Számítógép Architektúrák

9. Gyakorlat

AWK 2

Az awk beépített függvényei

- **length (s)**: Az s karaklerlánc hossza,
- **substr(s,m,n)**: Az s mezőben levő karakterláncnak az m-edik karakterétől kezdődő, n darab karaktert vágja ki. Ha az n-et elhagyjuk, akkor az m-ediktől az utolsóig írja ki a karakterláncot. (nem módosítja az eredetit)
- **split (s, a, c)**: Az S-t c karakter szerint a[l]... a[n] mezőkre bontja, visszaadja n-et. Ha nem adunk meg c-t, akkor az FS értékét használja.
- **index (s1, s2)**: Az s2 helyzete az s1 karakterláncban. 0-t ad vissza, ha nincs benne.
- **sprintf (fmt, ...)**: ...-ot az fmt szerint formázza
- **sin (kif)**: A kif szinusza. (radiánban)
- **cos (kif)**: A kif koszinusza. (radiánban)
- **exp (kif)**: A kif exponenciális függvénye. (e^{kif})
- **log (kif)**: A kif természetes logaritmusa.
- **int (kif)**: A kif egészrésze.
- **getline ( )**: A következő bemenő sort olvassa, 0-t ad vissza, ha fájl vége, 1-et, ha nem.
- **tolower()**: kisbetűssé alakítva adja vissza a stringet (nem módosítja az eredetit)
- **toupper()**: nagybetűssé alakítva adja vissza a stringet (nem módosítja az eredetit)
- **sub**(regexp,replstring,mystring): karaktersorozatot cserél mystringben
- **gsub**(regexp,replstring,mystring): karaktersorozatokat cserél mystringben

Üres példaprogram

```
#! /bin/awk

BEGIN {
    mystring="How are you doing today?"
    honapok="Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec"
}

{

}

END {

}
```

length(kif)

- Az s karaklerlánc hossza. Ha elmarad a „kif”, akkor a teljes sor hosszát adja vissza.

```
END {
  print length(mystring)
}
```

```
$ cat valami_bemenet | awk -f awk_prg
> 24
```

index(s1,s2)

Az s2 helyzete az s1 karakterláncban. 0-t ad vissza, ha nincs benne.

```
END {
  print index(mystring,”you”)
}
```

```
> 9
```

tolover(s1) toupper(s1)

- Nagy illetve kisbetűssé alakítva adja vissza az s1 karakterláncot. Az eredetit nem módosítja!

```
END {
  print tolower(mystring)
  print toupper(mystring)
}
```

```
> how are you doing today?
> HOW ARE YOU DOING TODAY?
```

substr(mystring,startpos,maxlen)

- mystring: vagy string tartalom, vagy string vált.
- startpos: kezdő karakter pozíciója
- maxlen: kivágni kívánt szöveg hossza
- Ha **(startpos + maxlen) > length(mystring)** akkor az eredményt csonkolva kapjuk.

```
END {
  print substr(mystring,9,3)
}

> you
```

sin(kif)

- Trigonometrikus függvények. A „kif” függvényértékét adják vissza radiánban.

```
END {
  print sin(180)
  print (length(mystring))
}
```

```
> -0.801153
> -0.905578
```

exp(kif)

- A „kif” exponenciális függvénye. (e^{kif})

```
END {
  print exp(0)
  print exp(1)
}
```

```
> 1
> 2.71828
```

log(kif)

- A „kif” természetes alapú logaritmusát adja vissza.

```
END {
  print exp(0)
  print exp(1)
}

> 0.999999
> 1
```

int(kif)

- A „kif” egész részét adja vissza.

```
END {
  print int(3.1415)
}

> 3
```

[g]sub(regexp,replstring,mystring)

- mystring: ebben a karaktersorozatban keres
- regexp: raguláris kifejezésnek megfelelő mintát
- replstring: erre cseréli

- sub(…): CSAK az első egyezőt
- gsub(…) Az összeset.

- MÓDOSÍTJA mystring eredeti tartalmát!!!

[g]sub(regexp,replstring,mystring)

```
END {
  sub(/o/, "O", mystring)
  print mystring
  mystring="How are you doing today?"
  gsub(/o/, "O", mystring)
}
```

> HOw are you doing today?
> HOw are yOu dOing tOday?

Vezérlési szerkezetek

- **if**

```
if  (feltetel)
        utasitas1
else
        utasitas2
```

- **for**

```
for (kif1;feltetel;kif2)
        utasitas
```

for(i = 1; i<=NF; i++) //a ciklus végigfut az 1, 2, ... értékeken a mezők számáig.

Vezérlési szerkezetek

- **while**

```
kif1
while (feltetel) {
        utasitas
        kif2
}
```

Tömbök

- ugyanúgy, mint a változókat nem kell előre deklarálni;
- egy tömb méretét csak a gépben elérhető tárterület korlátozza.

- awk '{ sor[NR] = $0 }
- END { for(i=NR; i>0; i--) print sor[i] }' lista

- Az alábbi program a lista állomány minden sorát indexelve külön tömbelembe gyűjti, majd fordított sorrendben kiírja.

Társtömbök

- Egy tömb indexe rendszerint csak egész szám lehet, az awk-ban azonban bármilyen értéket használhatunk tömbindexként.

- for (valt in tomb)
  utasitas

A valt értékét minden indexre beállítja, így szervezi ciklusba a tömb indexeit (nem az elemeit!!!).

split(s,a,c)

- s: stringet
- c: mezőelválasztó szerint
- a: tömbbe teszi
- mymonths: létrehozza a tömböt

```
END {
        numelements=split(honapok,mymonths,",")
        print
mymonths[1],mymonths[numelements]
}
```

> Jan Dec

Példa: 1

- Van egy állományunk (lista.txt), mely tartalmazza, hogy ki mennyi kockacukrot evett meg egy-egy alkalommal. Számoljuk ki, hogy összesen, ki mennyi kockacukrot evett meg!

Példa: 1

```
#!/bin/awk
{
        sum[$1]+=$2
}
END   {
        for (name in sum)
        print name, sum[name]
}
```

```
cat lista.txt | awk –f program1
```

- „lista.txt” fájl tartalma
- ----------
- Tamás 200
- Jóska 123
- Tibor 365
- Zsuzsa 564
- Edit 214
- Tibor 112
- Elemér 423
- Jóska 433
- Tibor 452
- Tamás 423
- ----------

Példa: 1

- Egy tömbelemnek van értéke és indexe
- A nevekre $1-el hivatkozunk, a nevek lesznek a tömb indexei
- Az index alapján adunk a tömbelemnek értéket
- Ha a név megegyezik egy korábbi névvel, vagyis az index azonos, akkor hozzáadja (+=) a meglévő értékhez
- Ha a „+=” helyett csak „=” lenne, akkor azt tudnánk meg, hogy ki mennyi kockacukrot evett meg a legutóbbi alkalommal.

Példa: 2

- A következő program megszámolja, hogy a „lista” állományban melyik szóból mennyi található.

```
#!/bin/awk
{ for (i=1; i<= NF; i++) num[$i]++ }
END  {for (word in num)
print word, num[word]
}
```

```
awk –f program2
lista.txt
ls | awk –f program2
```

Példa: 3

- A bemenet sorait számolja meg.
- Az input az „ls” parancs eredménye

```
#!/bin/awk
        { s+=1 }
END  { print „összeg: ”,s}
```

```
$ ls | awk –f program3 lista
```

Példa: 4

- A program kiírja az inputként megadott leghosszabb sort.

```
#!/bin/awk

BEGIN {
        max=0; sor=""
}
{ if (length($0)>max){
  max=length($0); sor=$0
  }
}
END {print sor}
```

Példa: 5

- A program kiírja a számokat 1-től 10-ig.

```
#!/bin/awk

BEGIN {
        i=0
        while (i<10)
        {
                print i
                i++
        }
}
```

Példa: 6

- Írj awk programot, mely megszámolja az inputban megadott karakterek, szavak és sorok számát.

```
#!bin/awk
{
        kar+=length($0)
        szo+=NF
}
END{
        print "A karakterek száma: "kar; print "A szavak
száma: "szo
        print "A sorok száma: "NR
}
```

```
$ ls | awk -f pelda6
$ cat lista.txt | awk -f pelda6
```

Példa: 7

- Add meg azt a parancsot mellyel módosítod a ”who” parancs eredményét úgy, hogy a mezőelválasztó karakter ” @ ” a rekordelválasztó pedig ”$” legyen.

- who | awk 'BEGIN{OFS="@";ORS=”$”}{print $1,$2,$3,$4,$5,$6}'

Példa: 8

- Írasd azon felhasználók nevét, akik 14:35-kor léptek be!

```
rwho | awk '{if ($5=="14:35"){print $1}}'
```

Példa: 9

- Írasd ki azon felhasználók nevét akik 10 perce nem csináltak semmit!

```
$ rwho | awk '{if ($6==":11"){print $1}}
```

Példa: 10

- Írasd ki azon felhasználók számát akik több mint tíz perce nem csináltak semmit.

- $ rwho | awk '{print $6}' | awk 'BEGIN{FS=":"}{print $2}'| grep -v "^$" | awk '{if ($1>10)print $1}' | wc -l