Michał Krystecki 342906 Lab 9 – Pojęcie Skryptu

Zad.1

Kod źródłowy :

```
#!/bin/bash

echo "Podaj sciezke: "

read p

echo "podaj rozszerzenie: "

read e

mkdir -p "$HOME/backup"

find "$HOME/$p" -name "*.$e" -exec cp {} "$HOME/backup" \;

echo "Skopiowano"
```

Użytkownik najpierw wpisuje ścieżkę z katalogu domowego do katalogu który chcemy przeszukać, następnie podaje rozszerzenie (bez kropki). Zamiast znaku „~” użyłem $HOME ponieważ program z użyciem tyldy nie działał prawidłowo, nie znajdywał ścieżki do katalogu. Po wczytaniu danych program utworzy katalog backup (jeżeli takowy nie istnieje, jeżeli istnieje to komenda nic nie zrobi, skrypt będzie kontynuował działanie) i po przefiltrowaniu zadanego folderu przekopiuje odpowiednie pliki do katalogu backup.

```
┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ls ~/backup

┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ls ~/sprawozdania
 Sprawozdanie1.docx   Sprawozdanie2.odt     'Sprawozdanie 3.odt'   Sprawozdanie_lab_8.docx   Sprawozdanie_lab_8.pdf
 Sprawozdanie1.odt    Sprawozdanie2pdf.pdf   Sprawozdanie3.pdf     Sprawozdanie_lab_8.odt

┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ./skrypt1.bash
Podaj sciezke:
sprawozdania
podaj rozszerzenie:
pdf
Skopiowano

┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ls ~/backup
Sprawozdanie2pdf.pdf  Sprawozdanie3.pdf  Sprawozdanie_lab_8.pdf

┌──(kali㉿kali)-[~/repo/skrypty]
└─$
```

Zad.2

Kod źródłowy:

```
#!/bin/bash
echo "Naciśnij 'q', aby zakończyć działanie skryptu."
sleep 1
while true; do
  clear
  echo "=== MONITOR SYSTEMU ==="
  echo "Nazwa hosta: $(hostname)"
  echo "Wersja jądra: $(uname -r)"
  echo "Czas działania: $(uptime -p)"
  echo -n "Wykorzystanie CPU: "
  mpstat | tail -1 | awk '{print 100-$12"%"}'
  echo "Użycie dysku (Total):"
  df -h --total | tail -1
  echo "---------------------------------"
  echo "5 najbardziej zasobożernych procesów:"
  ps -eo pid,ppid,cmd,%cpu --sort=-%cpu | head -6
  read -t 1 -N 1 input
  if [[ "$input" == "q" ]]; then
    echo -e "\nZakończono działanie."
    break
  fi
done
```

Skrypt wyświetla zadane w poleceniu zmienne i czeka przez sekundę na jedno znakowy sygnał do zakończenia pracy , tu literę q. Warto zwrócić uwagę na komendy wyświetlające

użycie CPU, użycie dysku i najbardziej zasobożerne procesy. Użycie CPU: Mpstat generuje statystyki użycia CPU, tail -1 ogranicza je tylko do ostatniego czyli podsumowania, awk dzieli dane wejściowe na kolumny a potem wyświetla różnice 100 i dwunastej kolumny czyli %idle procesora. Użycie dysku: pobiera informacje o zapełnieniu dysku, ustawia ich format na łatwo czytelny dla człowieka (np. w GB), dodaje linijkę podsumowującą i ogranicza wyświetlany tekst tylko do niej.Najbardziej zasobożerne procesy: pozyskuje zadane informacje o procesach sortuje je malejąco po użyciu CPU (%cpu) i ogranicza wyświetlany wynik do 6 pierwszych rzędów co daje nam rząd z nagłówkami kolumn i 5 procesów .

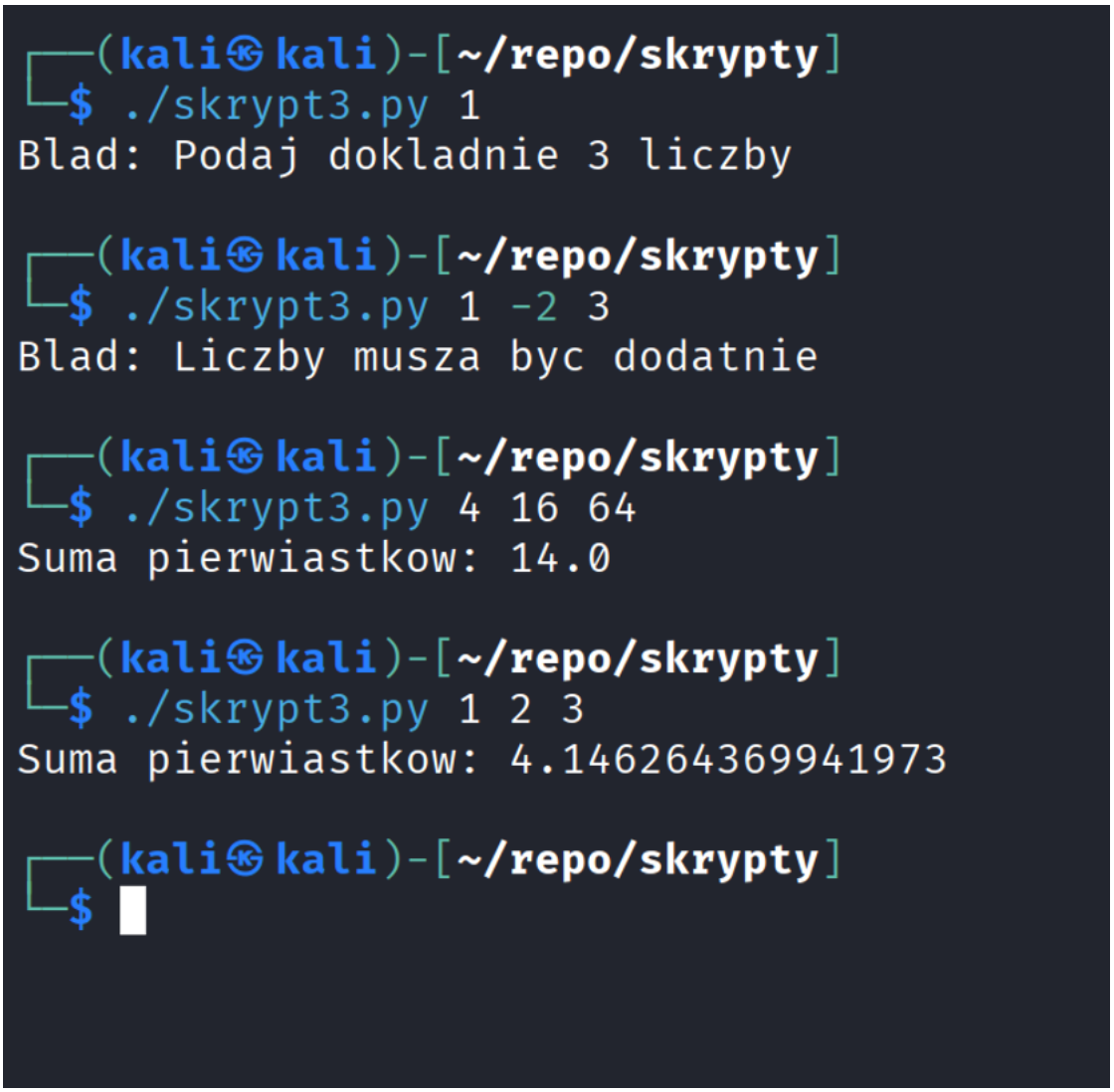
```
Nazwa hosta: kali
Wersja jądra: 6.16.8+kali-arm64
Czas działania: up 1 minute
Wykorzystanie CPU: 100%
Użycie dysku (Total):
total            69G   28G   38G  43% -
---------------------------------
5 najbardziej zasobożernych procesów:
    PID    PPID CMD                         %CPU
   1880    1820 /usr/share/code/code --type 17.1
   1915    1813 /proc/self/exe --type=utili  9.4
   1912    1813 /proc/self/exe --type=utili  6.8
   1853    1817 /usr/share/code/code --type  6.2
   1012    1002 /usr/lib/xorg/Xorg :0 -seat  3.1
q
Zakończono działanie.
```

Zad.3

Kod źródłowy:

```
#!/usr/bin/env python3

import sys

import math

if len(sys.argv) != 4:

   print("Blad: Podaj dokladnie 3 liczby")

   sys.exit()

a = float(sys.argv[1])

b = float(sys.argv[2])

c = float(sys.argv[3])

if a > 0 and b > 0 and c > 0:

   suma = math.sqrt(a) + math.sqrt(b) + math.sqrt(c)

   print("Suma pierwiastkow:", suma)

else:

   print("Blad: Liczby musza byc dodatnie")
```

Program importuje biblioteki sys i math, następnie sprawdza czy zostały podane 4 parametry (pierwszy z nich to nazwa uruchamianego skryptu jeżeli uruchomimy go wpisując np. ./skrypt3.bash 1 2 3 ; a 3 kolejne to argumenty do użycia przez funkcję). Następnie ustawia wartości zmiennych a,b,c jako wpisane przez użytkownika argumenty, sprawdza czy są większe od zera, jeśli tak to wyświetla sumę ich pierwiastków , a w przeciwnym razie informuje o błędzie.


┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ./skrypt3.py 1
Blad: Podaj dokladnie 3 liczby
┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ./skrypt3.py 1 -2 3
Blad: Liczby musza byc dodatnie
┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ./skrypt3.py 4 16 64
Suma pierwiastkow: 14.0
┌──(kali㉿kali)-[~/repo/skrypty]
└─$ ./skrypt3.py 1 2 3
Suma pierwiastkow: 4.146264369941973
┌──(kali㉿kali)-[~/repo/skrypty]
└─$

Zad.4

Kod źródłowy:

```
#!/bin/bash
if [ $# -ne 3 ]
then
   echo "Podaj dokladnie 3 argumenty"
   exit 1
fi
if [ $1 -gt 0 ] && [ $2 -gt 0 ] && [ $3 -gt 0 ]
then
   echo "Argumenty poprawne. Uruchamiam skrypt Python..."
   arg1=$(expr $1 + 1)
   arg2=$(expr $2 + 1)
   arg3=$(expr $3 + 1)
   echo "Wywolanie 1 (argumenty + 1; $arg1 $arg2 $arg3):"
   python3 skrypt3.py $arg1 $arg2 $arg3
   arg1=$(expr $1 + 2)
   arg2=$(expr $2 + 2)
   arg3=$(expr $3 + 2)
   echo "Wywolanie 2 (argumenty + 2); $arg1 $arg2 $arg3):"
   python3 skrypt3.py $arg1 $arg2 $arg3
   arg1=$(expr $1 + 3)
   arg2=$(expr $2 + 3)
   arg3=$(expr $3 + 3)
   echo "Wywolanie 3 (argumenty + 3; $arg1 $arg2 $arg3):"
   python3 skrypt3.py $arg1 $arg2 $arg3
```

```
else
    echo "Blad: Wszystkie argumenty musza byc liczbami dodatnimi"
fi
```

Skrypt sprawdza czy podano 3 argumenty, następnie sprawdza czy są większe od 0. Jeżeli tak to wywołuje skrypt z poprzedniego zadania odpowiednio dla argumentów o 1,2,3 większe. Jeżeli podane argumenty nie są większe od zera lub jest ich za mało skrypt zwraca informacje o błędzie.