Systemy operacyjne

wykład 4 - Instrukcja warunkowa oraz podstawowe pętle - WHILE oraz UNTIL

dr Marcin Ziółkowski

Instytut Matematyki i Informatyki
Akademia im. Jana Długosza w Częstochowie

7 kwietnia 2016 r.

Instrukcja READ

Jedną z najważniejszych komend-instrukcji w interpreterze poleceń jest polecenie READ, służące do wprowadzania danych. Dokładniej rzecz ujmując, polecenie READ pozwala na podstawienie do zmiennych wartości wprowadzonych przez użytkownika w konsoli. Oto przykład prostego skryptu, który realizuje dodawanie dwóch liczb całkowitych wprowadzonych przez użytkownika, zawierającego tą instrukcję.

SKRYPT - DODAWANIE

```
echo -n "Podaj pierwszą liczbę:"
read a
echo -n "Podaj drugą liczbę:"
read b
echo "$a + $b = $[a+b]"
```

Dodawanie bez instrukcji READ

Oto ciekawy przykład skryptu dodającego dwie liczby, które są zapisane do plików, a następnie z nich odczytywane. Nie używamy wówczas instrukcji READ.

DODAWANIE BEZ INSTRUKCJI READ - pomysł: ZUZANNA ŁAŚ - studentka 3 roku informatyki AJD

```
echo 5 > plik1
echo 25 > plik2
echo $(($(cat plik1) + $(cat plik2)))
```

Instrukcja warunkowa IF-ELIF-ELSE

Jedną z najważniejszych instrukcji sterujących w każdym języku programowania jest instrukcja warunkowa. W BASHU schemat tej instrukcji jest następujący:

INSTRUKCJA WARUNKOWA - SCHEMAT

```
if [ warunek 1 ]
then
  instrukcje
elif [ warunek 2 ]
then
  instrukcje
else
  instrukcje
fi
```

Oczywiście elementy składniowe ELIF oraz ELSE nie muszą wystąpić. Należy zwrócić uwagę na obowiązkowe spacje oddzielające warunki od nawiasów kwadratowych oraz słowa THEN oraz FI.

Instrukcja warunkowa IF-ELIF-ELSE

Oto przykład skryptu, który realizuje dzielenie dwóch liczb całkowitych i dodatkowo jest zabezpieczony przed dzieleniem przez zero z oczywistym wykorzystaniem instrukcji warunkowej.

SKRYPT - DZIELENIE

```
echo -n "Podaj pierwszą liczbę:"
read a
echo -n "Podaj drugą liczbę:"
read b
if  [ $b = 0 ]
then
  echo "Nie wolno dzielić przez zero"
else
  echo "$a : $b = $[a/b]"
fi
```

Instrukcja warunkowa IF-ELIF-ELSE

Oto kolejny przykład skryptu - tym razem sprawdzającego, czy wprowadzona liczba jest dodatnia, ujemna lub zerowa.

SKRYPT - ZNAK LICZBY

```
echo -n "Wprowadź liczbę:"
read a
if [ $a -gt 0 ]
then
  echo "Wprowadzona liczba jest dodatnia"
elif [ $a -lt 0 ]
then
  echo "Wprowadzona liczba jest ujemna"
else
  echo "Liczba jest zerem"
fi
```

Podkreślmy, że w BASHU nie używamy znaków relacyjnych $>, <$ tylko wyrażeń **-lt** oraz **-gt**.

Instrukcja warunkowa IF-ELIF-ELSE

Jednak znaczenie instrukcji warunkowej w BASHU jest dużo poważniejsze niż w innych językach programowania, ponieważ warunki nie muszą być arytmetyczne i mogą dotyczyć również atrybutów plików czy katalogów. Najczęściej używane wyrażenia to między innymi:

-ge większe lub równe
-le mniejsze lub równe
!= sprawdza czy wyrażenia są różne
-a plik istnieje
-f plik istnieje i jest plikiem zwykłym
-n wyrażenie ma długość większą niż 0
-d wyrażenie istnieje i jest katalogiem
-z wyrażenie ma zerową długość
-r można czytać plik
-w można zapisywać do pliku
-x można plik wykonać

Instrukcja warunkowa IF-ELIF-ELSE

Oto przykład znacznie ciekawszego skryptu wykorzystującego instrukcję warunkową.

SKRYPT - KASOWANIE

```
cd NOWY
if [ -a 1.txt ]
then
  rm 1.txt
else
  rm 2.txt
fi
```

Działanie tego skryptu polega na wybiórczym kasowaniu plików zawartych w określonym katalogu. Jak widać instrukcji warunkowej można użyć również do zabezpieczenia operacji przed możliwym pojawieniem się komunikatu o błędzie.

Instrukcja iteracyjna WHILE

Nie wyobrażamy sobie programowania bez instrukcji iteracyjnych, czyli tzw. PĘTLI. Podstawową pętlą w BASHU jest pętla WHILE, której składniowy schemat jest następujący:

INSTRUKCJA WHILE - SCHEMAT

```
while [ warunek kontynuacji ]
do
  instrukcje
done
```

Należy również zwrócić uwagę na spacje oddzielające warunki od nawiasów kwadratowych oraz słowa DO oraz DONE.

Instrukcja WHILE - przykłady

Oto przykład skryptu wykorzystującego instrukcję WHILE.

SKRYPT - LICZBY

```
i=10
while [ $i -gt 0 ]
do
    echo $i
    i=$[i-1]
done
```

Działanie tego skryptu polega na wypisaniu liczb całkowitych od 10 do 1.

Instrukcja WHILE - przykłady

A teraz przykład bardziej praktycznego skryptu.

SKRYPT - TWORZENIE SERII PLIKÓW

```
i=1
while [ $i -lt 10 ]
do
    touch $i.txt
    echo $i > $i.txt
    i=$[i+1]
done
```

Działanie tego skryptu polega na jednorazowym stworzeniu dziesięciu plików i wypełnieniu ich liczbami (nazwami).

Instrukcja iteracyjna UNTIL

Instrukcja UNTIL jest składniowo identyczna jak instrukcja WHILE. Jedyną różnicą jest to, że zamiast warunku kontynuacji dajemy warunek zakończenia. Instrukcje wewnątrz pętli są zatem wykonywane do momentu spełnienia warunku. Schemat jest zatem następujący:

INSTRUKCJA UNTIL - SCHEMAT

```
until [ warunek zakończenia ]
do
  instrukcje
done
```

Instrukcja UNTIL - przykłady

I przykład skryptu, który dzisiaj już był przedstawiony, tym razem użyjemy pętli UNTIL. Zauważmy, że zaprzeczeniem relacji -**gt** jest oczywiście relacja -**le**.

SKRYPT - LICZBY

```
i=10
until [ $i -le 0 ]
do
   echo $i
   i=$[i-1]
done
```

NIESKOŃCZONE PĘTLE A WIRUSY

Niekiedy używa się instrukcji iteracyjnych do tworzenia nieskończonych pętli, które są błędem programistycznym wykorzystywanym na przykład w tworzeniu WIRUSÓW komputerowych. Oto przykłady dwóch takich wirusów.

WIRUS 1

```
i=1
while [ $i -gt 0 ]
do
  echo "1" >> 1.txt
done
```

WIRUS 2

```
i=1
while [ $i -gt 0 ]
do
  mkdir "$i"
  cd "$i"
done
```