:**

* **Читаемость и расширяемость**: Python значительно превосходит Bash в плане читаемости кода и гибкости. Bash хорош для простых скриптов, но при увеличении сложности задача становится неуправляемой из-за ограниченной функциональности.
* **Автоматизация и функциональность**: Python подходит для более сложных задач автоматизации, обработки данных и работы с сетевыми операциями. Bash, напротив, ограничен взаимодействием с командной строкой и не имеет встроенных инструментов для работы с данными.
* **Производительность**: Несмотря на то, что Python интерпретируемый язык, он выигрывает по сравнению с Bash благодаря лучшей оптимизации, поддержке многопоточности и сторонним библиотекам, таким как NumPy для вычислений.

#### **Python и Go:**

Оба языка являются популярными, но их применения сильно отличаются.

1. **Go** – это язык для создания высоконагруженных системных и сетевых приложений.
   * **Сильные стороны Go**:
     + **Высокая производительность**: Go компилируется в машинный код, что делает его быстрее интерпретируемых языков, таких как Python.
     + **Многозадачность**: Встроенные механизмы конкурентности и параллелизма в Go (горутины) делают его идеальным для разработки высоконагруженных серверных приложений.
     + **Безопасность типов**: Строгая типизация уменьшает количество ошибок на этапе компиляции.
2. **Python** – это язык для быстрого прототипирования и решения задач с минимальными затратами на разработку.
   * **Сильные стороны Python**:
     + **Гибкость и быстрота разработки**: Python позволяет писать код быстрее благодаря своей читаемости и простоте.
     + **Обширная экосистема библиотек**: Python имеет множество готовых решений для обработки данных, машинного обучения, анализа и автоматизации.

**Сравнение:**

* **Производительность**: Go значительно быстрее Python, особенно в задачах, требующих высокой производительности (обработка сетевых запросов, вычисления). Python, будучи интерпретируемым, имеет ограничения по скорости.
* **Удобство разработки**: Python проще в использовании и позволяет быстрее разрабатывать и отлаживать код. Go требует больше времени на компиляцию и отладку, но предоставляет лучшую производительность и надежность.
* **Многозадачность и параллелизм**: Go был специально разработан для работы с параллельными вычислениями, тогда как Python полагается на библиотеки для реализации многопоточности, что делает его менее эффективным для таких задач.

### **Заключение**

Каждый из языков программирования Go, Bash и Python имеет свои сильные и слабые стороны, которые определяют их применение в зависимости от задач.

* **Bash** – лучший выбор для простых системных скриптов и автоматизации в командной строке.
* **Python** – идеален для быстрой разработки и решения сложных задач обработки данных, автоматизации и интеграции различных систем.
* **Go** – превосходит по производительности и масштабируемости, особенно для задач с высокой нагрузкой и параллелизмом.