Podstawy informatyki

1) Skrypty powłoki - wprowadzenie

Skrypty powłoki sš programami składajšcymi się z poleceń powłoki i poleceń sterujšcych programem, takich jak : IF, FOR, REPEAT, WHILE itp.
Skrypty umożliwiajš pobranie danych od użytkownika lub z pliku, przetworzenie ich i wy&#65533;wietlenie wyniku na ekranie monitora lub zapisanie go na dowolnym urzšdzeniu.

Aby nasz skrypt mógł zostać uruchomiony należy nadać mu prawo do uruchamiania poleceniem :

chmod +x nazwa_skryptu

Każdy skrypt powinien rozpoczynać się linijkš :

#!/bin/bash
Zapis ten oznacza to, że do przetworzenie skryptu zostanie użyta powłoka bash.

Przykładowy skrypt może wyglšdać następujšco :

```
#!/bin/bash
# to jest tylko komentarz
echo 'skrypt wyswietla aktualny katalog roboczy'
echo -e "biezacy katalog :\n\a`pwd`"
```

```
#!/bin/bash
echo "informacje o uruchomionym skrypcie"
echo "nazwa skryptu : $0"
echo "ilosc parametrow : $#"
echo "pierwszy parametr to : $1"
echo "drugi parametr to : $2"
echo -e "zestawienie wszytkich parametrow :\n$*"
```

2) Skrypty powłoki - wyrażenie read

Wyrażenie **read** używane jest do pobierania danych z klawiatury i zapisywania ich w zmiennych.

Składnia wyrażenia read jest następujšca:

read zmienna1,zmienna2,..zmiennaN

Przykładowy skrypt :

```
#!/bin/bash
echo "Podaj swoje imię:"
read imie
echo "Cze&#65533;ć $imie !"
```

3) Skrypty powłoki - instrukcje sterujšce w skryptach

Bash umożliwia wykorzystanie w skryptach instrukcji sterujšcych wykonywanym skryptem takich jak :

- instrukcje warunkowe : **`if`**
- pętle : **`for , while , until`**
- wyrażenie : **`case`**

Instrukcja warunkowa `if`

`if warunek`
`then`
wykonuje wszystkie polecenia je&#65533;li warunek jest równy zero (true)
`elif warunek 2`
`then`
wykonuje wszystkie polecenia je&#65533;li warunek 2 jest równy zero (true)
`else`
wykonuje wszystkie polecenia je&#65533;li żaren z powyższych warunków nie jest spełniony
`fi`

Warunek musi mieć następujšcš postać :

```
[ wyrażenie1 operator wyrażenie2 ]
```

Między nawiasami a tre&#65533;ciš warunku muszš być postawione spacje.

Operatory wykorzystywane w warunkach :

`-eq` jest równe (= =)
`-ne` jest różne (!=)
`-lt` jest mniejsze (<)
`-le` jest mniejsze lub równe (<=)
`-gt` jest większe (>)
`-ge` jest większe lub równe (>=)

Przykładowy skrypt :

```
#!/bin/sh
if [ $1 -gt 0 ]
then
echo "liczba $1 jest dodatnia"
elif [ $1 -lt 0 ]
then
echo "liczba $1 jest ujemna"
elif [ $1 -eq 0 ]
then
echo "liczba $1 jest zerem"
else
echo "$1 nie jest liczba"
fi
```

Shelle: tcsh i bash

Pętla `for`

Podczas wykonywania pętli `for` , zmiennej zostaje przypisana każda warto&#65533;ć z `listy`, wykonujšc jednocze&#65533;nie instrukcje zawarte miedzy operatorami : do done w kolejnych etapach przypisania.

`for { nazwa zmiennej } in { lista }`
`do`
instrukcje wykonywane sš tyle razy ile jest elementów `listy`
`done`

Przykładowy skrypt :

```
#!/bin/bash
for i in 1 2 3
do
echo $i
done
```

Pętla `while` i pętla `until`

Pętla `while` wykonywana jest tak długo jak długo podany warunek jest prawdziwy.
Pętla `until` wykonywana jest tak długo jak długo podany warunek jest nie prawdziwy.
Obie pętle majš takš samš budowę.

```
while [ warunek ]
do
polecenie1
polecenie2
.......
done
```

Przykładowy skrypt :

```
#!/bin/bash
x=1;
while [ $x -le 10 ]; do
echo $x
x=$[x + 1] x= expr x + 1
done
```

Wyrażenie `case`

Wyrażenie `case` pozwala na dokonanie wyboru z po&#65533;ród kilku wzorców.

```
case zmienna in
"wzorzec1") polecenie1 ;;
```

```
"wzorzec2") polecenie2 ;;
"wzorzec3") polecenie3 ;;
*) polecenie_domylne
esac
```

Przykładowy skrypt :

```
#!/bin/bash
echo "Wybierz liczbę od 1-4 i zobacz co wygrałe◆ :"
read liczba
case "$liczba" in
"1") echo "samochód" ;;
"2") echo "dom" ;;
"3") echo "10000 zł" ;;
"4") echo "przykro mi ale nic nie wygrywasz" ;;
*) echo "aby wziść udział w losowaniu należy wybrać liczbę z
przedziału 1-4"
esac
```

funkcje:

definicja:

function nazwa_funkcji()
{
(...)
}

wywołanie:

nazwa_funkcji arg1 arg2

przykład:

function funkcja()
{
echo "to jest funkcja"
}
echo "pierwsze wywołanie funkcji:"
funkcja
echo "drugie wywołanie funkcji:"
funkcja

FTP

VI