

## Zajęcia 6

**Ćwiczenie 0.** Napisz skrypt, który sam sobie nada uprawnienia do zapisu (o ile tych uprawnień nie miał)

i zabierze uprawnienia do zapisu (jak tych uprawnień miał).

**Ćwiczenie 1.** Napisz skrypt, w który wyświetla wszystkie swoje argumenty w odwrotnej kolejności.

**Ćwiczenie 2.** Napisz skrypt, w którym użytkownik podaje numer argumentu, który ma być wyświetlony na ekranie, a następnie wyświetlamy ten argument.

**Ćwiczenie 3.** Napisz skrypt, który policzy ilość linii w plikach z rozszerzeniem java w wybranym katalogu (argument skryptu)

**Ćwiczenie 4.** Napisz skrypt, który zapisze do pliku środkowe linie z wszystkich plików z bieżącego katalogu.

**Ćwiczenie 5.** Napisz skrypt, który wyświetli silnie z liczby a, która jest wprowadzona z klawiatury. Liczba a musi być dodatnia.

**Ćwiczenie 6.** Napisz skrypt, który wyświetli wszystkie silnie od liczby a do liczby b. Liczby a, b muszą być dodatnie.

**Ćwiczenie 7.** Napisz skrypt, który sprawdzi, czy podany argument jest nazwą katalogu i jak tak to go wyświetli wraz ze wszystkimi podkatalogami i policzy rozmiar jego rozmiar w bajtach. Dodatkowo, nazwy wszystkich plików lub katalogów z ustawionym atrybutem wykonywania dla właściciela powinny zostać zapisane do pliku o nazwie, która zostanie podana jako drugi argument. Jeżeli drugiego argumentu nie ma to skrypt ma nic nie robić.

Przydatne funkcje: `expr`, `while`, `read`, `>`, `<`, `test`, `set`

**Ćwiczenie 8.** Mamy plik o zawartości jak poniżej:

```
100 300
20 10
11 0
55
11 11a11
333 3
```

Każda linia zawsze składa się z dwóch kolumn. W każdej kolumnie mamy dwa teksty oddzielone spacją. Napisz skrypt, który pobierze nazwę takiego pliku i wykona operacje matematyczne (+-\*/) na kolejnych parach liczb odczytanych z naszego pliku. Wyniki mają zostać zapisane do pliku, którego nazwa jest przekazana jako drugi parametr i dodatkowo wyświetlone na ekran. Trzeba wyświetlić informacje o błędach - dlaczego zadanej linii nie możemy przetworzyć, o ile taka sytuacja wystąpiła.

**Ćwiczenie 9.** Mamy plik jak poniżej:

```
1 2 3 4 5
4 3 2 3 4 5 4 3
```

```
2 2 2 2 2 3 3 4 4 4 5 5 5
d 3 4 3 5 4 5 3
```

Napisz skrypt, który pobierze nazwę takiego pliku jako pierwszy argument i wyznaczy sumę (opcja -s) tych liczb i/lub iloczyn (opcja -i) tych liczb.

Przydatne funkcje: for, while, printf

**Ćwiczenie 10.** Napisz skrypt, który wyświetli tabelkę jak poniżej:

```
1 2 3 4 5 6 7 8 9 10 11 12
2 4 6 8 10 12 14 16 18 20 22 24
3 6 9 12 15 18 21 24 27 30 33 36
4 8 12 16 20 24 28 32 36 40 44 48
5 10 15 20 25 30 35 40 45 50 55 60
6 12 18 24 30 36 42 48 54 60 66 72
7 14 21 28 35 42 49 56 63 70 77 84
8 16 24 32 40 48 56 64 72 80 88 96
9 18 27 36 45 54 63 72 81 90 99 108
10 20 30 40 50 60 70 80 90 100 110 120
11 22 33 44 55 66 77 88 99 110 121 132
12 24 36 48 60 72 84 96 108 120 132 144
```

## Helpik

```
test
-z string True if the length of string is zero.
-n string True if the length of string is non-zero.
```

```
STRINGS CMP string1 = string2
True if the strings are equal.
string1 != string2
True if the strings are not equal.
```

```
! expr True if expr is false.
expr1 -a expr2 True if both expr1 AND expr2 are true.
expr1 -o expr2 True if either expr1 OR expr2 is true.
```

```
NUMBERS CMP arg1 OP arg2
OP is: -eq, -ne, -lt, -le, -gt, -ge.
```

These arithmetic binary operators return true if arg1 is equal, not-equal, less-than, less-than-or-equal, greater-than, or greater-than-or-equal than arg2, respectively. Arg1 and arg2 may be positive integers, negative integers, or the special expression -l string, which evaluates to the length of string.

## File operators

-b FILE exists and is block special

-c FILE  
FILE exists and is character special

-d FILE  
FILE exists and is a directory

-e FILE  
FILE exists

-f FILE  
FILE exists and is a regular file

-g FILE  
FILE exists and is set-group-ID

-G FILE  
FILE exists and is owned by the effective group ID

-h FILE  
FILE exists and is a symbolic link (same as -L)

-k FILE  
FILE exists and has its sticky bit set

-L FILE  
FILE exists and is a symbolic link (same as -h)

-O FILE  
FILE exists and is owned by the effective user ID

-p FILE  
FILE exists and is a named pipe

-r FILE  
FILE exists and read permission is granted

-s FILE  
FILE exists and has a size greater than zero

-t FD file descriptor FD is opened on a terminal

-u FILE  
FILE exists and its set-user-ID bit is set

-w FILE  
FILE exists and write permission is granted

-x FILE  
FILE exists and execute (or search) permission is granted

Dodatkowe, aczkolwiek bardzo ważne “elementy”, bez których praktycznie żaden skrypt nie byłby w stanie istnieć względnie działać J

\$0, \$1, \$2, \$3, ....., \$#, \$@, \$\*, \$?

```
#Pętla for
#!/bin/bash
```

```
for i in `seq 1 10`;
do
    echo $i
done
```

```
#Pętla while
#!/bin/bash
```

```
COUNTER=0
while [ $COUNTER -lt 10 ]; do
    echo "The counter is $COUNTER"
    let COUNTER=COUNTER+1
done
```

```
#Pętla until
#!/bin/bash
```

```
COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo "COUNTER = $COUNTER"
    let COUNTER-=1
done
```

```
#Znajoma pętla for
#!/bin/bash
read N
for((k=1; $k<=$N; k++)) ;
do
```

```
for((i=1; $i<=$k; i++)) ;  
do  
    ...  
done  
done
```

```
#Odczyt linia po linii z pliku  
#!/bin/bash
```

```
PLIK="ala.txt"  
cat "$PLIK" | while read LINIA  
do  
    echo -n $LINIA | wc -c  
done
```

```
#Odczyt linia po linii z pliku  
#!/bin/bash
```

```
PLIK="ala.txt"  
while read LINIA  
do  
    echo -n $LINIA | wc -c  
done < "$PLIK"
```

```
#ot ciekawostka  
#!/bin/bash
```

```
LINIA ="10 20"  
set $LINIA  
expr $1 + $2
```