Temat: Automatyzacja pracy – skrypty –zmienne, parametry, instrukcje warunkowe w Linux.

Cel ogólny lekcji opartej na poniższych instrukcjach to zapoznanie uczniów z podstawami automatyzacji pracy za pomocą skryptów powłoki (shellowych) w systemie Linux.

Cele szczegółowe lekcji:

1. Wyjaśnienie pojęcia skryptu powłoki i prezentacja sposobów uruchamiania skryptów powłoki.
2. Przeprowadzenie ćwiczeń związanych z tworzeniem zmiennych i przypisywaniem im wartości w skryptach.
3. Zapoznanie uczniów z różnicami między użyciem pojedynczych i podwójnych cudzysłowów w wyświetlaniu tekstu na ekranie za pomocą funkcji echo w skryptach powłoki.
4. Praktyczne ćwiczenie, w którym uczniowie samodzielnie tworzą skrypt i wykorzystują w nim wiedzę zdobytą na lekcji.
5. Zapoznanie uczniów z instrukcją CASE w języku bash
6. Nauczenie uczniów jak korzystać z instrukcji CASE w języku bash
7. Zrozumienie przez uczniów składni instrukcji CASE i sposobu jej działania
8. Zapoznanie uczniów z instrukcją warunkową IF w języku bash
9. Nauczenie uczniów jak korzystać z instrukcji warunkowej IF w języku bash
10. Zrozumienie przez uczniów składni instrukcji warunkowej IF i sposobu jej działania
11. Nauczenie uczniów jak pisać warunki w instrukcji warunkowej IF w języku bash
12. Zapewnienie uczniom praktyki w pisaniu skryptów w języku bash przy użyciu instrukcji warunkowych i instrukcji CASE.

Wprowadzenie:
Skrypt powłoki (shellowy) to rozbudowana wersja listy pewnych poleceń, które mają zostać wykonane przez system. Wykonaj i przetestuj działanie wszystkich skryptów poniżej. W zszycie zapisz nowe informacje.

1. Pierwszy skrypt.

Utwórz plik tekstowy o nazwie skrypt

touch skrypt

Skrypt (działający w BASH-u) powinien się zaczynać definicji powłoki, która będzie odpowiedzialna za wykonanie ciągu instrukcji umieszczonych wewnątrz skryptu za pomocą strumienia tekstu wyświetlanego na ekran:

```
echo '#!/bin/bash' > skrypt
```

i za pomocą dowolnego edytora dopisz w kolejnej linii do pliku skrypt:

echo ”Moj pierwszy skrypt” #wyświetlenie tekstu

Użyj dwie metody, aby uruchomić skrypt:

a) wyświetl ścieżkę plików wykonywalnych dla $SHELL

which $SHELL

skrypt uruchomiamy akceptując polecenie:

bash skrypt

Wyświetlił na ekranie monitora komunikatu o treści: Moj pierwszy skrypt

b) należy mu nadać atrybut wykonywalności za pomocą polecenia:

chmod +x skrypt

Jeśli katalog bieżący w którym znajduje się skrypt nie jest dopisany do zmiennej PATH, to skrypt uruchomiamy akceptując polecenie:

./skrypt

Wyświetlił na ekranie monitora komunikatu o treści: Moj pierwszy skrypt

Nie widać tekstu zaczynającego się od # to komentarz kodu.

Zgłoszenie1

2. Zmienne.

W każdym języku programowania potrzebne są zmienne, aby przy ich pomocy wykonać pewne operacje, definiuje się je w następujący sposób:

ZMIENNA=witam # jeżeli w wartości zmiennej nie ma spacji nie trzeba używać znaku cudzysłowie

lub

ZMIENNA=12345 # jeżeli przypisujemy liczby lub cyfry nie używamy znaku cudzysłowie

lub

ZMIENNA=”witam was” # jeżeli w wartości zmiennej znajduje się znak spacji, to wartość tą trzeba umieścić w cudzysłowie w przeciwnym wypadku będzie to zinterpretowane jako błędny zapis.

Podczas deklarowania zmiennych i przypisywania im pewnych wartości należy zwrócić uwagę na jedną bardzo istotną rzecz, po obu stronach znaku = (który służy do przypisania wartości do zmiennej) nie mogą znajdować się spacje, ponieważ spowoduje to powstanie błędu.

ZMIENNA = witam #błędny zapis !!!

Nazwy zmiennych pisze się dużymi literami (oczywiście jak ktoś nazwę zmiennej napisze małymi literami to się nic nie stanie i skrypt będzie działał poprawnie, ale przy programowaniu skryptów przyjęło się, że nazwy zmiennych są pisane dużymi literami). BASH rozróżnia duże i małe litery, Odwoływanie się do zmiennej odbywa się przez użycie dokładnie takiej samej nazwy zmiennej przy pomocy jakiej ta zmienna została zadeklarowana

1. Odwołanie się do wartości zadeklarowanej wcześniej zmiennej odbywa się w sposób:

ZMIENNA=witaj #deklaracja zmiennej

$ZMIENNA #odwołanie się do zmiennej w sposób prawidłowy

$zmienna #odwołanie się do zmiennej w sposób nieprawidłowy

W Bashu podczas wyświetlania teksu na ekranie monitora można się spotkać z dwoma konstrukcjami:

- echo ”Mój pierwszy skrypt”
- echo ‘Mój pierwszy skrypt’

w obu, przedstawionych powyżej przypadkach efekt będzie taki sam – na ekranie monitora pokaże się napis:

Mój pierwszy skrypt

2. różnicę w tych konstrukcjach sprawdź na przykładzie skryptu:

#!/bin/bash

ZMIENNA=”Mój pierwszy skrypt”

echo ”$ZMIENNA”

echo ‘$ZMIENNA’

Po uruchomieniu skryptu na ekranie monitora ukaże się następujący rezultat:

Mój pierwszy skrypt

$ZMIENNA

Wewnątrz cudzysłowie nazwy zmiennych, które są poprzedzone znakiem $ są zastępowane przez przypisaną im wartość, natomiast wewnątrz apostrofów nie.

Niekiedy zachodzi potrzeba, aby nazwy zmiennych ująć w cudzysłowy. Jest to istotne, gdy wartość zmiennej zawiera spacje lub jest pustym ciągiem.

3. Prześledź działanie następującego skryptu:

```
#!/bin/bash
ZMIENNA=""
if [ -n $ZMIENNA ]; then # -n sprawdza czy argument nie jest pusty
        echo " Zmienna o nazwie ZMIENNA nie jest pusta";
else
        echo " Zmienna o nazwie ZMIENNA jest pusta";
fi
```

Uruchom powyższy skrypt, na ekranie monitora pojawi się napis:

Zmienna o nazwie ZMIENNA nie jest pusta

Dzieje się tak dlatego, że shell zamienił $ZMIENNA na wartość zmiennej, czyli na ciąg pusty co nie jest argumentem, jeżeli nie ma argumentu to skrypt zwraca prawdę.

4. Sprawdź, czy skrypt działał poprawnie:

```
#!/bin/bash
ZMIENNA=""
if [ -n "$ZMIENNA" ]; then
        echo " Zmienna o nazwie ZMIENNA nie jest pusta"
else
        echo " Zmienna o nazwie ZMIENNA jest pusta"
fi
```

Na ekranie monitora wyświetli się wynik:

Zmienna o nazwie ZMIENNA jest pusta.

Dzieje się tak ponieważ shell zinterpretuje warunek w następujący sposób: [-n ""] (wcześniej interpretacja wyglądała tak [-n]). Podoba sytuacja jest w następującym przypadku:

```
ZMIENNA="Ala ma kota"
if [ -n $ZMIENNA ]; then
...
```

Shell zinterpretuje warunek jako [-n Ala ma kota] co jest zapisem błędnym. A

5. Wykonaj poprawny skrypt:

```
ZMIENNA="Ala ma kota"
if [ -n "$ZMIENNA" ]; then
...
```

W tym przypadku shell zinterpretuje warunek jako: [-n "Ala ma kota"] co jest zapisem poprawnym.

Jak wypisać na ekranie monitora wartość danej zmiennej, a bezpośrednio za tą wartością (bez znaku spacji) wyświetlić jakiś tekst, np. wyświetlić wartość ZMIENNA, a bezpośrednio za tą wartością słowo 'stopni'. Sprawa może wydaje się prosta, skrypt:

```
#!/bin/bash
ZMIENNA=49
echo "$ZMIENNAstopni"
```

Po uruchomieniu skryptu ekran monitora pozostanie pusty. Dlatego, że shell zinterpretował ten skrypt jako próbę wyświetlenia wartości zmiennej o nazwie ZMIENNAstopni, która nie została zadeklarowana, czyli nie istnieje. Aby skrypt dał poprawny rezultat to nazwę zmiennej trzeba oddzielić od tekstu nawiasami klamrowymi {}.

6. Wykonaj poprawny skrypt:

```
#!/bin/bash
ZMIENNA=49
echo "${ZMIENNA}stopni"
```

Po jego wywołaniu na ekranie monitora pojawi się napis:

49stopni

Przy pomocy zmiennych możemy również wywoływać polecenia linux'a. Dlatego, że shell rozwija zmienne.

7. Aby to zrozumieć prześledź działanie poniższego skryptu.

```
#!/bin/bash
Z1="ls"
Z2="-l"
$Z1 $Z2
```

Wywołanie tego skryptu spowoduje wykonanie przez shella następującego polecenia: ls –l

Sposób przekazywania do zmiennej wartości, która została wprowadzone przez użytkownika przy pomocy klawiatury, należy użyć polecenia read.

8. Aby to zrozumieć prześledź działanie poniższego skryptu.

```
#!/bin/bash
echo "Podaj swoje imie:"
read IMIE
echo "Na imie masz: $IMIE"
```

Skrypt działa następująco: na ekranie monitora pojawi się napis o treści 'Podaj swoje imie:' po czym skrypt będzie oczekiwał aż użytkownik wpisze z klawiatury swoje imię (np. Ala) i przyciśnie klawisz

ENTER, a następnie na ekranie monitora pojawi się napis o treści ‘Na imie masz: Ala’ co zakończy działanie uruchomionego skryptu. Na zmiennych można wykonywać również proste operacje matematyczne.

9. Aby to zrozumieć prześledź działanie poniższego skryptu.

```
#!/bin/bash
echo ”Podaj pierwsza liczbe:”
read L1
echo ”Podaj druga liczbe:”
read L2
SUMA=$(($L1 + $L2))
echo ”wynik dodawania wynosi: $SUMA”
c=$(expr $L1/$L2)
d=$(expr $L1 % $L2)
echo ”wynik dzielenia: $L1/$L2=$c Rest:$d”
```

Zapisz jako plik: skrypt2

Nadaj atrybut wykonywalności: chmod +x skrypt2

Uruchom akceptując polecenie: ./skrypt2

Zgłoszenie2

3. Parametry.

Do wywoływanego skryptu można przekazywać parametry jego wywołania. Parametrów można używać dokładnie tak samo jak zmiennych, nie można zmienić ich wartości. Każdy z parametrów wywołania posiada swój numer, dzięki czemu może odwołać się do każdego z nich:

S* - wszystkie parametry począwszy od parametru numer 1

$0 – nazwa programu

$1 – parametr nr 1

…

$9 – parametr nr 9

$# - liczba parametrów

$$ - numer procesu

1. Sprawdź działanie parametrów skrypt:

```
#!/bin/bash
echo ”nazwa skryptu: $0”
echo ”liczba parametrow: $#”
echo ”wszystkie parametry wywolana: $*”
echo ”parametr pierwszy: $1”
echo ”parametr drugi: $2”
echo ”parametr trzeci: $3”
echo ”numer procesu: $$”
```

2. Zapisz skrypt do pliku o nazwie skrypt3
3. Nadaj atrybut wykonywalności: chmod +x skrypt3
4. Wywołaj skrypt następująco:

./skrypt3 123 434 2312

wynik wygląda tak:

nazwa skryptu: ./skrypt3
liczba parametrow: 3
wszystkie parametry wywolania: 123 434 2312
parametr pierwszy: 123
parametr drugi: 434
parametr trzeci: 2312
numer procesu: 138

Parametry można wykorzystywać w bardziej skomplikowany sposób.

Zgłoszenie3

4. Instrukcja CASE.

W skryptach pisanych pod basha można używać instrukcji CASE. Składnia tej instrukcji wygląda następująco:

```
case $ZMIENNA in
        1) zadanie numer 1;;
        2) zadanie numer 2;;
        *) zadanie w każdym innym przypadku;;
esac
```

Instrukcja case zaczyna się od słowa case, a kończy się zapisaniem tego słowa od końca (esac), a pomiędzy tymi słowami wypisane są wszystkie opcje obsługiwane przez tą instrukcję. Zastosowanie

konstrukcji *) obejmuje wszystkie pozostałe możliwości, które nie zostały zdefiniowane w instrukcji case.

1. Sprawdź działanie skryptu:

```
#!/bin/bash
case $1 in
        piatek) echo “weekend jest juz tuz tuz”;;
        sobota) echo “juz weekend”;;
        niedziela) echo “weekend się juz powoli konczy”;;
        poniedzialek) echo ”już poniedziałek znowu trzeba isc do pracy”;;
        *) echo ”srodek tygodnia trzeba ciezko pracowac”;;
esac
```

2. Zapisz skrypt do pliku o nazwie skrypt4
3. Nadaj atrybut wykonywalności: chmod +x skrypt4
4. Wywołaj skrypt następująco:
   a. ./skrypt4 piątek - wyświetlenie tekstu ‘weekend jest już tuz tuz’,
   b. ./skrypt4 sobota - wyświetlenie teksu ‘już weekend’
   c. ./skrypt4 niedziela - wyświetlenie teksu ‘weekend się już powoli konczy’
   d. ./skrypt4 poniedzialek - wyświetlenie teksu ‘już poniedziałek znowu trzeba isc do pracy
   e. ./skrypt4 <jakikolwiek inny dzień tygodnia> - wyświetlenie teksu ‘srodek tygodnia trzeba ciezko pracowac’

Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Następnie wykorzystywany jest operator case wraz z wartością $1 (pierwszy argument podany przy uruchomieniu skryptu). Skrypt przeszukuje różne przypadki (wartości) używając case:

a Jeśli wartość $1 to "piatek", skrypt wypisuje komunikat "weekend jest już tuż tuż".

b Jeśli wartość $1 to "sobota", skrypt wypisuje komunikat "już weekend".

c Jeśli wartość $1 to "niedziela", skrypt wypisuje komunikat "weekend się już powoli kończy".

d Jeśli wartość $1 to "poniedzialek", skrypt wypisuje komunikat "już poniedziałek, znowu trzeba iść do pracy".

e Jeśli wartość $1 to cokolwiek innego niż wymienione powyżej, skrypt wypisuje komunikat "środek tygodnia, trzeba ciężko pracować".

f *) oznacza domyślny przypadek, gdy żaden z wcześniejszych przypadków nie pasuje do wartości $1. W takim przypadku, skrypt wyświetli komunikat "środek tygodnia, trzeba ciężko pracować".

3. Instrukcja esac zamyka blok case, oznaczając zakończenie wyboru.

Skrypt wykorzystuje konstrukcję case do wybierania różnych działań w zależności od wartości podanej jako argument $1.

Podsumowując, ten skrypt wybiera i wyświetla odpowiednie komunikaty w zależności od wartości pierwszego argumentu $1, który jest przekazywany przy uruchomieniu skryptu.

Zgłoszenie4

5. Instrukcja warunkowa IF.

W instrukcji warunkowej if wyróżnia się trzy zasadnicze składnie polecenia:

1. w przypadku, gdy zachodzi potrzeba sprawdzenia jakiegoś warunku np. czy dwie liczby są równe składnia wygląda następująco:

```
if warunek; then
        wyrażenie1;
        wyrażenie2;
fi
```

2. Przykład:

```
x=15
if [ "$x" -gt 10 ]; then
   echo "x jest większe od 10"
   x=$((x - 1))
fi
```

3. Zapisz skrypt do pliku o nazwie skrypt5.1
4. Nadaj atrybut wykonywalności: chmod +x skrypt5.1
5. Wywołaj skrypt następująco: ./skrypt5.1

Opis:

Zmienna x jest ustawiona na wartość 15.

Warunek ["$x" -gt 10] sprawdza, czy x jest większe niż 10.

Jeśli warunek jest prawdziwy, wykonują się poniższe instrukcje:

echo "x jest większe od 10": Wypisuje komunikat "x jest większe od 10".

x=$((x - 1)): Zmniejsza wartość x o 1.

Instrukcja fi kończy blok warunkowy "if".

6. w przypadku, gdy należy wykonać inny zestaw poleceń, jeżeli zadany warunek kończy się wynikiem negatywnym, składnia polecenia wygląda następująco:

```
if warunek; then
                wyrażenie1;
else
                wyrażenie2;
fi
```

7. Przykład:

```
x=7
if [ "$x" -gt 10 ]; then
   echo "x jest większe niż 10"
   echo "Wykonano wyrażenie 1."
else
   echo "x nie jest większe niż 10"
   echo "Wykonano wyrażenie 2."
fi
```

8. Zapisz skrypt do pliku o nazwie skrypt5.2
9. Nadaj atrybut wykonywalności: chmod +x skrypt5.2
10. Wywołaj skrypt następująco: ./skrypt5.2

Opis:

Zmienna x jest ustawiona na wartość 7.

Warunek ["$x" -gt 10] sprawdza, czy x jest większe niż 10.

Jeśli warunek jest prawdziwy, wykonują się poniższe instrukcje w bloku "then":

echo "x jest większe niż 10": Wypisuje komunikat "x jest większe niż 10".

echo "Wykonano wyrażenie 1.": Wypisuje "Wykonano wyrażenie 1.".

Jeśli warunek nie jest prawdziwy (czyli x nie jest większe niż 10), wykonują się poniższe instrukcje w bloku "else":

echo "x nie jest większe niż 10": Wypisuje komunikat "x nie jest większe niż 10".

echo "Wykonano wyrażenie 2.": Wypisuje "Wykonano wyrażenie 2.".

Instrukcja fi kończy blok warunkowy "if-else".

11. W przypadku, gdy należy sprawdzić inny warunek, jeżeli wcześniejszy warunek nie został spełniony, składnia polecenia if wygląda następująco (ilość wyrażeń elif jest dowolna):

```
if warunek1; then
```

```
        wyrażenie;
elif warunek2; then
        wyrażenie2;
fi
```

Instrukcja warunkowa if działa w taki sposób, że polecenia wewnątrz bloku if/fi, if/elif, a następnie elif/fi jeśli zadany warunek jest prawdziwy.

12. Przykład:

```
x=15
if [ "$x" -gt 20 ]; then
   echo "x jest większe niż 20"
   echo "Wykonano wyrażenie 1."
elif [ "$x" -gt 10 ]; then
   echo "x jest większe niż 10, ale nie większe niż 20"
   echo "Wykonano wyrażenie 2."
fi
```

13. Zapisz skrypt do pliku o nazwie skrypt5.3
14. Nadaj atrybut wykonywalności: chmod +x skrypt5.3
15. Wywołaj skrypt następująco: ./skrypt5.3

Opis:

Zmienna x jest ustawiona na wartość 15.

Pierwszy warunek ["$x" -gt 20] sprawdza, czy x jest większe niż 20.

Jeśli warunek pierwszy jest prawdziwy, wykonują się poniższe instrukcje w bloku "then":

echo "x jest większe niż 20": Wypisuje komunikat "x jest większe niż 20".

echo "Wykonano wyrażenie 1.": Wypisuje "Wykonano wyrażenie 1.".

Jeśli warunek pierwszy nie jest prawdziwy, skrypt przechodzi do drugiego warunku ["$x" -gt 10].

Jeśli drugi warunek jest prawdziwy, wykonują się poniższe instrukcje w bloku "then":

echo "x jest większe niż 10, ale nie większe niż 20": Wypisuje komunikat "x jest większe niż 10, ale nie większe niż 20".

echo "Wykonano wyrażenie 2.": Wypisuje "Wykonano wyrażenie 2.".

Instrukcja fi kończy blok warunkowy "if-elif".

Ten przykład pokazuje, jak używać instrukcji "if-elif" do sprawdzania wielu warunków sekwencyjnie i wykonania odpowiednich działań na podstawie spełnionego warunku.

Zgłoszenie5

6. Zapis warunków w instrukcji warunkowej IF.

Do testowania warunków używa się operatorów, które w wyniku swojego działania zwracają prawdę lub fałsz w zależności od tego czy sprawdzany warunek jest prawdziwy czy nieprawdziwy. Zazwyczaj warunek jest zapisywany w następującej postaci:

[operand1 operator operand2]

W niektórych przypadkach warunki zapisuje się tylko przy pomocy jednego operandu (drugiego - operand2). Bardzo istotny jest fakt, że pomiędzy nawiasami, operandami operandami operatorem musi istnieć odstęp w postaci spacji. Jeżeli tego odstępu nie ma to wtedy wszystko to co jest napisane między nawiasami kwadratowymi jest traktowane jako jeden operand bez jakichkolwiek operatorów.

1. Przetestuj

[$X=$Y] /# błędny zapis !!!

[$X = $Y] # zapis prawidłowy

Bardzo istotne jest także używanie znaków cudzysłowu we wszystkich warunkach, w których operator przyjmuje postać -n (na początku zostało wyjaśnione, dlaczego).

Zgłoszenie6

7. Operatory instrukcji warunkowej IF.

Najczęściej używanymi operatorami w instrukcji warunkowej if są:

-n – operand ma niezerową długość (jeden operand)

-z – operand ma zerową długość (jeden operand)

-d – istnieje katalog o nazwie operand (jeden operand)

-f – sprawdza czy operand jest plikiem (jeden operand)

-e – sprawdza czy zbiór (plik, katalog) o nazwie operand (jeden operand) istnieje

-L – sprawdzenie czy plik o nazwie operand jest dowiązaniem symbolicznym (jeden operand)

-eq – sprawdza, czy operandy są równymi liczbami (dwa operandy)

-neq – sprawdza, czy operandy są różnymi liczbami (dwa operandy)

= - sprawdza, czy operandy są jednakowymi ciągami znaków (dwa operandy)

!= - sprawdza czy operandy są różnymi ciągami znaków (dwa operandy)

-lt – sprawdza czy operand 1 jest mniejszy operand2 (dwa operandy które są liczbami całkowitymi)

-le – sprawdza czy operand1 jest równy lub mniejszy od operand2 (dwa operandy które SA liczbami całkowitymi)

-gt – sprawdza czy operand1 jest większy od operand2 (dwa operandy które SA liczbami całkowitymi)

-ge – sprawdza czy operand1 jest równy lub większy od operand2 (dwa operandy które SA liczbami całkowitymi)

To nie są wszystkie typy operatorów, które są dostępne w bashu, ale wystarczają do napisania większości skryptów. Istnieje możliwość poznania reszty operatorów które są dostępne w bashu. Skorzystaj z manuala wpisując polecenie:

man bash

1. Napisz skrypt wykorzystujący instrukcję warunkową if

```
#!/bin/bash
if [ "$1" -eq "$2" ]; then
   echo "liczby $1 i $2 są równe";
else
   echo "liczby $1 i $2 są różne";
fi

if [ -e "$3" ]; then
   echo "plik o nazwie $3 istnieje";
   if [ -L "$3" ]; then
      echo "i jest dowiązaniem symbolicznym";
   elif [ -f "$3" ]; then
      echo "i jest zwykłym plikiem";
   fi
else
   echo "plik o nazwie $3 nie istnieje"
fi
```

2. Uruchomienie skryptu:

```
chmod +x nazwa_skryptu.sh
./nazwa_skryptu 1 3 test.txt
chmod +x skrypt7
./skrypt7 1 3 test.txt
```

Wynik działania skryptu (zakładając ze plik podany jako trzeci parametr istnieje i że jest zwykłym plikiem):

```
liczby 1 i 3 sa rozne
plik o nazwie test.txt istnieje
```

i jest zwykłym plikiem

Opis krok po kroku działanie powyższego skryptu:

1. Skrypt rozpoczyna się deklaracją #!/bin/bash, która określa, że skrypt ma być interpretowany przy użyciu powłoki Bash.
2. W pierwszym warunku if:

a Skrypt porównuje wartości dwóch argumentów, które zostały podane przy uruchomieniu skryptu: $1 i $2.

b -eq to operator równości w kontekście liczbowym.

c Jeśli wartości są równe, skrypt wypisuje komunikat: "liczby [wartość1] i [wartość2] są równe".

d W przeciwnym przypadku, gdy wartości nie są równe, skrypt wypisuje komunikat: "liczby [wartość1] i [wartość2] są różne".

3. W drugim bloku if:
   a Skrypt sprawdza czy istnieje plik o nazwie podanej jako trzeci argument (zmienna $3).
   b -e to operator sprawdzający istnienie pliku.
   c Jeśli plik istnieje, skrypt wypisuje komunikat: "plik o nazwie [nazwa_pliku] istnieje".
   d Następnie skrypt wchodzi w drugi warunek if:
      1. Skrypt sprawdza, czy podany plik jest dowiązaniem symbolicznym (używając -L) i wypisuje odpowiedni komunikat.
      2. Jeśli to nie jest dowiązanie symboliczne, skrypt sprawdza, czy plik jest zwykłym plikiem (-f) i wypisuje odpowiedni komunikat.
4. Jeśli plik o nazwie podanej jako trzeci argument nie istnieje, skrypt wypisuje komunikat: "plik o nazwie [nazwa_pliku] nie istnieje".

Ogólnie, skrypt ten porównuje dwie liczby oraz sprawdza istnienie i typ pliku, a następnie wypisuje odpowiednie komunikaty w zależności od wyników tych operacji.

Zgłoszenie7

Podsumowanie:

Po wykonaniu wszystkich czynności z powyższej instrukcji przeczytaj ponownie z zrozumieniem cel ogólny i cele szczegółowe, które znajdują się na pierwszej stronie instrukcji. Jeżeli one zostały niezrealizowane to powtarzaj wykonie tej instrukcji w szkole lub/i w domu do momentu zrealizowania.