Programowanie skryptów bash

Zmienne Skryptowe

Wartości zmiennych są używane kiedy nazwa zmiennej jest poprzedzona znakiem $. Zdefiniowany jest szereg użytecznych zmiennych dostępnych w skryptach i shell'u. Ota kilka wybranych zmiennych:

- $$ = numer PID procesu wykonującego shell'a.
- $? = zmienna statusu zakończenia.
- $0 = nazwa polecenia użytego do wywołania programu (wykonania skryptu)
- $1 = pierwszy argument w linii polecenia.
- $2 = drugi argument w linii polecenia.
- $n = n-ty argument w linii polecenia.
- $* = wszystkie argumenty w linii polecenia.
- $# = ilość agrumentów w linii polecenia.

Polecenie "shift" może być użyte do przesunięcia argumentów w lewo, przykładowo $1 stanie się wartością $2, $3 przesunie się do $2. itd. Polecenie "shift 2" przesunie argumenty o dwie pozycje w lewo powodując że $1 będzie starą wartością $3 itd.

Przypisanie Zmiennych

Zmienne są przypisywane w skryptach (i shellu) następująco:

```
DONE=no
```

Zmienne używane są w następujący sposób:

```
while [ $DONE = no ]
```

Testowanie Warunków

Bash dostarcza funkcję o nazwie test, która zwraca wartości true (prawda) bądź false (fałsz) zależnie od wyniku tesotwanego wyrażenia. Jej składnia jest następująca

```
test expression
```

Może być ona również użyta w formie ukrytej:

```
[ expression ]
```

Poniższa lista zawiera warunki testowe dostarczone przez shell'a:

- -b file = True jeżeli plik istnieje i jest to plik blokowy.
- -c file = True jeżeli plik istnieje i jest to plik znakowy.
- -d file = True jeżeli plik istnieje i jest to katalog.
- -e file = True jeżeli plik istnieje.
- -f file = True jeżeli plik istnieje i jest to plik regularny.
- -g file = True jeżeli plik istnieje i ma ustawiony bit set-group-id (GID)
- -k file = True jeżeli plik istnieje i ma ustawiony bit "sticky".
- -L file = True jeżeli plik istnieje i jest to link symboliczny.
- -p file = True jeżeli plik istnieje i jest to nazwany potok (named pipe).
- -r file = True jeżeli plik istnieje i możliwy jest z niego odczyt.
- -s file = True jeżeli plik istnieje i jego rozmiar jest większy od zera.
- -s file = True jeżeli plik istnieje i jest to gniazdo (socket).
- -t fd = True jeżeli deskryptor pliku jest otwarty dla termiala.
- -u file = True jeżeli plik istnieje i ma ustawiony bit set-user-id (UID).
- -w file = True jeżeli plik istnieje i możliwy jest do niego zapis.
- -x file = True jeżeli plik istnieje i jest wykonywalny.
- -O file = True jeżeli plik istnieje i jest w posiadaniu użytkownika efektywnego.

- -G file = True jeżeli plik istnieje i jest w posiadaniu grupy efektywnej.
- file1 -nt file2 = True jeżeli file1 jest nowszy ze względu na datę ostatniej modyfikacji niż file2.
- file1 ot file2 = True jeżeli file1 jest starszy niż file2.
- file1 ef file2 = True jeżeli file1 oraz file2 umieszczone są na tym samym urządzeniu i mają te same numery i-węzłów.
- -z string = True jeżeli długość string jest równa zero.
- -n string = True jeżeli długość string jest różna od zera.
- string1 = string2 = True jeżeli łańcuchy są równe.
- string1 != string2 = True jeżeli łańcuchy ne są równe.
- !expr = True jeżeli wyrażenie expr redukuje się w fałsz.
- expr1 -a expr2 = True jeżeli zarówno expr1 oraz expr2 są prawdziwe.
- expr1 -o expr2 = True jeżeli albo expr1 lub expr2 lub oba są prawdziwe.

Możliwe jest również użycie wyrażeń o następującej składnii:

```
arg1 OP arg2
```

gdzie OP jest jednym z: -eq, -ne, -lt, -le, -gt lub -ge. Argumenty arg1 oraz arg2 mogą być wartościami całkowitymi ze znakiem lub wyrażeniem specjalnym "-l string", które oblicza długość łańcucha string.

Sterowanie i Pętle Iteracyjne

- if - instrukcja stosowana do wykonania jedengo lub więcej działań zależnie do warunku. Przykład:

```
if [ ! -d /mnt ]                # upewnij sie ze katalog /mnt istnieje
then
    mkdir /mnt
fi
```

- case - isntrukcja stosowana do wykonania określonych działań w zależności od wartości zmiennej. Przykład:

```
case $NUM
1)
    echo Liczba równa się 1
    ;;
2)
    echo Liczba równa się 2
    ;;
*)
    echo Liczba jest różna od 1 oraz 2
    ;;
esac
```

- for - instrukcja używana do wykonania pętli dla wszystkich przypadków warunku. W przykładzie poniżej użyta została do skopiowania wszystkich plików z katalogu /mnt/floppy do katalogu /etc. Linie zostały ponumerowane w celu odniesienia do opisu:
    1. Pętla będzie się wykonywać do chwili aż wszystkie pliki zostaną odszukane.
    2. Warunek sprawdza czy odszukany plik jest normalnym plikiem regularnym a nie katalogiem.
    3. Linia komentarza.
    4. Linia wydobywa nazwę pliku z pełnej ścieżki wskazywanej przez zmienną $i a następnie umieszcza wydobytą nazwę w zmiennej $filename. Użyta tutaj metoda jest nazywana wyrażeniem parametrycznym (parameter expression) i jest ona udokumentowana na stronie podręcznika man dotyczącej bash'a. Więcej informacji o wyrażeniach parametrycznych znajdziesz w "Linux Programmer's Guide".
    5. Linia ta drukuje na standardowym wyjściu napis informujący jaki plik jest aktualnie kopiowany.
    6. Linia ta wykonuje kopiowanie używając opcji -p w celu zachowania atrybutów pliku. Przypis: Sprawne pisanie programowania skryptów wymaga znajomości szeregu poleceń, programów i narzędzi, co jest równie ważne jak znajomość składni. Jest to oczywiste dla każdego kto czytał scrypty startowe w /etc/rc.d oraz powiązanych z nim katalogami.
    7. Linia ta kończy blok instrukcji if.

8. Linia ta kończy blok instrukcji for.

```
1. for i in /mnt/floppy/*; do
2.         if [ -f $i ]; then
3.                 # if znaleziono plik regularny
4.                 filename=${i#/mnt/floppy/}
5.                 echo kopiowanie $i do /etc/$filename
6.                 cp -p $i /etc/$filename
7.         fi
8. done
```

- until - wykonuje pętle do chwili aż spełniony zostanie pewien warunek. Składnia jest nastepująca:

```
until [ expression ]
do
   statements
done
```

- while - wykonuje pętle tak długo, jak długo warunek jest spełniony. Poniższa pętla jest nieskończona:

```
while [ 1 ]
do
   statement(s)
done
```

Popularne Narzędzia Skryptowe

Lista wybranych programów przydatnych w skryptach: date, expr, find, cd, ls, pwd, mkdir, rmdir, cat, cp, csplit, ln, mv, rm, split, awk, cut, diff, grep, head, line, sed, tail, uniq, touch, wc, join, paste, sort, tr, pr, basename, dirname, kill, killall Zobacz również stronę podręcznika man textutils omawiającą narzędzia:

Wypisywanie zawartości plików:

- cat - łączenie i wypisywanie plików
- tac - łączenie i wypisywanie odwróconych plików
- nl - numerowanie linii i wypisywanie plików
- od - wypisywanie plików w formacie ósemkowym i innych

Formatowanie zawartości plików:

- fmt - reformatowanie akapitów tekstu
- pr - stronicowanie i kolumnowanie plików do wydruku
- fold - zawijanie linii wejściowych do zadanej szerokości

Wypisywanie części plików:

- head - wypisywanie początku plików
- tail - wypisywanie końca plików
- split - podział pliku na części stałej wielkości
- csplit - podział pliku na części zależne od kontekstu

Statystyka plików:

- wc - wypisywanie liczby bajtów, słów i linii
- sum - wypisywanie sumy kontrolnej i liczby bloków
- csum - wypisywanie sumy CRC liczby bloków
- md5sum - wypisywanie lub sprawdzanie skrótu danych (patrz też sha1sum).

Sortowanie i operacje na plikach posortowanych:

- sort - sortowanie plików tekstowych
- uniq - pozostawianie unikalnych linii w pliku
- comm - porównywanie dwu posortowanych plików liniami
- ptx - tworzenie indeksu permutacyjnego zawartości pliku
- tsort - sortowanie topologiczne

Operacje na polach wewnątrz linii:

- cut - wypisywanie wybranych części linii
- paste - zlepianie linii plików
- join - łączenie linii według wspólnego pola

Operacje na znakach:

- tr - zamiana, ściskanie, usuwanie znaków
- expand - zamiana tabulacji na spacje
- unexpand - zamiana spacji na tabulacje

Możliwości Powłoki

Przekierowania

- < = otwarcie pliku jako stdin, przykładowo: wc < plik.txt
- > = otwarcie jako stdout, przykładowo: ls > lista.txt
- >> = dopisanie do pliku, przykładowo: badblocks /dev/hde1 >> uszkodzonebloki.txt.
- | = połączenie w potok (pipe) stdout jednego programu z stdin drugiego, przykładowo: ls | sort