Skrypty powłoki
Są to wykonywalne pliki tekstowe, którymi można automatyzować czynności wykonywane w powłoce. Zawierają one polecenia powłoki i instrukcje sterujące. Skrypty można edytować za pomocą dowolnego edytora tekstu (nano, pico, vi, itp.)

Pierwsza linia pliku powinna zawierać „zakomentowane" polecenie uruchamiające wybraną powłokę. U nas będzie to bash, więc pierwsza linia powinna wyglądać:

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

Hello world:

```
#!/bin/bash
echo "Hello, World"
```

Aby plik uruchomić, należy przyznać mu prawa do uruchomienia!

Zmienne:
Najczęściej w skryptach wykorzystujemy zmienne lokalne, które ustawiamy po prostu przez przyrównanie. np.

```
i=1
```

Jeżeli skrypt wywołuje inny skrypt, i w nim też chcemy korzystać ze zmiennych, musimy użyć zmiennej globalnej:

```
export i
```

Ale jeżeli odwołujemy się do takiej zmiennej, sam łańcuch tekstowy „i" czy „zmienna" byłby traktowany jako parametr lub wywołanie programu. Należy użyć znaku $:

```
echo $i
echo $zmienna
```

lub, jeszcze lepiej:

```
echo ${i}
echo ${zmienna}
```

Istnieją zmienne, które występują zawsze w programie:

$0 – nazwa skryptu
$1, $2 … - kolejne parametry przekazane do skryptu.
$# - liczba parametrów przekazanych do skryptu
$* - wszystkie parametry jako jeden ciąg
$? - status ostatniego polecenia (w uproszczeniu odpowiednik windowsowego ERRORLEVEL).

Program echo
W SunOS/Solaris oraz niektórych dystrybucjach GNU, aby móc korzystać ze znaków escape'owanych (\n, \r, \t itp.) musimy użyć parametru -e.

Linię zakomentowujemy znakiem #.

Cudzysłów i apostrof:

Cudzysłów podwójny - ” - cały ciąg znaków rozszerzony o zmienną, np.

```
i=4
echo ”zmienna:\t ${i};”
```

wypisze nam

```
zmienna:  4;
```

Gdybyśmy nie użyli znaków ”, nie moglibyśmy użyć ani \t ani tym bardziej znaku specjalnego ;

Apostrof - ‘ - cały ciąg znaków dosłownie. Rozwijane są tylko białe znaki (\t, \n itp.).
Tak więc powyższy przykład z użyciem ‘ dałby efekt:

```
zmienna:  $i;
```

Apostrof odwrócony (grawis) - ` - za ciąg znaków podstawiany jest wynik polecenia ujętego w odwrócony apostrof.
Np.

```
pliki=`ls -l ~/ | wc -l`
echo ”Ilosc elementow: ${pliki}”
```

Arytmetyka

Do realizacji działań matematycznych służy program expr. Kolejne jego parametry to kolejne elementy działania, a wynik wyprowadzany jest jako wyjście. Stąd np.

```
zmienna=`expr $i + 2`
```

Podstawowe operatory: +, -, *, /, %

Do arytmetyki zmiennoprzecinkowej można użyć rozszerzonego parsera równań **bc**. Program ten przyjmuje jednak dane z pliku, nie jako parametry. Można to jednak ominąć używając przekierowań:

```
echo ”2 / 4” | bc -l
```

Wyświetli .50000000000000000000
Parametr -l wyświetla liczby z długą precyzją (zamiast domyślnego 0 miejsc po przecinku).

Jest to **najbardziej kompatybilny zestaw do arytmetyki**. Praktycznie każda powłoka go posiada. W zasadzie dopóki użytkownik nie używa csh przedstawione wyżej rozwiązanie zadziała. Użytkownik może nie mieć Basha i uruchamiać skrypt w innej powłoce - z reguły zadziała. Stąd wciąż to podejście jest stosowane.

W powłoce Bash, o ile jesteśmy pewni, że użytkownik uruchomi skrypt powłoki właśnie w Bashu (a we współczesnych systemach często tak jest), możemy użyć dwóch charakterystycznych dla Basha konstrukcji:

```
a=3
let ”a = $a + 2”
echo $a
```

Druga możliwość to:

```
a=3
a=$(($a+2))
echo $a
```

Tutaj możemy sobie pozwolić na spacje pomiędzy operatorami lub nie.

Pętla for

```
for zmienna in lista; do
  . . .
done
```

Albo bez średnika:

```
for zmienna in lista
do
 . . .
done
```

Lista może być zdefiniowana jako np. wybór plików za pomocą wieloznaczników, zestaw ciągów znaków dzielonych spacjami, zestaw parametrów programu, czy zestaw znaków ({1,2,3}). Spacja ma tu wyższy priorytet niż przecinek.

W nowszych wersjach narzędzi mamy program **seq**, który sam wygeneruje sekwencję, dzięki temu można posiłkować się takim podejściem:

```
for i in `seq 1 8`
do
 echo $i
done
```

Wyświetli nam w kolejnych linijkach 1, 2, 3, 4, 5, 6, 7, 8.

Zadania:

Napisz skrypt przyjmujący w parametrach ciąg liczb całkowitych, a wyprowadzający na ekran ich średnią arytmetyczną.