awk

trajanje: 1 dan

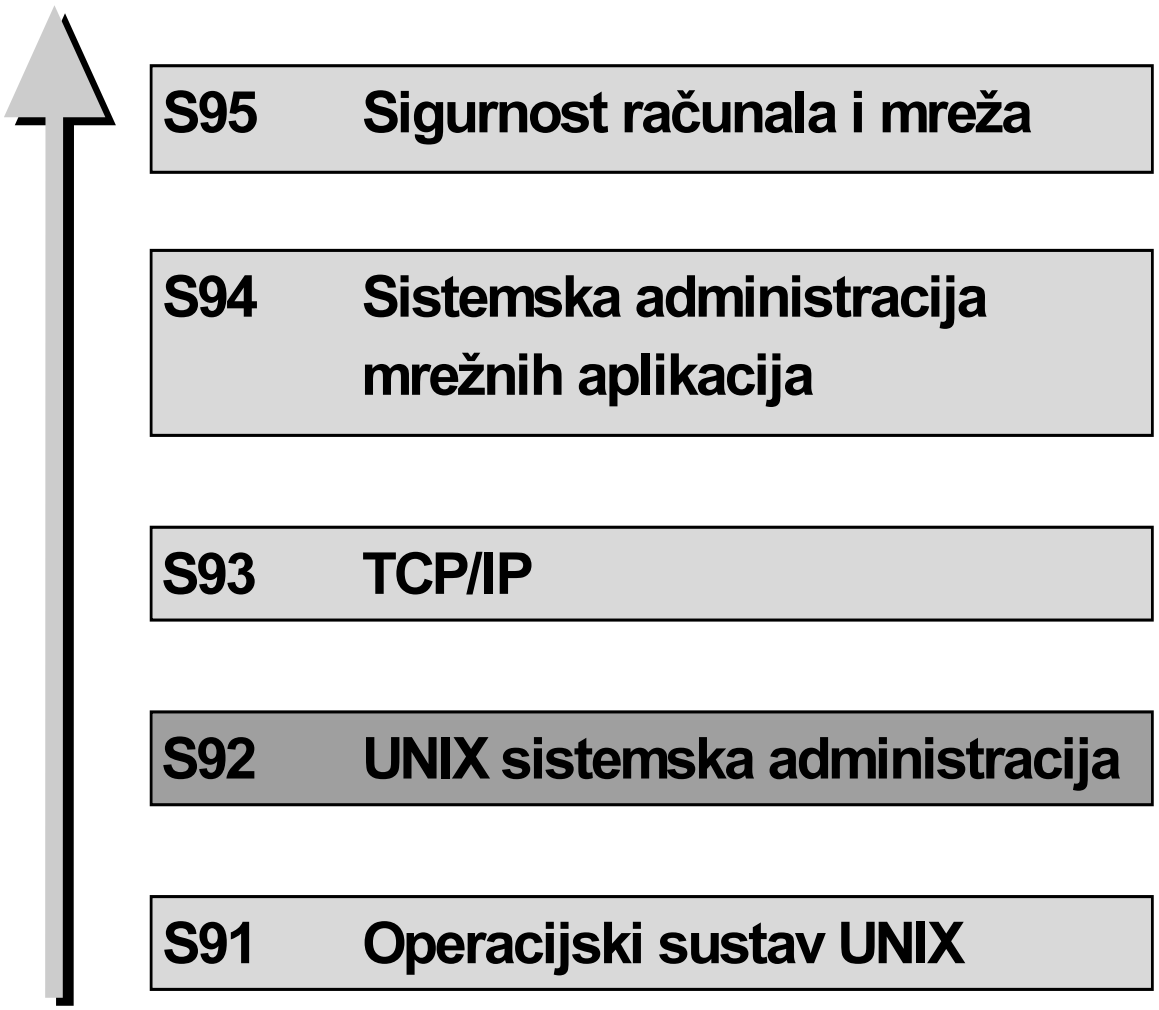
priredili:

Rajnović Damir

Delija Damir

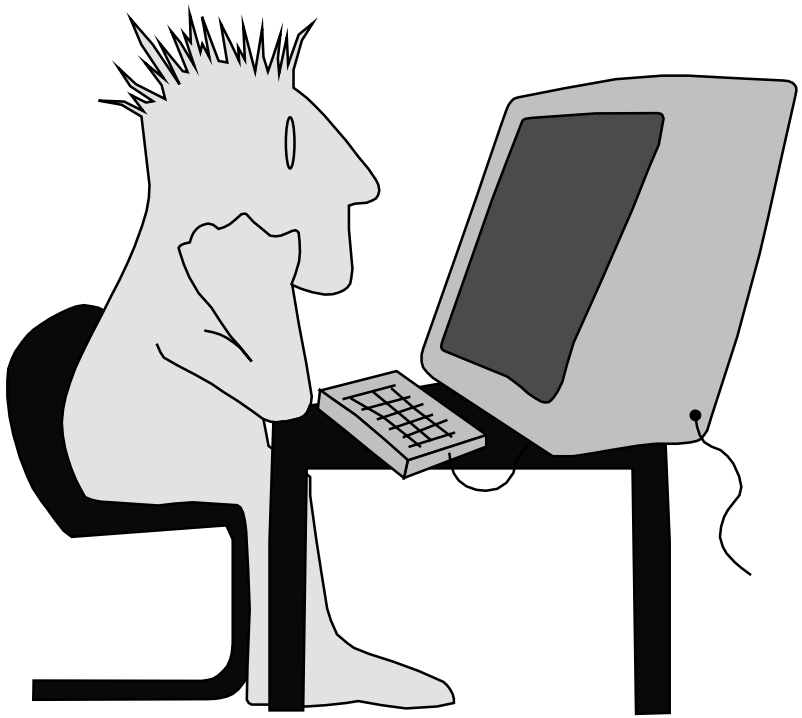
verzija 1.0

Kolovoz 1998.


S95 Sigurnost računala i mreža
S94 Sistemska administracija mrežnih aplikacija
S93 TCP/IP
S92 UNIX sistemska administracija
S91 Operacijski sustav UNIX

Ciljevi tečaja

- upoznati se sa awk jezikom i alatima
- osposobiti se za samostalni rad sa awk jezikom



Područja

- Pregled awk jezika
- Primjeri korištenja awk jezika

Potrebno predznanje

- Korisnički rad sa UNIX-om
- Poznavanje shell-a
- Poznavanje UNIX-a utilities

Sadržaj
Dan 1

Uvod	15 min
Upoznavanje awk-a	30 min
Pauza	
Upoznavanje awk-a	45 min
Pauza	
Upoznavanje awk-a	45 min
Pauza	
Upoznavanje awk-a	45 min
Ručak	
Primjeri awk koda	90 min

Što nećete naučiti na tečaju

- Puno toga!
- Neć ete dobiti gotove ‘recepte’ za sve vaše probleme

Izvori

- Awk & Sed, O'Reilly & Associates, inc.
- SunOS dokumentacija
- OSF1 dokumentacija,

Uvod

awk

- promjena podataka
- jezik za generiranje izvještaja
- jezik za pisanje interpretiranih filter program

awk - opis

SINOPSIS

```
awk [-F string][-f prog][-v var=value][program]var=value][file]
```

OPIS

- awk je jezik za obradu datoteka, podesan za manipuliranje i dohvat podataka iz ASCII datoteka.

- Datoteke navedene u komadnoj liniji se koriste kao ulazni podaci za awk, inače awk čita podatke sa standardnog ulaza (konvencija filter programa)

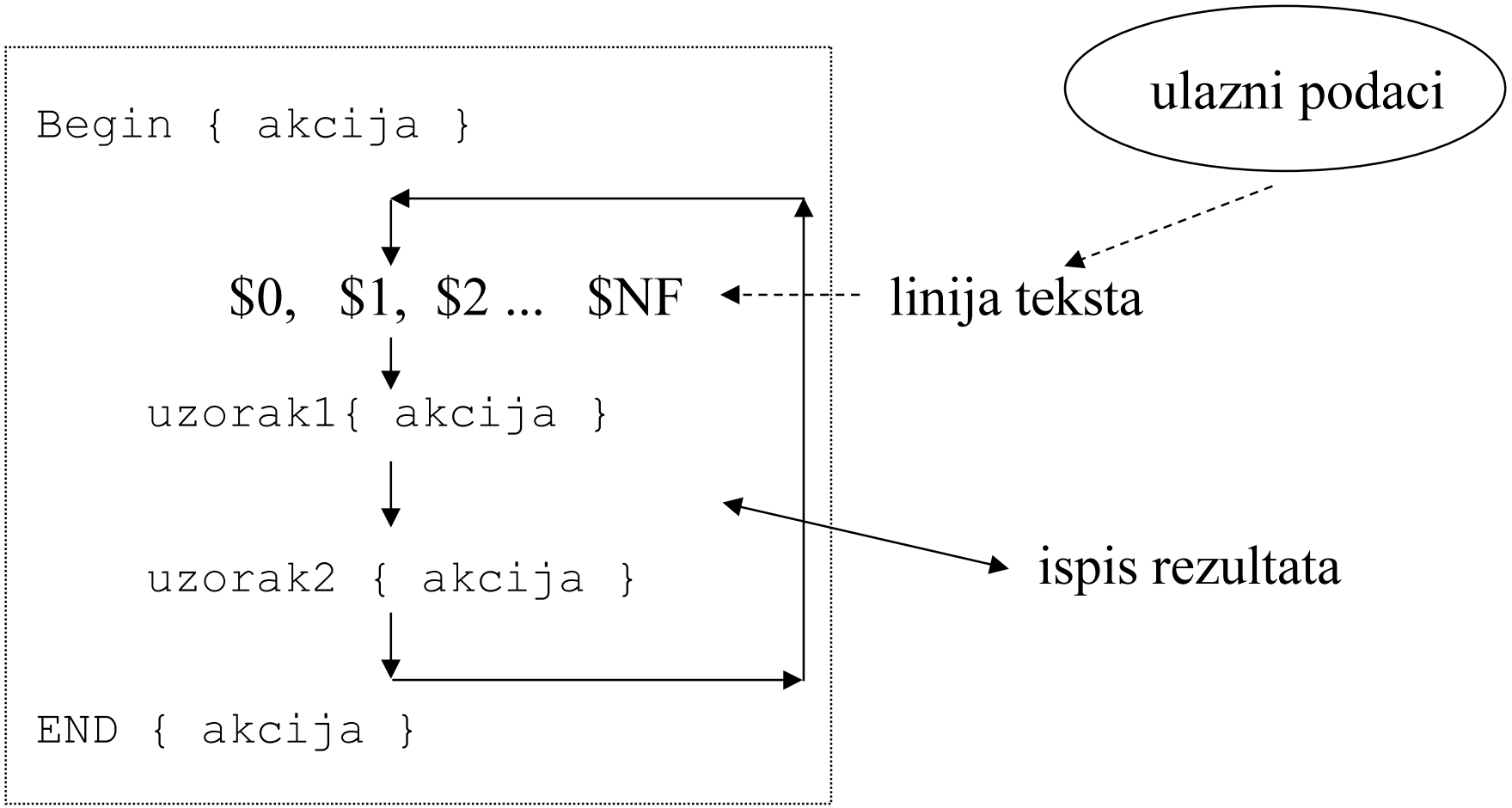
awk program

- awk program se sastoji iz proizvoljnog broja definiranih funkcija i pravila oblika:

```
pattern {action}
```

- Prije izvođenja programa kod se prevodi !

awk program


Begin { akcija }
$0, $1, $2 ... $NF
uzorak1{ akcija }
uzorak2 { akcija }
END { akcija }
ulazni podaci
linija teksta
ispis rezultata

Pokretanje awk programa

Postoje dva načina pokretanja awk programa

- Sa komandne linije

program zadan kao komanda ljuske, obično u zatvorena u apostrofe ('), da bi se izbjegla ekspanzija parametara

```
awk '{ print $2 }'
```

- Korištenjem -f prog opcije awk-a

```
awk -f prog1.awk
```

awk opcije

`-F string`

definira string kao separator polja

`-f prog`

ako ima više -f opcija u komandnoj liniji svi programi se spajaju u jedan prije izvođenja

`-v var=value`

dodjeljuje varijabli var vrijednost var prije izvođenja programa.

```
awk -v v1=10 -f prog datafile
```

Inicijalizacija varijabli sa komandne linije

`var=value`

`awk -f progfile` *`a=1 f1 f2 a=2 f3`*

postavlja *a* na 1 prije čitanja iz datoteka *f1* i *f2*, te postavlja *a* na 2 prije čitanja iz datoteke *f3*

- Inicijalizacija varijabli navedenih prije prove datoteke u komadnoj liniji događa se odmah nakon izvođenja BEGIN pravila u programu.
- Inicijalizacija varijabli navedenih poslije zadnje datoteke u komadnoj liniji događa se prije izvođenja END pravila u programu.

Varijable

Tipovi varijabli u awk-u:

- **identifikatori**
  - Identifikator je niz **slova** i **brojeva** te **_**, a ne smiju početi sa brojem
- **polja elemenata**
  - Polja su asocijativna polja oblika

```
identifier[subscript]=value
```

- awk varijable ne treba ni deklarirati ni inicijalizirati. Vrijednost se automatski postavlja na **“”** (prazni string)

Izrazi

- Izrazi u awk se sastoje iz:
  - konstatni, varijabli, funkcija,
  - regularnih izraza,
  - `subscript in array' uslova

  vezanih operatorima.

- Varijable i izrazi imaju string vrijednost i numeričku vrijednost, kontekst odlučuje sto se koristi

String konstante posebnog značenja

\a	audible bell
\b	backspace
\f	formfeed
\n	newline
\r	carriage return
\t	horizontal tab
\v	vertical tab
\ooo	octal value ooo
xdd	hexadecimal value dd
\"	quote
\c	any other character c

Konstante

- Brojčane konstante su niz dekadskih znamenki

```
a=123.5
```

- String konstante su navedene u navodnicima

```
a="ima i nema 12.3"
```

Symbol Table

Symbol table se može dohvatiti kroz ugrađeno polje SYMTAB.

```
SYMTAB[expr]
```

- je isto što i varijabla čije ime daje expr

```
SYMTAB["var"]
```

- je sinonim za varijablu var.

Environment

- awk program može doseći environment pomoć u *ENVIRON* polja

```
ENVIRON[name] = value
```

Primjer: Ispis je isti kao i za komadu `env(1)`:

```
BEGIN {
   for (i in ENVIRON)
      printf("%s=%s\n", i,ENVIRON[i])
   exit
}
```

Ugrađene varijable ARGV i ARGC

- ARGV polje sadrž i argumente komandne linije
- ARGC broj argumenata komandne linije

Operatori

- awk koristi standardnu hijerarhiju matematičkih operatora

- ako dva izraza nisu odvojena operatorom, awk ih povezuje kao jedan string.

```
a1 b2 ⇒ a1b2
```

Uvjetno izvođenje
uvjetni operator

```
expr ? expr1 : expr2
```

- vraća expr1 ako je expr istina, inače vraća expr2
- istina je sve različito od 0 ili “”

Posebni operatori regularni izrazi

`~' daje 1 ako regularni izraz na lijevoj strani odgovara desnoj strani

`!~' daje 1 ako regularni izraz na lijevoj strani ne odgovara desnoj strani

`$2 ~ /[0-9]/` selektira sve linije u kojima drugo polje sadrž i bar jednu znamenku

Pridruž enje vrijednosti varijabli

- varijabla var poprima vrijednost izraza expr

```
var = expr
```

```
var op= expr ⇔ var = var op expr
```

`a=b+0`	forsira numeričko izračunavanje
`a=a+0`	forsira a kao broj
`s=s""`	forsira s kao string

Hijerarhija operatora

`V[a]`	operacija sa poljima
`V++ V-- ++V --V`	increment, decrement
`A^B`	eksponent
`A -A !A`	unary plus,unary minus,logički NOT
`A*B A/B A%B`	množenje, dijelenje, ostatak
`A+B A-B`	zbrajanje, oduzimanje
`AB`	spajanje stringova
`A<B A>B A<=B`	
`A>=BA!=B A==B`	uspoređivanje
`A~B A!~B`	regularni izrazi
`A in V`	pripadnost polju
`A && B`	logički AND
`A || B`	logički ILI
`A ? B : C`	uslovno izvođenje
`V=B V+=B V-=B`	aritmetičke operacije

Ulaz - (input)

- awk dijeli ulaz (input) u zapise ili linije (records)

RS daje trenutni znak odjeljivač zapisa (record separator character)

```
RS=\n
```

- awk dijeli zapise u riječi `$1 ..... $N`

$i je i-ta riječ u tekućem zapisu

$0 je cijeli zapis, tekuća linija ulaza

- FS daje tekuće znakove za odjeljivanje riječi u liniji

```
FS = "[,:$]"
```

zarez,dvotočka,$ su separatori za riječi u liniji

Ulaz - dodatne varijable

`NF`	daje broj riječi u tekuć oj liniji
`NR`	daje broj linja pročitanih do sada
`FILENAME`	daje ime datoteke iz koje se čita tekući zapis
`FNR`	daje broj linja pročitanih do sada iz tekuće datoteke

getline

- getline čita vrijednost `$0` iz:
  - tekućeg ulaza,
  - datoteke,
  - ili pipe.

- Rezultat getline je broj
  - 0 uspjeh,
  - 1 end-of-file,
  - -1 greška

Format getline

`getline`

čita liniju i podešava sve varijable `($i,NF, NR,FNR)`

`getline var`

čita liniju i stavlja je u var, ne dijeli je u $i, ali podešava NR i FNR

`getline <expr`

čita liniju iz datoteke čije je ime expr, podešava varijable $i,NF

`getline var <expr`

čita liniju iz datoteke čije je ime expr u varijablu var, ne dijeli je u $i

getline i pipes

`expr | getline`

čita linije iz pipea koji nastaje izvođenjem expr, radi kao `getline <expr.`

`expr | getline var`

čita linije iz pipea koji nastaje izvođenjem expr,
radi kao getline `var <expr`

Važno:

Datoteke i pipeovi se zatvaraju sa komandom close

`close(expr)`

Ugrađene matematičke funkcije

```
int(expr)

exp(expr), log(expr), sqrt(expr)

sin(expr), cos(expr)

atan2(expr1, expr2)
```

Ugrađene string funkcije

Ako se ispuste argumenti koristi se `$0, i FS` varijabla

```
len  = length(expr)
n    = split(string, array, regexp)
str  = substr(string, offset, len)
pos  = index(string, str)
pos  = match(string, regexp)
n    = sub(regexp, repl, string)
n    = gsub(regexp, repl, string)
str  = sprintf(fmt, expr, expr...)
n    = ord(expr)
str  = tolower(expr)
str  = toupper(expr)
```

Funkcija system

system funkcija izvodi vanjsku komandu expr i vraća rezultat

```
status = system(expr)
```

```
system("tail " $1)
```

Korisnički definirane funkcije

```
function name(parameter-list) {
   statements
}
```

- Definicija funkcije može biti bilo gdje u tijelu programa
- Lista parametara se sastoji iz varijabli odvojenih zarezom, to su ujedno i lokalne varijable za funkciju.
- Sve ostale varijable navedene u funkciji su globalne
- Skalarni argumenti se prenose po vrijednosti, a polja po referenci
- Funkcija se vraća na return komandu ili nakon izvođenja svoje zadnje komande

Uzorci i akcije
(patterns & actions)

```
uzorak {
   komanda; komanda
   komanda
}
```

- **uzorak** (pattern) je regularni izraz, poseban pattern, raspon ili aritmetički izraz

- **akcija** je niz komandi koje završavaju sa **`; \n }`**

- *ako je uzorak ispušten, linija se uvijek izvodi*!

Posebni uzorci

`BEGIN`

definira akcije koje se izvode prije čitanja ulaznih podataka

`END`

definira akcije koje se izvode poslije čitanja ulaznih podataka

`pattern1,pattern2`

raspon uzoraka (pattern range), hvata sve linije koje zadovoljavaju uzorke od pattern1 do pattern2 uključivo

IF - THEN - ELSE

```
# if statement AKO TADA INAČE

if (condition){
    statement
} else {
    statement
}
```

Komentar

```
# prvi znak u liniji,
# komentar je sve do kraja linije


komanda  ;# za komentar u istoj
         #liniji sa komandama
```

Petlje (loops)

`while`

`do`

`for`

while

```
# while loop
while (uslov) {
  komanda
}
```

- odvija se sve dok je uslov istina (!= 0)

do-while

```
# do-while loop
do {
  komanda
} while (uslov)
```

- odvija se dok je uslov laž !

for za asocijativna polja

```
# for
for (i in polje) {
  komanda
}
```

- awk izvodi komandu za svaki element polja, i sadrži novi indeks
- ako je polje višedimenzionalno i je dan kao jedan string,
- znak definiran u `SUBSEP` se može koristiti za odjeljivanje podindeksa za višedimenzionalna polja

for

```
# for petlja, klasična c for petlja
for (expr1; cond; expr2) {
  statement
}
```

- `expr1` početno stanje brojača u petlji
- `cond` petlja radi dok je cond istina
- `expr2` uvećavanje brojača

Komande za kontrolu petlji

`break`

prekida while petlju

`continue`

prekida tekuću iteraciju petlje i poćinje novu

`next`

prekida oradu tekuće ulazne linije i odmah nastavlja sa procesiranjem nove ulazne linije

exit

```
exit [(expr)]
```

- prelazi na `END` uzorak i vraća iz programa rezultat expr izraza, (def. 0)

return

```
return [expr]
```

- vraća program iz funkcije, rezultat je expr

delete

```
delete array[i]
array[i]=“”
```

- briše i-ti element iz polja array

print, printf

```
print expr, expr, ...
```

- štampa argumente
- ako nema argumenata štampa $0

```
printf fmt, expr, expr, ...
printf (fmt, expr, expr, ...)
```

- štampa argumente expr prema formatu fmt
- radi kao C printf format

Izlaz

`ORS` nakon svakog ispisa stavlja **Output Record Separator (**def ORF=“ \n”)

`OFS` **Output Field Separator** odvaja pojedina polja u ispisu (def OFS=“ “)

`OFMT` daje def. format za ispis floating point brojeva za print komandu. (def OFMT=`%.6g’)

Redirekcija izlaza u datoteku

```
print > expr
printf fmt,expr1 ...exprn > expr
```

- awk uzime string vrijednost expr kao ime datoteke u koju piše izlaz

```
print >>expr dodavanje na kraj datoteke
```

- *razlika `>' i `>>' je važna samo kod prvog pisanja u datoteku!*

Redirekcija izlaza u pipe

```
print |expr
```

- awk uzima string vrijednost od expr kao ime komande u koju usmjeri pipe

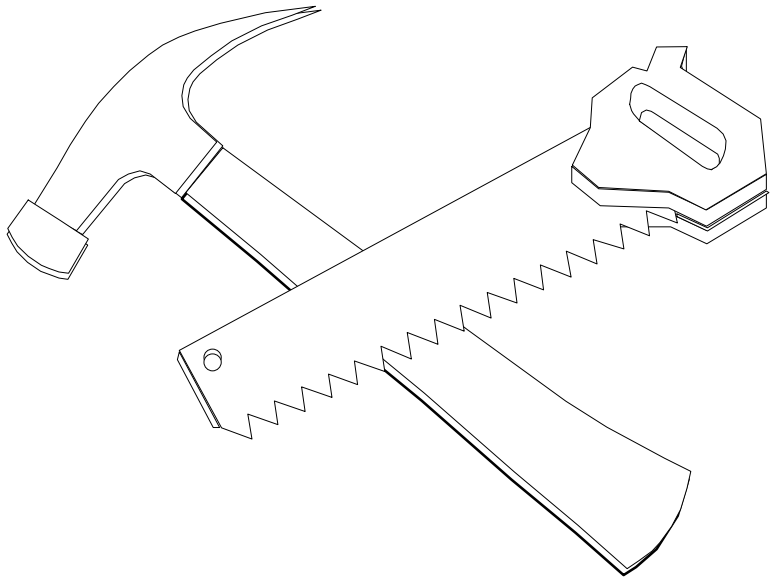
Inačice awk-a

- awk standardni awk
- nawk new awk
- gawk gnu awk (ima i date&time funckije)

- kompajler za awk
- MSwindows (NT,95,98 3xx), DOS
- awk to perl kompajler

Pauza
Ručak

Praktičan rad i primjeri



Ispis i brojenje linija datoteke

```
awk '{print NR SEP $0}' SEP=":" input1
awk '{print NR ":" $0}' input1
```

- Primjer ispisuje linije datoteke input1 i ispred svake piše njen redni broj

Pozivanje programa -f opcijom

- awk izvodi program iz addline.awk:

```
awk -f addline.awk input1
```

- Isti rezultat navođenjem koda u programskoj liniji isto kao i pozivanje addline.awk

```
awk `{print NR ":" $0}' input1
```

Primjena uzoraka

- Ovaj program sve ulazne linije koje počinju sa `January' piše u datoteku jan (koja mož e i ne mora postojati), a sve linije koje počinju sa `February' ili `March' piše u datoteku febmar.

```
/^January/ {print >> "jan"}

/^February|^March/ {print >> "febmar"}
```

Sumiranje i statistike ulaza

- Ovaj program računa ukupan broj učitan riječi u tekstu i srednju duž inu linije teksta po broju riječi

```
{s += $NF}

END{print "sum is",s,"average is",s/NR}
```

Zamjena redoslijeda polja u ulazu

- Program zamjenjuje redoslijed prvog i drugog polja u ulazu

```
{
  tmp = $1
  $1 = $2
  $2 = tmp
  print
}
```

Korištenje printf komande

- Program umeće brojeve linija. Brojevi su lijevo poravnani

```
{printf "%-6d: %s\n", NR, $0}
```

Štampanje ulaza u obratnom redoslijedu

- Program štampa ulaz u obratnom redoslijedu, koristeći polje (veliki utrošak memorije)

```
{a[NR] = $0; #index using record number }

END {
  for (i = NR; i>0; --i)
  print a[i]
}
```

Brojenje linija po nekom ključu

- Broje se linije koji imaju istu prvu riječ

```
#array indexed using the first field
{ ++a[$1]}

#note output will be in undefined order
END {
  for (i in a)
    print a[i], "lines start with", i
}
```

Brojenje linija po ulaznim datotekama

- Broje se linije za svaku ulaznu datoteku

```
{ ++a[FILENAME] }
END {
  for (file in a)
  if (a[file] == 1)
    print file, "has 1 line"
  else
    print file, "has", a[file], "lines”
}
```

Korištenje dvodimenzionalnih polja u awk-u (I)

- Za ulazne podatke provo polje je broj produkta, drugo polje je mjesec, a treć e polje količinu. Program generira tabelu koliko je čega bilo u kom mjesecu

```
    Jan Feb Mar
1#
2#
```

Korištenje dvodimenzionalnih polja u awk-u (II)

```
BEGIN { NUMPROD = 5 }
{array[$1,$2] += $3 }
END {
   print "\t Jan\t Feb\tMarch\tApril\t May\t" \
   "June\tJuly\t Aug\tSept\t Oct\t Nov\t Dec”
   for (prod = 1; prod <= NUMPROD; prod++) {
      printf "%-7s", "prod#" prod
      for (month = 1; month <= 12; month++){
           printf "\t%5d",array[prod,month]
      }
      printf "\n"
   }
}
```

Korištenje funkcija (I)

- Program inicijalizira polje u memoriji koristeći slučajne brojeve.
- Na osnovi unosa dvaju polja (koristi ih kao koordinate) utvrđuje da li je došlo do pogotka ili ne.
- Za inicijaliziranje se koristi funkcija, definirana na početku koda.

Korištenje funkcija (II)

```
function randint(){return (int((rand()+1)*10))}

BEGIN {
  prize[randint(),randint()] = "$100";
  prize[randint(),randint()] = "$10”;
  prize[1,1] = "the booby prize”
}
{
  if (($1,$2) in prize)
     printf "You have won %s!\n", prize[$1,$2]
}
```

Obratno štampanje sadržaja linije

- Programa štampa linije koje imaju isto prvo i zadnje polje, a polja u liniji štampa od zadnjeg prema prvom

```
$1==$NF {
  for (i = NF; i > 0; --i)
    printf "%s", $i(i>1 ? OFS : ORS)
}
```

Rekurzija

- Računanje faktorjela rekurzivnim pozivima

  `f(x)=x * f(x-1), x > 1, f(1)=1`

- *do 20 u dubinu!*

```
function f (num) {
  if (num <= 1)return 1
  else return num * f(num - 1)
}
{ print $0 " factorial is " f($0) }
```

Getline & pipe (I)

Čitanje podataka iz pipe-a i njihova obrada u awk-u

```
function words(file, string) {
  string = "wc " fn
  string | getline
  close(string)
  return ($2)
}
```

Getline & pipe

```
BEGIN {
  for (i=1; i<ARGC; i++) {
  fn = ARGV[i]
  printf "There are %d words in %s.", \
                  words(fn), fn
  }
}
```

Pitanja

?