

Bash - ściągą

Położenie: (nie dotyczy)

© 3bird Projects 2008, <http://3bird.net>

Ogólne

Pliki nie mogą nazywać się "test". Na początku muszą zawierać deklarację powłoki:

```
#!/bin/bash
```

Aby każde wykonywane polecenie było widoczne, należy wpisać na początku skryptu:

```
set -x
```

Operatory

= (lub **-eq**) - jest równe

!= (lub **-ne**) - nie jest równe

-a - koniunkcja, operator logiczny AND

-o - alternatywa, operator logiczny OR

-lt - mniejsze niż

-le - mniejsze lub równe niż

-gt - większe niż

-ge - większe lub równe niż

Cudzysłowy

"Wyrażenie" - działają wewnątrz 3 znaki specjalne: **\$zmienna**, **\maskowanieZnakówSpecjalnych**;; **`polecenieJakoZmienna`**.

'Wyrażenie' - wszystko wewnątrz traktowane jest jako tekst.

`Wyrażenie` - wyrażenie traktowane jest jako polecenie, np. **`ls -af`**.

Zmienne

Przykład tworzenia zmiennych:

```
stworzonaNazwaZmiennej="Treść zmiennej"
```

Odwołanie do zmiennej:

```
$zmienna
```

Manipulacje na zmiennych:

\${jakaśZmienna#/home} - zwróci zmienną bez "/home" znajdującego się na początku.

\${jakaśZmienna%/home} - zwróci zmienną bez "/home" znajdującego się na końcu.

\${#jakaśZmienna} - jakiej długości jest zmienna.

Predefiniowane zmienne:

\$0 - ścieżka do skryptu (odpowiednik *pwd*).

\$1, \$2... - zawartość kolejnych parametrów podanych przy wywołaniu skryptu.

\$# - liczba argumentów skryptu.

\$\$ - numer procesu (*PID*) aktualnej powłoki.

\$! - numer procesu ostatnio puszczanego w tło.

Funkcje

Utworzenie funkcji:

function nazwaFunkcji

```
{  
instrukcja1  
instrukcja2  
}
```

Odwołanie do funkcji:

nazwaFunkcji

Odwołanie do funkcji w zewnętrznym pliku:

. nazwaPliku
nazwaFunkcji

Instrukcje warunkowe

poolecenie1 && polecenie2 - *poolecenie2* zostanie wykonane tylko wtedy, gdy wcześniej zostanie wykonane *poolecenie1* (gdy zwróci 0);

poolecenie1 || polecenie2 - *poolecenie2* zostanie wykonane tylko wtedy, gdy nie zostanie wykonane *poolecenie1* (wystąpi błąd, wartość różna od 0);

Przykład:

cat plik | grep -q aaa && echo "Znaleziono" || echo "Nie znaleziono"

Instrukcja if

```
if [ test ]  
then  
instrukcja1  
instrukcja2  
elif [ test ]  
then  
instrukcja3  
else  
instrukcja4  
fi
```

Zapytania test

- d** - czy plik istnieje i jest katalogiem?
- e** - czy plik istnieje?
- f** - czy plik istnieje i jest zwykłym plikiem?
- L** - czy plik istnieje i jest dowiązaniem symbolicznym?
- r** - czy plik istnieje i jest do odczytu?
- s** - czy plik istnieje i ma rozmiar większy od 0?
- w** - czy plik istnieje i jest do zapisu?
- x** - czy plik istnieje i jest wykonywalny?
- z ciągZnaków** - czy *ciągZnaków* jest równy 0?
- n ciągZnaków** - czy *ciągZnaków* jest większy od 0?
- plik1 -nt plik2** - czy *plik1* jest nowszy niż *plik2*?
- plik1 -ot plik2** - czy *plik1* jest starszy niż *plik2*?

Instrukcja case

```
case $jakaśZmienna in  
"ccc"|"aaa")  
instrukcja1 ;;  
"bbb")  
instrukcja2
```

```
instrukcja3 ;;  
*)  
instrukcjaDomyślna ;;  
esac
```

Pętle

while

```
while test  
do  
instrukcja1  
instrukcja2  
done
```

for

Wartością zmiennej staje się po kolei każdy element listy, na wszystkich wykonywana jest instrukcja:

```
for jakaśZmienna in listaElementów  
do  
instrukcja  
done
```

Czytanie

read *tworzonaZmienna* - czyta z klawiatury i przypisuje do zmiennej całą linię tekstu

read *zmienna1 zmienna2 zmienna3* - czyta z klawiatury i przypisuje do zmiennych kolejne linie tekstu.

getkey - czyta z klawiatury tylko jeden znak

Ciekawe polecenia

``cut -f 1 --delimiter=: /etc/passwd`` - ucina wyrażenia, spowoduje wyświetlenie wszystkich nazw użytkowników (i tylko tych nazw) z pliku /etc/passwd.

Ostatnia aktualizacja: 16 marzec 2008.