C2110 *Operační systém UNIX a základy programování*

10. lekce / modul 2

PS/2020 Distanční forma výuky: Rev4

Petr Kulhánek

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kamenice 5, CZ-62500 Brno

Obsah

AWK

- K čemu slouží jazyk AWK?
- Struktura skriptu, průběh vykonávání
- Struktura bloku, regulérní výrazy, spuštění skriptů
- Proměnné, operace nad proměnnými
- Formátovaný a neformátovaný výstup

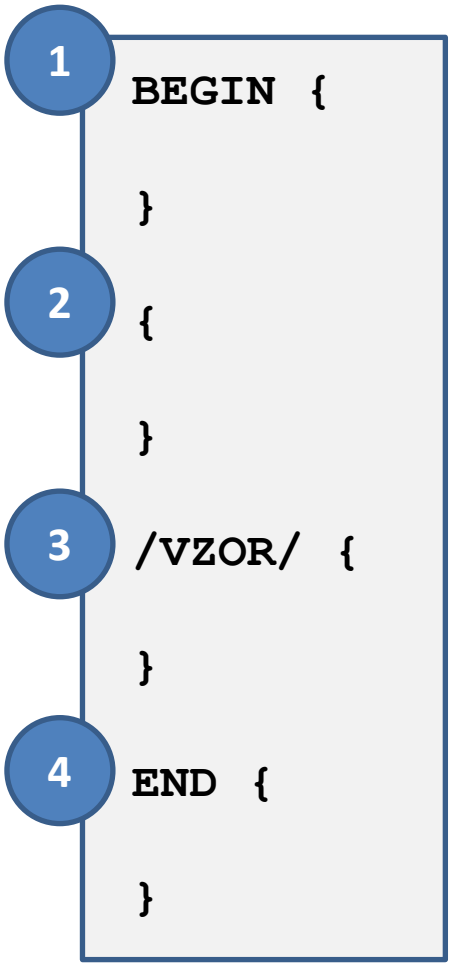
AWK

http://www.gnu.org/software/gawk/gawk.html

AWK je skriptovací jazyk navržený pro **zpracovávání textových dat**, ať už v podobě textových souborů nebo proudů. Jazyk využívá **řetězcové datové typy**, **asociativní pole** (pole indexovaná řetězcovými klíči) a **regulární výrazy**.

adaptováno z www.wikipedia.org

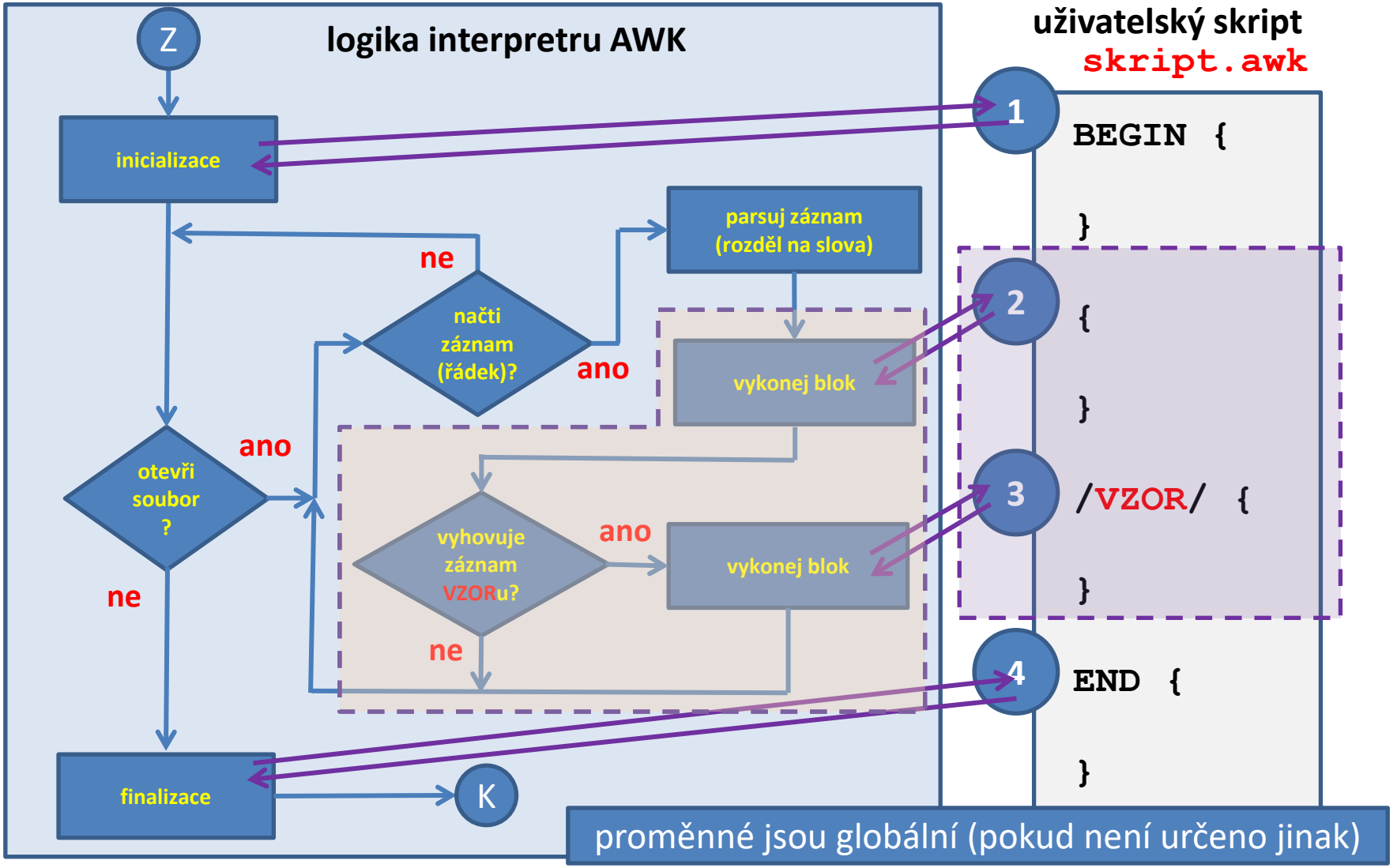
Průběh vykonávání skriptu



- Blok BEGIN (1) se vykoná (pokud je ve skriptu obsažen) před analýzou souboru.
  - Načte se záznam ze souboru. Ve výchozím nastavení je záznamem celý řádek analyzovaného souboru nebo proudu. Záznam se rozdělí na pole. Ve výchozím nastavení jsou pole jednotlivá slova v záznamu.
  - Pro daný záznam se vykoná blok (2).
  - Pokud záznamu vyhovuje VZORu, vykoná se blok (3).
  - .... vykonají se případně další bloky ....
- Blok END (4) se vykoná (pokud je ve skriptu obsažen) po analýze celého souboru.

Průběh vykonávání skriptu

```
awk -f skript.awk soubor1.txt soubor2.txt
```

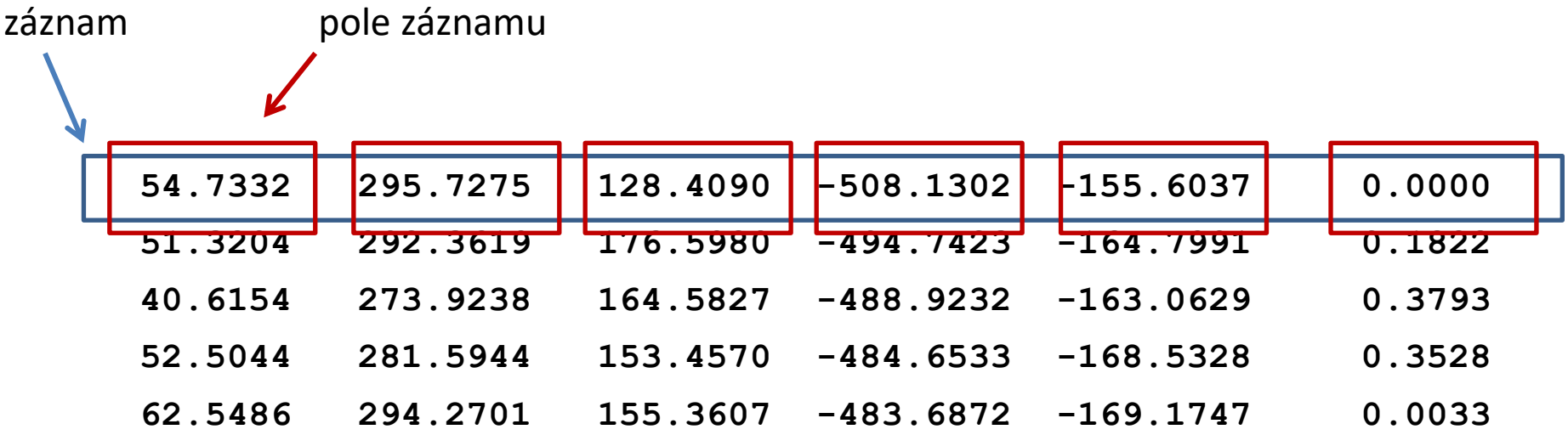


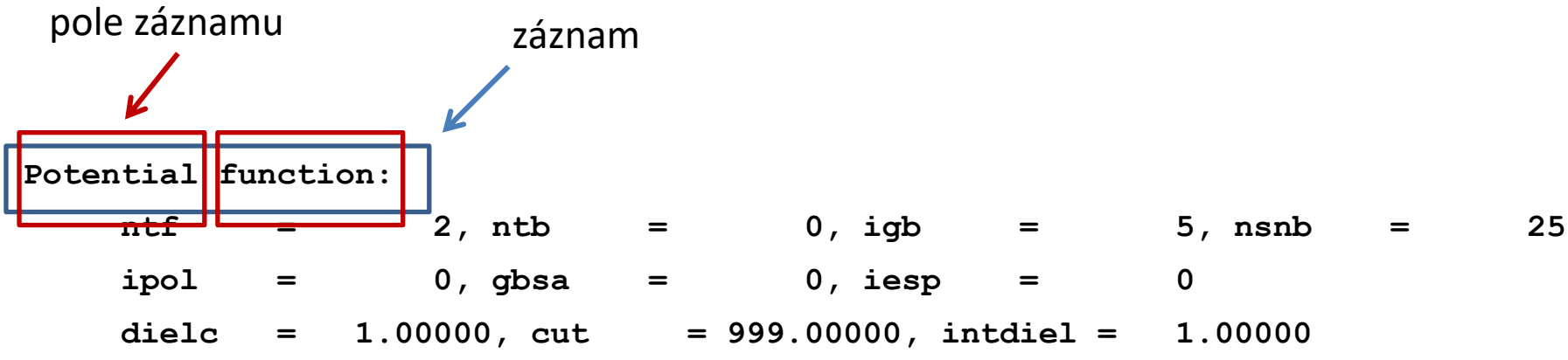
Analýza textových souborů

```
   54.7332   295.7275   128.4090  -508.1302  -155.6037     0.0000
   51.3204   292.3619   176.5980  -494.7423  -164.7991     0.1822
   40.6154   273.9238   164.5827  -488.9232  -163.0629     0.3793
   52.5044   281.5944   153.4570  -484.6533  -168.5328     0.3528
   62.5486   294.2701   155.3607  -483.6872  -169.1747     0.0033
```

```
Potential function:
     ntf     =       2, ntb     =       0, igb     =       5, nsnb    =      25
     ipol    =       0, gbsa    =       0, iesp    =       0
     dielc   =   1.00000, cut     = 999.00000, intdiel =   1.00000
```

Analýza textových souborů





Analýza textových souborů

```
  54.7332   295.7275   128.4090  -508.1302  -155.6037     0.0000
  51.3204   292.3619   176.5980  -494.7423  -164.7991     0.1822
  40.6154   273.9238   164.5827  -488.9232  -163.0629     0.3793
  52.5044   281.5944   153.4570  -484.6533  -168.5328     0.3528
  62.5486   294.2701   155.3607  -483.6872  -169.1747     0.0033
```

```
Potential function:

     ntf    =       2, ntb    =       0, igb    =       5, nsnb   =      25
     ipol   =       0, gbsa   =       0, iesp   =       0
     dielc  =   1.00000, cut    = 999.00000, intdiel =   1.00000
```

Ukázka

vstup.txt

```
  54.7332   295.7275   128.4090   -508.1302   -155.6037     0.0000
  51.3204   292.3619   176.5980   -494.7423   -164.7991     0.1822
  40.6154   273.9238   164.5827   -488.9232   -163.0629     0.3793
  52.5044   281.5944   153.4570   -484.6533   -168.5328     0.3528
  62.5486   294.2701   155.3607   -483.6872   -169.1747     0.0033
```

script.awk

```
{
    print $2;
}
```

jeden jednoduchý blok

```
$ awk -f script.awk vstup.txt
```

nebo

```
$ awk '{ print $2; }' vstup.txt
```

```
295.7275
292.3619
273.9238
281.5944
294.2701
```

Struktura bloku, příklad

```
# blok pocita mezisoucet druheho sloupce
# a mezisoucet ctvrteho sloupce, pokud je ve tretim
# sloupci hodnota 5
{
    # toto je komentar
    f = f + $2; # tady pocitam mezisoucet
    printf("Mezisoucet je %10.3f\n",f);
    if( $3 == 5 ) {
        k = k + $4; # mezisoucet pro ctvrty sloupec
    }
}
# blok pocita kumulativní sumu teploty (paty sloupec)
# na radcich obsahujicich klicove slovo "TEMP"
/TEMP/ {
    temp = temp + $5;
}
```

- komentáře jsou uvozeny znakem #
- příkazy se uvádějí na samostatné řádky, které je vhodné ukončit středníkem
- středník je nutné použít, pokud uvádíme dva a více příkazů na jeden řádek

VZOR - Regulární výrazy

```
/VZOR/  {

}
```

Pokud záznamu vyhovuje VZOR, tak se blok vykoná.

Vzor je **regulární výraz**.

Regulární výraz je jazyk, který popisuje strukturu textového řetězce. Jazyk se využívá k vyhledávání textových řetězců, k nahrazování části řetězců.

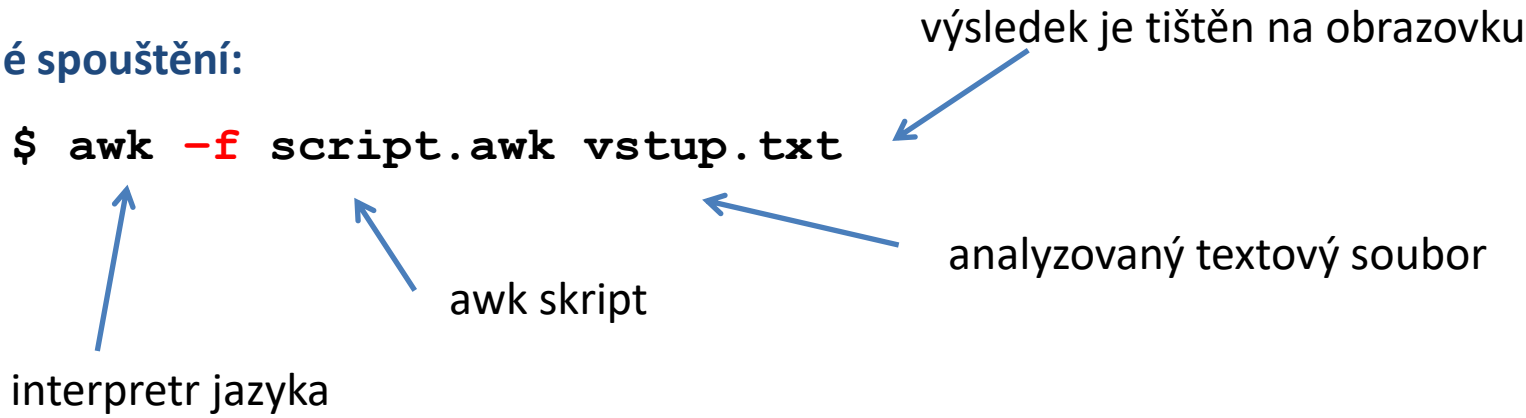
Příklady jednoduchých regulárních výrazů:

TEXT - je splněno, pokud je v daném záznamu obsažen TEXT (může být kdekoliv)
^TEXT - je splněno, pokud je v daném záznamu obsažen TEXT na začátku
TEXT$ - je splněno, pokud je v daném záznamu obsažen TEXT na konci

Spouštění AWK skriptů

Zpracování textového souboru:

Nepřímé spouštění:



Analyzovaná data lze zaslat přes standardní vstup:

```
$ awk -f script.awk < vstup.txt

$ cat soubor.txt | awk -f script.awk
```

Cvičení 1

1. Vytvořte adresář awk-data.
2. Do adresáře awk-data zkopírujte soubory matice.txt, produkt.log a rst.out z adresáře /home/kulhanek/Documents/C2110/Lesson10.
3. Napište skript, který vytiskne druhý sloupec ze souboru matice.txt.
4. Napište skript, který vytiskne druhý a čtvrtý sloupec ze souboru matice.txt.

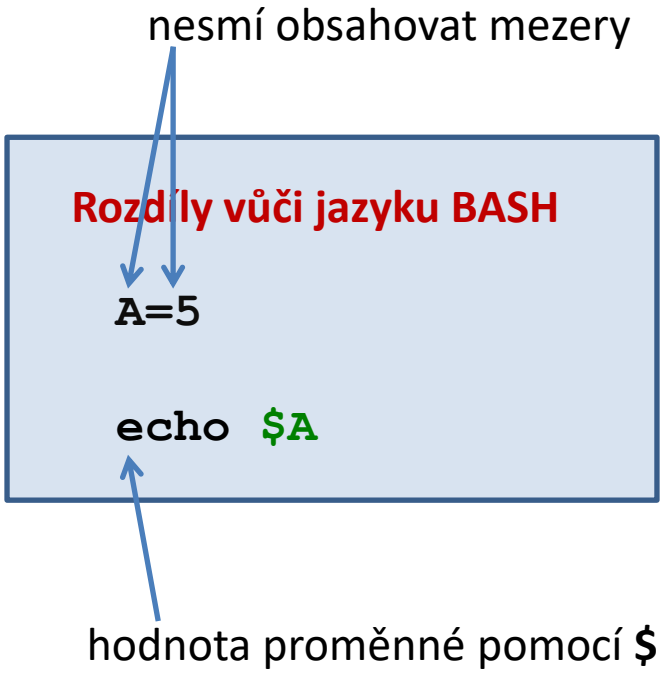
Proměnné

Přiřazení do proměnné:

```
A = 10;
B = "toto je text"
C = 10.4567;
D = A + C;
```

Hodnota proměnné:

```
print A + C;
print B;
```



Speciální proměnné:

NF	počet polí v aktuálně prováděném záznamu (Number of Fields)
NR	pořadí prováděného záznamu (Number of Records)
FS	oddělovač polí v záznamu (Field Separator), **výchozí je mezera a tabulátor**
RS	oddělovač záznamů (Record Separator) , **výchozí je znak nové řádky \n**
$0	celý záznam
$1, $2, $3 ...	jednotlivé pole záznamu

Proměnné, ...

$0	celý záznam
$1, $2, $3 ...	jednotlivé pole záznamu

znak **$** umožňuje programový přístup k jednotlivým polím záznamu

Příklad:

```
i=3;
print $i;
```

vytiskne hodnotu pole určeného hodnotou proměnné *i*

Matematické operace

Pokud lze proměnnou interpretovat jako číslo, lze použít následující aritmetické operátory:

++ hodnotu proměnné zvýší o jedničku

```
A++;
```

-- hodnotu proměnné sníží o jedničku

```
A--;
```

+ sečte dvě hodnoty

```
A = 5 + 6;
A = A + 1;
```

- odečte dvě hodnoty

```
A = 5 - 6;
A = A - 1;
```

***** vynásobí dvě hodnoty

```
A = 5 * 6;
A = A * 1;
```

/ vydělí dvě hodnoty

```
A = 5 / 6;
A = A / 1;
```

+= k proměnné přičte hodnotu

```
A += 3;
A += B;
```

-= od proměnné odečte hodnotu

```
A -= 3;
A -= B;
```

***=** proměnnou vynásobí hodnotou

```
A *= 3;
A *= B;
```

/= proměnnou podělí hodnotou

```
A /= 3;
A /= B;
```

Příkaz print

Příkaz **print** slouží k neformátovanému vypisování řetězců a čísel.

Syntaxe:

```
print hodnota1[,] hodnota2[,] ...;
```

pokud jsou hodnoty oddělené čárkou, ve výstupu se hodnoty oddělí mezerou

Příklady:

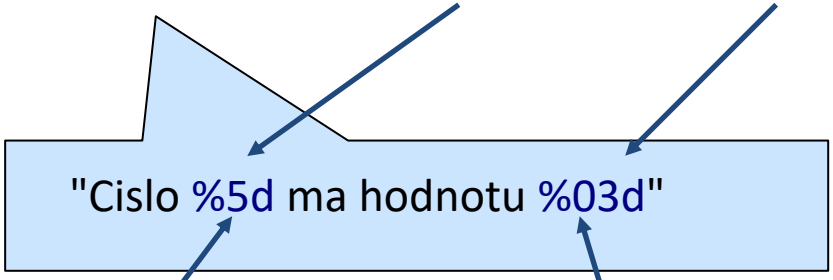
```
i = 5;
k = 10.456;
j = "hodnota promenne i =";
print j, i;
print "hodnota promenne k =", k;
```

Funkce printf

Funkce **printf** slouží k vypisování formátovaných textů a čísel.

Syntaxe:

```
printf("format", hodnota1, hodnota2, ...);
```



do tohoto místa vlož **hodnotu2** v daném formátu

do tohoto místa vlož **hodnotu1** v daném formátu

Rozdíl vůči jazyku BASH:

```
printf [format] [hodnota1] [hodnota2] ...
```

příkaz

argumenty příkazu se oddělují mezerou

Cvičení 2

- Napište skript, který sečte čísla v druhém sloupci souboru matice.txt.
- Napište skript, který vytiskne počet řádků, které obsahuje soubor matice.txt. Výsledek ověřte pomocí příkazu wc.

Samostudium

Spouštění AWK skriptů, ...

Přímé spouštění

```
$ ./script.awk vstup.txt

$ ./script.awk < vstup.txt

$ cat soubor.txt | ./script.awk
```

Skript `script.awk` **musí** mít nastaven příznak x (**executable**) a interpreter AWK (součást skriptu).

```
#!/usr/bin/awk -f
{
    i += NF;
}
END {
    print "Pocet slov je:", i;
}
```

Tento způsob spouštění nedoporučuji používat.