Temat: Automatyzacja pracy - skrypty - pętle w Linux.

Cel ogólny lekcji: Zdobycie umiejętności zaawansowanego tworzenia i wykorzystywania skryptów oraz pętli w języku Bash w celu automatyzacji pracy i efektywnego zarządzania systemem Linux.

Cele Szczegółowe:

1. Podczas tej lekcji uczestnicy zdobędą umiejętności, które umożliwią im:
2. Zrozumienie kluczowego znaczenia automatyzacji pracy oraz roli, jaką odgrywają skrypty i pętle w osiąganiu tego celu.
3. Poznanie składni i mechanizmu działania instrukcji case w języku Bash i wykorzystanie jej do tworzenia warunkowych operacji w skryptach.
4. Wykorzystanie pętli for do iteracji przez listy oraz realizacji określonych poleceń na kolejnych elementach.
5. Stworzenie interaktywnego menu użytkownika poprzez zastosowanie pętli select, umożliwiającego wybieranie opcji.
6. Zapoznanie się z pętlami while i until w celu tworzenia iteracyjnych operacji opartych na warunkach logicznych.
7. Tworzenie skryptów, które wykorzystują polecenie read do interaktywnego pobierania danych od użytkownika i przetwarzania ich w skrypcie.
8. Ćwiczenie umiejętności tworzenia i uruchamiania skryptów, nadawania im prawa wykonywalności.
9. Zrozumienie różnic między pętlą while a until oraz identyfikacja odpowiednich sytuacji do ich zastosowania.
10. Zrozumienie znaczenia i zastosowania zmiennych specjalnych w Bashu, takich jak $0, $# oraz $*.
11. Opanowanie umiejętności korzystania z mechanizmu interpretacji wyrażeń arytmetycznych za pomocą $((...)) oraz $[...] do przeprowadzania obliczeń.
12. Nauczenie się tworzenia i wykorzystywania funkcji w skryptach Bash, zwiększając czytelność kodu i modularność.
13. Zrozumienie procesu załączania plików z funkcjami do skryptów oraz umiejętność korzystania z funkcji zdefiniowanych w innych plikach.
14. Ćwiczenie przekazywania parametrów do funkcji i wykorzystywanie ich wewnątrz funkcji.
15. Zapoznanie się z poleceniem let do wykonywania operacji arytmetycznych i przypisywania wyników do zmiennych.
16. Stworzenie bardziej zaawansowanych skryptów, które operują na plikach HTML w sposób zdefiniowany przez uczestników.

Osiągnięcie tych celów pozwoli uczestnikom skutecznie wykorzystywać zaawansowane mechanizmy języka Bash, takie jak skrypty, pętle oraz instrukcje warunkowe, w celu automatyzacji zadań, manipulacji danymi oraz zarządzania systemem operacyjnym Linux.

Poprzez osiągnięcie tych celów, uczestnicy nabędą umiejętności tworzenia bardziej zaawansowanych skryptów w języku Bash, wykorzystując zaawansowane funkcje, mechanizmy matematyczne oraz zmienne specjalne. Uczestnicy będą w stanie tworzyć bardziej złożone skrypty, które efektywnie automatyzują powtarzalne zadania oraz manipulują danymi zgodnie z konkretnymi wymaganiami.

Wykonaj i przetestuj działanie wszystkich skryptów poniżej. W zszycie zapisz nowe informacje.

1. Pętle.

Pętle stosuje się w celu wykonania pewnych instrukcji dla kolejnych iteracji lub kilku parametrów. W bashu przy tworzeniu skryptów możemy się posługiwać pętlami. Do najczęściej stosowanych pętli zalicza się pętle for i while.

a) **Instrukcja case** Pozwala na dokonanie wyboru spośród kilku wzorców. Sprawdzana jest wartość zmiennej po słowie kluczowym **case** i porównywana ze wszystkimi wariantami po kolei. Musi być taka sama jak wzorzec, do którego chcemy się odwołać. Jeśli dopasowanie zakończy się sukcesem wykonane zostanie polecenie lub polecenia przypisane do danego wzorca. W przeciwnym wypadku użyte zostanie polecenie domyślne oznaczone symbolem gwiazdki: ***) polecenie_domyślne**. Co jest dobrym zabezpieczeniem na wypadek błędów popełnionych przez użytkownika skryptu. **Składnia:**

```
case zmienna in
  "wzorzec1") polecenie1 ;;
  "wzorzec2") polecenie2 ;;
  "wzorzec3") polecenie3 ;;
  *) polecenie_domyślne
esac
```

Wykonaj przykład:

```
#!/bin/bash
echo "Podaj cyfrę dnia tygodnia"
read d
case "$d" in
  "1") echo "Poniedziałek" ;;
  "2") echo "Wtorek" ;;
  "3") echo "Środa" ;;
```

```
  "4") echo "Czwartek" ;;
  "5") echo "Piątek" ;;
  "6") echo "Sobota" ;;
  "7") echo "Niedziela" ;;
  *) echo "Nic nie wybrałeś"
esac
```

Skrypt bashowy jest przykładem użycia instrukcji case do mapowania liczby wprowadzonej przez użytkownika na odpowiadający jej dzień tygodnia. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt wykorzystuje polecenie echo do wypisania komunikatu: "Podaj cyfrę dnia tygodnia".
3. Skrypt używa polecenia read do odczytania wartości wprowadzonej przez użytkownika i przypisania jej do zmiennej d.
4. Skrypt wykorzystuje instrukcję case do porównania zmiennej d z różnymi wartościami:
5. Jeśli d to "1", skrypt wyświetla "Poniedziałek".
6. Jeśli d to "2", skrypt wyświetla "Wtorek".
7. Jeśli d to "3", skrypt wyświetla "Środa".
8. Itd., aż do "7", gdzie skrypt wyświetla "Niedziela".
9. W przypadku, gdy d nie pasuje do żadnej z powyższych wartości, skrypt wyświetla "Nic nie wybrałeś".
10. Skrypt kończy się.

Podsumowując, ten skrypt pozwala użytkownikowi wprowadzić cyfrę od 1 do 7 i na podstawie tej cyfry wyświetla odpowiadający jej dzień tygodnia. Jeśli wprowadzona cyfra nie mieści się w zakresie 1-7, wyświetla komunikat "Nic nie wybrałeś".

Utwórz folder cw2_1516 a w nim utwórz plik skryptu o nazwie skrypt1a.

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh)

```
chmod +x skrypt1a
```

Aby uruchomić skrypt, otwórz terminal i wykonaj komendę:

```
./skrypt1a
```

Skrypt poprosi Cię o wprowadzenie cyfry dnia tygodnia, a następnie wyświetli odpowiadający dzień lub komunikat "Nic nie wybrałeś".

zgłoszenie 1

Przy zastosowaniu; po przypasowaniu do warunku i wykonaniu instrukcji wychodzi z konstrukcji case.

Przy zastosowaniu; **&** wykonuje instrukcje także kolejnego warunku.

Jak widać mamy w skrypcie wzorce od **1** do **7** odpowiadające liczbie dni tygodnia, każdemu przypisane jest jakieś polecenie, tutaj ma wydrukować na ekranie nazwę dnia tygodnia. Jeśli podamy 1 polecenie **read** czytające dane ze standardowego wejścia przypisze zmiennej **d** wartość 1 i zostanie wykonany skok do wzorca **1**, na ekranie zostanie wyświetlony napis *Poniedziałek*. W przypadku, gdy podamy cyfrę o liczbie większej niż **7** lub wpiszemy inny znak na przykład literę to wykonany zostanie wariant defaultowy oznaczony gwiazdką:
***) echo "Nic nie wybrałeś"**.

b) **Pętla for w**ykonuje polecenia zawarte wewnątrz pętli, na każdym składniku listy (iteracja). **Składnia:**

```
for zmienna in lista
do
  polecenie
done
```

Przykład:

```
for x in jeden dwa trzy
do
  echo "To jest $x"
done
```

Zmiennej **x** przypisana jest lista, która składa się z trzech elementów: **jeden**, **dwa**, **trzy**.
Wartością zmiennej **x** staje się po kolei każdy element listy, na wszystkich wykonywane jest polecenie: **echo "To jest $x"**. Pętla for jest bardzo przydatna w sytuacjach, gdy chcemy wykonać jakąś operację na wszystkich plikach w danym katalogu.

1. Na przykład chcemy uzyskać listę wszystkich plików o danym rozszerzeniu znajdujących się w jakimś katalogu:

```
#!/bin/bash
for x in *html
do
  echo "To jest plik $x"
done
```

Skrypt bashowy wykorzystuje pętlę for do iteracji przez pliki w bieżącym katalogu, które mają rozszerzenie .html. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.

2. Skrypt używa pętli for do iteracji przez pliki w bieżącym katalogu, które mają rozszerzenie .html.
   a. Dla każdego pliku spełniającego ten warunek, skrypt wykonuje następujące kroki:
      1. Wykorzystuje polecenie echo do wypisania komunikatu "To jest plik [nazwa_pliku]".
      2. Zmienna x przechowuje nazwę aktualnego pliku w iteracji.
3. Skrypt kończy się.

Podsumowując, ten skrypt wypisuje komunikaty o plikach w bieżącym katalogu, które mają rozszerzenie .html.

Utwórz pliki .html:

```
touch plik{,2}.html
```

Utwórz plik skryptu o nazwie skrypt1b1

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1b1
```

Uruchom skrypt, wykonaj komendę:

```
./skrypt1b1
```

Skrypt wyświetli komunikaty o każdym pliku .html w tym katalogu.

4. lub jeśli chcemy zmienić nazwy plików pisane *DUŻYMI* literami na nazwy pisane *małymi* literami:

```
#!/bin/bash
for nazwa in *
do
  mv $nazwa `echo $nazwa | tr '[A-Z]' '[a-z]'`
done
```

Za zmianę *DUŻYCH* liter na *małe* (i na odwrót) odpowiedzialne jest polecenie **tr**.

Skrypt Bash zmienia nazwy wszystkich plików w bieżącym katalogu, zamieniając duże litery na małe litery. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa pętli for do iteracji przez wszystkie pliki i foldery w bieżącym katalogu.
   a. Dla każdej nazwy (pliku lub folderu) w iteracji:
      1. Skrypt wykorzystuje polecenie tr w celu zamiany dużej litery na małą w nazwie.
      2. Używa polecenia mv do zmiany nazwy pliku na nową nazwę z małymi literami.
3. Skrypt kończy się.

Podsumowując, ten skrypt zmienia nazwy wszystkich plików i folderów w bieżącym katalogu, zamieniając duże litery na małe litery.

Otwórz terminal, przejdź do katalogu, w którym znajdują się pliki i foldery, i utwórz pliki o nazwach *DUŻYCH* liter:

```
touch PLIK{,2}
```

Utwórz plik skryptu o nazwie skrypt1b2

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1b2
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1b2
```

Skrypt przejdzie przez wszystkie elementy katalogu i zmieni nazwy plików/folderów o dużych literach na ich wersje z małymi literami.

c) **Pętla select w**ygeneruje z listy słów po **in** proste ponumerowane menu, każdej pozycji odpowiada kolejna liczba od 1 wzwyż. Poniżej menu znajduje się znak zachęty PS3 gdzie wpisujemy cyfrę odpowiadająca wybranej przez nas pozycji w menu. Jeśli nic nie wpiszemy i wciśniemy ENTER, menu będzie wyświetlone ponownie. To co wpisaliśmy zachowywane jest w zmiennej **REPLY.**
Gdy odczytane zostaje **EOF** (ang. *End Of File*) czyli znak końca pliku **(CTRL+D)** to select kończy pracę. Pętla działa dotąd, dopóki nie wykonane zostaje polecenie **break** lub **return**. **Składnia:**

```
select zmienna in lista
do
  polecenie
done
```

Wykonaj praktyczny przykład:

```
#!/bin/bash
echo "Co wybierasz?"
select y in X Y Z Quit
do
  case $y in
    "X") echo "Wybrałeś X" ;;
    "Y") echo "Wybrałeś Y" ;;
    "Z") echo "Wybrałeś Z" ;;
    "Quit") exit ;;
    *) echo "Nic nie wybrałeś"
```

```
  esac
break
done
```

Skrypt Bash to przykład użycia polecenia select do tworzenia interaktywnego menu wyboru. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt wykorzystuje polecenie echo do wypisania komunikatu: "Co wybierasz?".
3. Skrypt używa polecenia select do tworzenia interaktywnego menu wyboru, w którym użytkownik może wybrać jedną z opcji: "X", "Y", "Z" lub "Quit".
    a. Pętla select oczekuje na wybór użytkownika i przypisuje wybraną opcję do zmiennej $y.
    b. Skrypt wyświetla dostępne opcje oraz numeracje.
    c. Skrypt kontynuuje wyświetlanie menu do momentu, gdy użytkownik dokona wyboru i naciśnie Enter.
4. Skrypt używa instrukcji case do porównania zmiennej $y z różnymi wartościami:
    a. Jeśli $y to "X", skrypt wyświetla "Wybrałeś X".
    b. Jeśli $y to "Y", skrypt wyświetla "Wybrałeś Y".
    c. Jeśli $y to "Z", skrypt wyświetla "Wybrałeś Z".
    d. Jeśli $y to "Quit", skrypt wychodzi z programu za pomocą exit.
    e. W przeciwnym przypadku (jeśli wybrano inną opcję), skrypt wyświetla "Nic nie wybrałeś".
5. Skrypt używa polecenia break, aby zakończyć pętlę select po dokonaniu wyboru.
6. Skrypt kończy się.

Podsumowując, ten skrypt tworzy interaktywne menu, w którym użytkownik może wybierać opcje "X", "Y", "Z" lub "Quit". W zależności od wyboru, skrypt wypisuje odpowiednie komunikaty.

Utwórz plik skryptu o nazwie skrypt1c

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1c
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1c
```

Skrypt wyświetli menu wyboru, które umożliwi wybranie opcji i dostarczy odpowiadające im komunikaty. Każdy wybór jest odpowiednio obsługiwany przez instrukcje case.

Zobaczymy proste ponumerowane menu, składające się z czterech elementów: **X**, **Y**, **Z** i **Quit**, teraz wystarczy tylko wpisać numer inetersującej nas opcji, a resztę zrobi instrukcja case. Polecenie **break**,

które znajduje się w przedostatniej linii skryptu, kończy pracę pętli. Słowo kluczowe **continue** - przerywa wykonywanie instrukcji w ciele pętli i przechodzi do następnej iteracji.

zgłoszenie 2

d) **Pętla while** sprawdza warunek czy jest prawdziwy, jeśli tak to wykonane zostanie polecenie lub lista poleceń zawartych wewnątrz pętli, gdy warunek stanie się fałszywa pętla zostanie zakończona. **Składnia:**

```
while warunek
do
polecenie
done
```

Wykonaj przykład:

```
#!/bin/bash
x=1;
while [ $x -le 10 ]; do
echo "Napis pojawił się po raz: $x"
x=$[x + 1]
done
```

Skrypt Bash używa pętli while do wyświetlania komunikatu 10 razy, z numerem iteracji. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt przypisuje wartość 1 do zmiennej x.
3. Skrypt używa pętli while, która będzie działać, dopóki wartość zmiennej x jest mniejsza lub równa 10.
4. Wewnątrz pętli skrypt używa polecenia echo do wypisania komunikatu zawierającego numer iteracji: "Napis pojawił się po raz: [wartość_x]".
5. Skrypt zwiększa wartość zmiennej x o 1, korzystając ze składni $[x + 1].
6. Pętla while będzie kontynuować działanie dla wartości zmiennej x od 1 do 10.
7. Skrypt kończy się po zakończeniu pętli while.

Podsumowując, ten skrypt wyświetla komunikat "Napis pojawił się po raz: [numer_iteracji]" dla wartości x od 1 do 10.

Utwórz plik skryptu o nazwie skrypt1d

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1d
```

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1d

Skrypt wyświetli komunikaty, zaczynając od "Napis pojawił się po raz: 1" do "Napis pojawił się po raz: 10".

Sprawdzany jest warunek czy zmienna **x** o wartości początkowej 1 jest mniejsza lub równa 10, warunek jest prawdziwy w związku z czym wykonywane są polecenia zawarte wewnątrz pętli: **echo "Napis pojawił się po raz: $x"** oraz **x=$[x + 1]**, które zwiększa wartość zmiennej **x** o 1.

Gdy wartość **x** przekroczy 10, wykonanie pętli zostanie przerwane.

zgłoszenie 3

e) **Pętla until s**prawdza czy warunek jest prawdziwy, gdy jest fałszywy wykonywane jest polecenie lub lista poleceń zawartych wewnątrz pętli, między słowami kluczowymi **do** a **done**. Pętla **until** kończy swoje działanie w momencie, gdy warunek stanie się prawdziwy. **Składnia:**

```
until warunek
do
polecenie
done
```

Wykonaj przykład:

```
#!/bin/bash
x=1;
until [ $x -ge 10 ]; do
echo "Napis pojawił się po raz: $x"
x=$[x + 1]
done
```

Skrypt Bash używa pętli until do wyświetlania komunikatu 9 razy, z numerem iteracji. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt przypisuje wartość 1 do zmiennej x.
3. Skrypt używa pętli until, która będzie działać, dopóki wartość zmiennej x nie jest większa lub równa 10.
    a. Wewnątrz pętli skrypt używa polecenia echo do wypisania komunikatu zawierającego numer iteracji: "Napis pojawił się po raz: [wartość_x]".

b. Skrypt zwiększa wartość zmiennej x o 1, korzystając ze składni $[x + 1].

4. Pętla until będzie kontynuować działanie dla wartości zmiennej x od 1 do 9, ponieważ wyrażenie [$x -ge 10] stanie się prawdziwe, gdy x będzie wynosić 10.
5. Skrypt kończy się po zakończeniu pętli until.

Utwórz plik skryptu o nazwie skrypt1e

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x skrypt1e

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1e

Podsumowując, ten skrypt wyświetla komunikaty "Napis pojawił się po raz: [numer_iteracji]" dla wartości x od 1 do 9.

Mamy zmienną **x**, która przyjmuje wartość 1, następnie sprawdzany jest warunek czy wartość zmiennej **x**jest większa lub równa 10, jeśli nie to wykonywane są polecenia zawarte wewnątrz pętli. W momencie, gdy zmienna **x** osiągnie wartość, 10 pętla zostanie zakończona. **exit** - Zakończ wykonywanie skryptu. Jako argument przyjmuje stan zakończenia.

zgłoszenie 4

f) **Polecenie read** czyta ze standardowego wejścia pojedynczy wiersz. **Składnia:**

read -opcje nazwa_zmiennej

1. Wykonaj przykład:

```
#!/bin/bash
echo -n "Wpisz coś:\a"
read wpis
echo "$wpis"
```

Skrypt Bash to prosty przykład, który prosi użytkownika o wpisanie tekstu, a następnie wypisuje ten wpisany tekst. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo z opcją -n, co powoduje, że komunikat zostanie wypisany bez automatycznego dodawania znaku nowej linii.

a. Komunikat to "Wpisz coś:" oraz znak alarmu \a, który wygeneruje dźwięk (zależy to od ustawień terminala).

3. Skrypt używa polecenia read do odczytania wpisanego tekstu i przypisania go do zmiennej wpis.
4. Skrypt używa polecenia echo do wypisania wpisanego tekstu z wcześniej zdefiniowanej zmiennej wpis.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f1

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x skrypt1f1

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1f1

Podsumowując, ten skrypt prosi użytkownika o wpisanie tekstu, oczekuje na wprowadzenie, a następnie wypisuje wpisany tekst. Skrypt poprosi Cię o wpisanie tekstu, a po wprowadzeniu wypisze ten tekst.
To co zostało wpisane trafi do zmiennej **wpis**, której to wartość czyta polecenie **read wpis**, zmienna nie musi być wcześniej tworzona, jeśli istniała wcześniej, jej zawartość zostanie zastąpiona tym co wpisaliśmy. Poza tym, może generować dźwiękowy sygnał alarmu.

zgłoszenie 5

2. Wykonaj przykład:

```
#!/bin/bash
echo "Wpisz coś:"
answer="napis"
read
echo "$answer"
```

Skrypt Bash prosi użytkownika o wprowadzenie tekstu, jednak wprowadzony tekst nie zostaje przypisany do zmiennej answer, a następnie wypisuje wartość zmiennej answer. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo do wypisania komunikatu "Wpisz coś:".
3. Skrypt przypisuje stałą wartość "napis" do zmiennej answer, która pierwotnie nie jest używana w skrypcie.
4. Skrypt używa polecenia read do odczytania wprowadzonego tekstu, ale nie przypisuje go do zmiennej. Wartość wprowadzonego tekstu jest domyślnie przypisywana do zmiennej wbudowanej $REPLY.

5. Skrypt wykorzystuje polecenie echo do wypisania zawartości zmiennej answer, która ma przypisaną wartość "napis".
6. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f2

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1f2
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1f2
```

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie tekstu, ale wprowadzony tekst jest pomijany, a następnie wypisywana jest wartość zmiennej answer, która zawiera stałą wartość "napis". Skrypt wypisze "napis" po wprowadzeniu tekstu. Wcześniejsza wartość zmiennej **answer** została zastąpiona.

zgłoszenie 6

Polecenie **read** pozwala na przypisanie kilku wartości kilku zmiennym.

3. Wykonaj przykład:

```
#!/bin/bash
echo "Wpisz cztery wartości:"
read a b c
echo "Wartość zmiennej a to: $a"
echo "Wartość zmiennej b to: $b"
echo "Wartość zmiennej c to: $c"
```

Skrypt prosi użytkownika o wprowadzenie trzech wartości, a nie czterech. Oto poprawiony opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo do wypisania komunikatu "Wpisz trzy wartości:".
3. Skrypt używa polecenia read z trzema zmiennymi (a, b i c) do odczytania wprowadzonych wartości. Wprowadzone wartości są przypisywane do odpowiednich zmiennych.
4. Skrypt używa polecenia echo do wypisania wartości zmiennej a z odpowiednim komunikatem.
5. Skrypt używa polecenia echo do wypisania wartości zmiennej b z odpowiednim komunikatem.
6. Skrypt używa polecenia echo do wypisania wartości zmiennej c z odpowiednim komunikatem.

7. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f3

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x skrypt1f3

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1f3

Skrypt poprosi Cię o wprowadzenie trzech wartości. Po wprowadzeniu wartości i naciśnięciu Enter, skrypt wypisze te wartości wraz z komunikatami.

zgłoszenie 7

Wybrane opcje:

4. **-p -** Pokaże znak zachęty bez kończącego znaku nowej linii.

```
#!/bin/bash
read -p "Pisz:" odp
echo "$odp"
```

Skrypt Bash prosi użytkownika o wprowadzenie tekstu, a następnie wypisuje ten tekst. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia read z opcją -p do wypisania komunikatu "Pisz:" w tej samej linii, bez przechodzenia do nowej linii.
3. Skrypt używa zmiennej odp do odczytania wprowadzonego tekstu.
4. Skrypt wykorzystuje polecenie echo do wypisania zawartości zmiennej odp, która zawiera wprowadzony tekst.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f4

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x skrypt1f4

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1f4

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie tekstu, wypisuje komunikat "Pisz:" w tej samej linii, a następnie wypisuje wprowadzony tekst.

Skrypt poprosi Cię o wprowadzenie tekstu. Po wprowadzeniu tekstu i naciśnięciu Enter, skrypt wypisze ten tekst.

5. **-a -** Kolejne wartości przypisywane są do kolejnych indeksów zmiennej tablicowej.

Wykonaj Przykład:

```
#!/bin/bash
echo "Podaj elementy zmiennej tablicowej:"
read tablica
echo "${tablica[*]}"
```

Skrypt Bash prosi użytkownika o wprowadzenie elementów zmiennej tablicowej, a następnie wypisuje wszystkie elementy tej tablicy. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo do wypisania komunikatu "Podaj elementy zmiennej tablicowej:".
3. Skrypt używa polecenia read do odczytania wprowadzonych elementów zmiennej tablicowej. Wszystkie wprowadzone elementy zostaną traktowane jako jeden łańcuch znaków i zostaną przypisane do zmiennej tablica.
4. Skrypt wykorzystuje specjalną zmienną ${tablica[*]} do wypisania wszystkich elementów tablicy. Odpowiednio sformatowana zmienna ta wypisze wszystkie elementy oddzielone spacją.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f5

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1f5
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1f5
```

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie elementów zmiennej tablicowej jako pojedynczego łańcucha znaków, a następnie wypisuje te elementy oddzielone spacją.

Skrypt poprosi Cię o wprowadzenie elementów zmiennej tablicowej. Po wprowadzeniu elementów i naciśnięciu Enter, skrypt wypisze wszystkie elementy oddzielone spacją.

zgłoszenie 8

6. **-e -** Jeśli nie podano żadnej nazwy zmiennej, wiersz trafia do **$REPLY**.

Wykonaj przykłady:

```
#!/bin/bash
echo "Wpisz coś:"
read -e
echo "$REPLY"
```

Skrypt Bash prosi użytkownika o wprowadzenie tekstu i następnie wypisuje ten tekst. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo do wypisania komunikatu "Wpisz coś:".
3. Skrypt używa polecenia read z opcją -e do odczytania wprowadzonego tekstu. Opcja -e włącza funkcję edycji tekstu, umożliwiając modyfikację wprowadzanego tekstu przed zatwierdzeniem.
4. Skrypt wykorzystuje wbudowaną zmienną $REPLY do wypisania wprowadzonego tekstu. Wartość tej zmiennej zawiera wprowadzony tekst.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f6

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1f6
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1f6
```

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie tekstu, umożliwia edycję wprowadzonego tekstu przed zatwierdzeniem, a następnie wypisuje ten tekst.

Skrypt poprosi Cię o wprowadzenie tekstu. Po wprowadzeniu tekstu, możesz edytować go, a następnie naciśnięcie Enter spowoduje wypisanie zmodyfikowanego tekstu.

zgłoszenie 9

7. Opcje

-t *timeout* - czas wygaśnięcia w sekundach

-s - nie wyświetlaj znaków wpisanych przez użytkownika.

Wykonaj przykłady:

```
#!/bin/bash
#Hasło wpisywane bez echa, max. przez 30 sekund
read -p "Password: " -s  -t 30 password
echo $password
```

Skrypt Bash prosi użytkownika o wprowadzenie hasła bez wyświetlania wprowadzanych znaków (dla zachowania prywatności), a także określa maksymalny czas oczekiwania na wprowadzenie hasła (30 sekund). Następnie skrypt wypisuje wprowadzone hasło. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia read z opcją -p do wypisania komunikatu "Password: " w tej samej linii, bez przechodzenia do nowej linii.
3. Skrypt używa opcji -s, co powoduje, że wprowadzane znaki nie będą wyświetlane na ekranie.
4. Skrypt używa opcji -t 30, aby ustawić maksymalny czas oczekiwania na wprowadzenie hasła na 30 sekund.
5. Skrypt przypisuje wprowadzone hasło do zmiennej password.
6. Skrypt używa polecenia echo do wypisania wartości zmiennej password, która zawiera wprowadzone hasło.
7. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1f7

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x skrypt1f7
```

Aby uruchomić skrypt, wykonaj komendę:

```
./skrypt1f7
```

Podsumowując, ten skrypt prosi użytkownika o wprowadzenie hasła bez wyświetlania wprowadzanych znaków i określa maksymalny czas oczekiwania na wprowadzenie hasła. Następnie wypisuje to wprowadzone hasło.

Skrypt poprosi Cię o wprowadzenie hasła. Po wprowadzeniu hasła i naciśnięciu Enter, skrypt wypisze to hasło.

zgłoszenie 10

g) **Funkcje** coś w rodzaju podprogramów. Stosuje się je gdy w naszym skrypcie powtarza się jakaś grupa poleceń, po co pisać je kilka razy, skoro można to wszystko umieścić w funkcjach. Do danej funkcji

odwołujemy się podając jej nazwę, a wykonane zostanie wszystko co wpisaliśmy między nawiasy **{ }**, skraca to znacznie długość skryptu. **Składnia:**

```
function nazwa_funkcji
{
polecenie1
polecenie2
polecenie3
}
```

lub:

```
function nazwa_funkcji()
{
polecenie1
polecenie2
polecenie3
}
```

1. Wykonaj przykład:

```
#!/bin/bash
function napis
{
echo "To jest napis"
}
napis
```

Skrypt Bash definiuje funkcję o nazwie napis, która wypisuje komunikat "To jest napis". Następnie skrypt wywołuje tę funkcję. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt definiuje funkcję napis, używając słowa kluczowego function, a następnie otwiera i zamyka nawiasy klamrowe { ... }, w których zawarte są polecenia funkcji.
3. Wewnątrz funkcji, skrypt używa polecenia echo do wypisania komunikatu "To jest napis".
4. Skrypt wywołuje funkcję napis poprzez wpisanie jej nazwy na końcu skryptu.

5. Skrypt kończy się.

Utwórz plik skryptu o nazwie skrypt1g1

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x skrypt1g1

Aby uruchomić skrypt, wykonaj komendę:

./skrypt1g1

Podsumowując, ten skrypt definiuje funkcję o nazwie napis, która wypisuje komunikat, a następnie wywołuje tę funkcję.

Skrypt wypisze komunikat "To jest napis".

Nazwę funkcji umieszczamy po słowie kluczowym **function**, w powyższym przykładzie mamy funkcje o nazwie **napis**, odwołujemy się do niej podając jej nazwę, wykonane zostaną wtedy wszystkie polecenia, jakie jej przypiszemy.

zgłoszenie 11

Funkcje mogą się znajdować w innym pliku, co uczyni nasz skrypt bardziej przejrzystym i wygodnym, tworzy się własne pliki nagłówkowe, wywołuje się je tak:

. ~/naszplik_z_funkcjami

nazwa_funkcji

Trzeba pamiętać o podaniu kropki + spacja przed nazwą pliku

2. Wykonaj przykład:

```
#!/bin/bash
function nasza_funkcja
{
echo -e 'Właśnie użyłeś funkcji o nazwie "nasza_funkcja".\a'
}
```

Skrypt Bash definiuje funkcję o nazwie nasza_funkcja, która wypisuje komunikat "Właśnie użyłeś funkcji o nazwie "nasza_funkcja"." oraz wydaje sygnał dźwiękowy (bell). Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt definiuje funkcję nasza_funkcja, używając słowa kluczowego function, a następnie otwiera i zamyka nawiasy klamrowe { ... }, w których zawarte są polecenia funkcji.

3. Wewnątrz funkcji, skrypt używa polecenia echo z opcją -e do wypisania komunikatu.
Opcja -e pozwala na interpretację specjalnych sekwencji ucieczki (escape sequences), takich jak \a (sygnał dźwiękowy).
4. Skrypt kończy się.
Utwórz plik skryptu o nazwie funkcja
Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):
chmod +x funkcja
Aby uruchomić skrypt, wykonaj komendę:
./funkcja
Podsumowując, ten skrypt definiuje funkcję o nazwie nasza_funkcja, która wypisuje komunikat z wykorzystaniem specjalnej sekwencji ucieczki \a, co generuje sygnał dźwiękowy.
Aby wywołać tę funkcję, możesz dopisać poniższe polecenie w skrypcie:
nasza_funkcja
Skrypt wypisze komunikat oraz wyda sygnał dźwiękowy (jeśli to obsługiwane przez terminal).

3. Teraz pozostało jeszcze utworzyć skrypt, w którym wywołamy funkcje: **nasza_funkcja**:

```
#!/bin/bash
echo "Test funkcji."
. funkcja
nasza_funkcja
```

Skrypt Bash wypisuje komunikat "Test funkcji.", a następnie wczytuje zawartość pliku o nazwie "funkcja" i wywołuje funkcję nasza_funkcja. Przy założeniu, że plik "funkcja" zawiera definicję funkcji nasza_funkcja, Opis działania skryptu:
1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo do wypisania komunikatu "Test funkcji.".
3. Skrypt używa kropki i spacji (.) do załączenia zawartości pliku o nazwie "funkcja". To polecenie wczytuje i wykonuje zawartość pliku "funkcja" w bieżącym kontekście skryptu.
4. Skrypt wywołuje funkcję nasza_funkcja, która została wczytana z pliku "funkcja".
5. Skrypt kończy się.
Utwórz plik skryptu o nazwie glowny_skrypt
Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):
chmod +x glowny_skrypt

Aby uruchomić skrypt, wykonaj komendę:

./glowny_skrypt

Podsumowując, ten skrypt wypisuje komunikat, wczytuje i wykonuje zawartość pliku "funkcja", a następnie wywołuje funkcję nasza_funkcja.

Skrypt wypisze komunikat "Test funkcji.", a także wykona funkcję nasza_funkcja, jeśli taka funkcja jest zawarta w pliku "funkcja".

zgłoszenie 12

Przekazywanie parametrów do funkcji następuje dokładnie tak samo jak do każdego polecenia, które jest w skrypcie: nazwa_funkcji parametr_1 parametr_2

Wykonaj przykład:

```
#!/bin/bash
funkcja_z_parametrami()
{
        echo "Przekazano $# parametrów"
        echo "Parametr $1"
        echo "Parametr $2"
}
funkcja_z_parametrami "param1" "param2"
```

Skrypt Bash definiuje funkcję o nazwie funkcja_z_parametrami, która przyjmuje dwa parametry i wypisuje informacje o ilości przekazanych parametrów oraz wartości tych parametrów. Następnie skrypt wywołuje tę funkcję, przekazując dwa parametry "param1" i "param2". Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt definiuje funkcję funkcja_z_parametrami, używając nawiasów klamrowych { ... }, w których zawarte są polecenia funkcji.
3. Wewnątrz funkcji, skrypt używa specjalnej zmiennej $#, aby uzyskać liczbę przekazanych parametrów i wypisuje tę liczbę za pomocą polecenia echo.
4. Skrypt używa zmiennej $1 i $2, aby wypisać wartości przekazanych parametrów.
5. Skrypt wywołuje funkcję funkcja_z_parametrami i przekazuje dwa parametry "param1" i "param2".
6. Skrypt kończy się.

Utwórz plik skryptu o nazwie przekazywnie

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x przekazywnie

Aby uruchomić skrypt, wykonaj komendę:

./przekazywnie

Skrypt wypisze informacje o parametrach i ich wartościach. W wyniku otrzymasz:

Przekazano 2 parametrów

Parametr param1

Parametr param2

Podsumowując, ten skrypt definiuje funkcję funkcja_z_parametrami, która przyjmuje dwa parametry i wypisuje informacje o ilości i wartościach przekazanych parametrów. Następnie skrypt wywołuje tę funkcję, przekazując dwa parametry "param1" i "param2".

zgłoszenie 13

Zmienna specjalna $0 przechowująca nazwę skryptu nie jest dostępna!!! Choć na pierwszy rzut oka powinna przechowywać nazwę funkcji.

Interpretacja wyrażeń arytmetycznych. Kiedy zachodzi potrzeba przeprowadzenia jakichś obliczeń można skorzystać z mechanizmu interpretacji wyrażeń arytmetycznych, obliczenia dokonywane są na liczbach całkowitych, nie przeprowadzana jest kontrola przepełnienia (ang. *overflow*).

Składnia:

$((wyrażenie)) lub $[wyrażenie]

Wykonaj przykład:

```
#!/bin/bash
echo $((8/2))
wynik=$[4*5/2]
echo "$wynik"
```

Skrypt Bash wykonuje operacje arytmetyczne i wypisuje wyniki. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa polecenia echo $((8/2)) do wykonania operacji arytmetycznej dzielenia 8 przez 2 i wypisuje wynik (4).

3. Skrypt przypisuje wynik operacji arytmetycznej mnożenia 4 przez 5 i podzielenia przez 2 do zmiennej wynik.
4. Skrypt używa polecenia echo "$wynik" do wypisania wartości zmiennej wynik.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie wddz

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

```
chmod +x wddz
```

Aby uruchomić skrypt, wykonaj komendę:

```
./wddz
```

Skrypt wypisze wyniki operacji arytmetycznych:

4

10

Podsumowując, ten skrypt wykonuje dwie operacje arytmetyczne: dzielenie 8 przez 2 oraz mnożenie 4 przez 5 i podzielenie przez 2. Następnie wypisuje wyniki tych operacji.

W ten sposób (przykład poniżej) można ponumerować listę:

```
#!/bin/bash
for pliki_html in $(ls *.html)
do
numer=$((numer+1))
echo "$numer. "
echo $pliki_html
done
```

Skrypt Bash przegląda pliki z rozszerzeniem .html w bieżącym katalogu i przypisuje im numer kolejny.

Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt używa pętli for w celu przeglądania plików z rozszerzeniem .html w bieżącym katalogu.
3. W pętli, skrypt zwiększa wartość zmiennej numer o 1 za pomocą operacji numer=$((numer+1)).
4. Skrypt używa polecenia echo do wypisania numeru oraz nazwy pliku .html.
5. Skrypt kończy się.

Utwórz pliki .html:

```
touch plik{,2,3,4,5}.html
```

Utwórz plik skryptu o nazwie ponumerujhtml

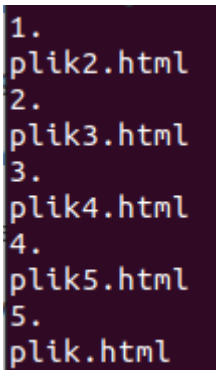
Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x ponumerujhtml

Uruchom skrypt, wykonaj komendę:

./ponumerujhtml

Skrypt wypisze numery oraz nazwy plików .html znajdujących się w bieżącym katalogu. Na przykład:



Wynikiem będzie ponumerowana lista wszystkich plików o rozszerzeniu **.html**, znajdujących się w bieżącym katalogu.

Podsumowując, ten skrypt przegląda pliki .html w bieżącym katalogu, nadając im kolejny numer oraz wypisując numer i nazwę tych plików.

zgłoszenie 14

Polecenie let - Do przeprowadzenia obliczeń można też skorzystać z polecenia **let.**

Wykonaj przykład:

```
#!/bin/bash
liczba1=5
liczba2=6
let wynik=liczba1*liczba2
echo $wynik
```

Skrypt Bash wykonuje operację mnożenia dwóch zmiennych i wypisuje wynik. Opis działania skryptu:

1. Skrypt zaczyna się od deklaracji #!/bin/bash, która wskazuje, że skrypt będzie interpretowany przy użyciu powłoki Bash.
2. Skrypt przypisuje wartość 5 do zmiennej liczba1 i wartość 6 do zmiennej liczba2.
3. Skrypt używa polecenia let wynik=liczba1*liczba2 do wykonania operacji mnożenia zmiennych liczba1 i liczba2, a wynik przypisuje do zmiennej wynik.

4. Skrypt używa polecenia echo $wynik do wypisania wartości zmiennej wynik.
5. Skrypt kończy się.

Utwórz plik skryptu o nazwie mnozenie

Nadaj prawa wykonywalności skryptowi (chmod +x nazwa_skryptu.sh):

chmod +x mnozenie

Uruchom skrypt, wykonaj komendę:

./mnozenie

Podsumowując, ten skrypt wykonuje operację mnożenia zmiennych liczba1 i liczba2, a następnie wypisuje wynik tej operacji.

Skrypt wypisze wynik operacji mnożenia zmiennych liczba1 i liczba2, który w tym przypadku wynosi 30.

zgłoszenie 15

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.

Na podstawie wersji oryginalnej kursu Bash'a - link