AWK

Allikas: Vikipeedia

AWK	
Faililaiendid	.awk
Paradigma	skriptimine, protseduraalne, sündmus-orienteeritud
Väljalaskeaeg	1977
Looja	Alfred Aho, Peter Weinberger ja Brian Kernighan
Viimane väljalase	IEEE Std 1003.1-2004 (POSIX) / 1985
Tüüpimine	puudub; tunneb sõnesid, täisarve ja ujukoma-arve; regulaaravaldised
Implementatsioonid	awk, GNU Awk, mawk, nawk, MKS AWK, Thompson AWK (kompilaator), Awka (kompilaator)
Dialektid	vana awk (oawk) 1977, uus awk (nawk) 1985, GNU Awk (gawk)
Mõjutatud keeltest	C, SNOBOL4, Bourne shell
Mõjutanud keeli	Perl, Korn Shell (ksh93, dtksh, tksh), Lua
Veebileht	cm.bell-labs.com/cm/cs/awkbook/index.html (http://cm.bell-labs.com/cm/cs/awkbook/index.html)

AWK on Belli laboris 1977 välja töötatud programmeerimiskeel, mis on loodud tekstitöötluseks. AWK-s võetakse sisendandmed kas tekstifailist või standardsisendist.

Keele nimi "AWK" on tuletatud keele autorite perekonnanimedest Alfred Aho, Peter Weinberger ja Brian Kernighan, kuid tavaliselt seda ei hääldata nagu tähtede järjendit (A-W-K), vaid pigem [ook] nagu alk, mis on inglise keeles *auk*. Alk on ka AWK vapilind, näiteks raamatu "*The AWK Programming Language*" (http://plan9.bell-labs.com/cm/cs/awkbook/index.html) kaanel - sellele raamatule viidatakse tihti, kasutades lühendit TAPL). Väiketähtedes kirjutatuna viitab `awk` UNIXis või Plan 9-s kirjutatud programmile, mis jooksutab teisi AWKis kirjutatud programme. AWKis kirjutatud arvutiprogrammi käivitamiseks UNIXi käsurealt tuleb AWK-programmi nime ette kirjutada käsk `awk`).

> "AWK on tekstifailide töötlemise keel. Faili käsitletakse kirjete jadana ja vaikimisi on iga rida eraldi kirje. Iga rida jagatakse omakorda väljadeks, mis tähendab, et rea esimene sõna on esimene väli, rea teine sõna teine väli ja nii edasi. AWKi programm koosneb muster-käsklausetest. AWKi programm loeb sisendit reakaupa. Sisse loetud rida võrreldakse programmis iga mustriga ja iga sobiva mustri korral täidetakse mustrile järgnev käsk." - Alfred Aho[1]

AWK on näide programmeerimiskeelest, mis kasutab laialdaselt sõne andmetüüpi ja regulaaravaldisi. Algse AWKi võimsus, lakoonilisus ja piirangud inspireerisid Larry Walli kirjutama Perli. Samas defineeriti AWKi võimsamad variandid POSIX AWK ja gawk (GNU AWK). Kuigi AWK ja sed projekteeriti algselt üherealiste programmide loomiseks, kirjutati isegi Belli laborites suuri ja hästi struktureeritud AWKi programme.

AWK oli üks väheseid vahendeid, mis oli saadav UNIXi seitsmendas versioonis ja kogus populaarsust UNIXi "torule" (*pipeline*) arvutuslike lisavõimaluste lisamise vahendina. Peaaegu igale moodsale UNIXi-laadsele operatsioonisüsteemile on sisse ehitatud mingisugune AWKi versioon. AWKi peetakse UNIXi-laadse operatsioonisüsteemi kohustuslikuks osaks. Lisaks Bourne'i shellile on AWK ainus standardses UNIXi keskkonnas kättesaadav skriptimiskeel.

Sisukord

AWKi programmide struktuur

AWKi programm on muster-tegevus-paaride seeria, mis on kirjutatud kujul

muster { *tegevus* },

kus muster on tavaliselt avaldis ja tegevus on käskude jada.

Iga sisendirida võrreldakse kõikide mustritega järgemööda ja tegevus täidetakse iga mustri kohta, mille avaldis on tõene.

Nii mustri kui ka tegevuse võib ära jätta. Mustri puudumisel sobib iga sisendrida. Vaikimisi tegevus on sisendirea väljastamine standardväljundisse.

Lisaks lihtsale AWKi avaldisele võib muster olla ka *BEGIN* või *END*, mille tulemusel täidetakse tegevus vastavalt kas enne või pärast sisendist kõikide ridade lugemist, või muster1, muster2, mis võtab terve hulga sisendridasid alates reast, mis sobitub muster1ga, kuni reani, mis sobitub muster2ga, enne kui võrdleb järjekordset sisendirida muster1ga.

Lisaks tavalistele aritmeetika ja loogika tehtemärkidele on AWKi avaldistes ka tilde-tehtemärk ~, mis võrdleb regulaaravaldist sõnega. Kasuliku süntaktilise suhkruna võrdleb /regexp/ (regulaaravaldis) ilma tilde-tehtemärgita käesoleva sisendreaga.

AWKi käsud

AWKi käsud on laused, mis tohivad eelnevates näidetes olla tegevuse asemel. AWKi käskudeks on funktsioonide väljakutsed, muutujate väärtustamine, arvutused ja eelnevate kombinatsioonid. AWKi on sisse ehitatud mitme funktsiooni tugi ja AWKi eri versioonid pakuvad neile rikkalikku lisa.

Lihtsuse huvides on järgnevates näidetes ära jäetud looksulud.

print-käsk

print-käsku kasutatakse teksti väljastamiseks. Väljastatav tekst on määratletud eeldefineeritud stringiga, mille nimetus on ORS (*output record separator* - väljundkirje eraldaja) ja vaikeväärtuseks reavahetus. print-käsu lihtsaim vorm on:

```
print
```

See kuvab käesoleva rea. AWKis on read jagatud väljadeks ja neid saab eraldi kuvada.

```
print $1
```

kuvab käsiloleva rea esimese välja.

```
print $1, $3
```

kuvab käsiloleva rea esimese ja kolmanda välja, mis on omavahel eraldatud väljundvälja eraldajaga (OFS - *output field separator*), mille vaikeväärtuseks on üks tühik.

Kuigi need väljad ($X) sarnanevad muutujatele (dollarimärk "$" viitab Perlis muutujale), märgivad nad tegelikult välju käesolevas reas. Erijuht $0 viitab tervele reale. Käsud "`print`" ja "`print $0`" on võrdväärsed.

print-käsku võib kasutada ka arvutuste ja funktsioonide tulemuste kuvamiseks:

print 3+2
print omafunktsioon(3)
print omafunktsioon(muutuja)
print sin(3-2)

Väljundi saab suunata faili:

print "avaldis" > "failinimi"

või UNIXi "toru" kasutades:

print "avaldis" | "käsk"

Standardmuutujad

AWKi sisseehitatud ehk standardsete muutujate hulka kuuluvad väljamuutujad $1, $2, $3 ja nii edasi ($0 tähistab tervet rida). Nad tagastavad välja väärtuse või väljal oleva teksti käesolevast reast.

On veel teisigi standardmuutujaid:

- NR sisaldab käsiloleva sisendist loetud tekstirea (sisendirea) järjekorranumbri.
- NF sisaldab sisendireas olevate sõnade arvu. Tekstirea viimasele väljale saab viidata muutujaga $NF.
- FILENAME sisaldab käsiloleva sisendfaili nime.
- FS sisaldab väljaeraldaja sümboli, mida kasutatakse väljade eraldajana sisendfailis. Väljaeraldaja vaikeväärtus on suvaline arv tühikuid või tabulaatorivajutusi. FS-i väärtust saab muuta vastavalt väljaeraldajale sisendfailis.
- RS sisaldab kirjeeraldaja. Kuna vaikimisi on iga sisendrida eraldi kirje, siis vaikimisi on kirjeeraldajaks

reavahetuse sümbol.
- OFS sisaldab väljundi väljaeraldaja. Vaikimisi on OFS väärtuseks tühik.
- ORS sisaldab väljundi kirjeeraldaja. Vaikimisi on ORS väärtuseks reavahetuse sümbol.
- OFMT sisaldab numbrilise väljundi vormingu. Vaikimisi vorming on "%.6g".

Muutujad ja süntaks

Muutujanimed võivad sisaldada kõiki ladina suur- ja väiketähti, numbreid ja allkriipsu, kuid muutujanimena ei tohi kasutada AWKi võtmesõnu.

Tehtemärgid +, -, * ja / tähistavad vastavalt liitmist, lahutamist, korrutamist ja jagamist. Sõnede liitmiseks (sidurdamiseks) tuleb kaks muutujat või sõnekonstanti teineteise järele kirjutada. Sõnekonstantide vahelt tohib tühiku ära jätta, kuid kahe muutuja liitmisel tuleb nende vahele kindlasti tühik kirjutada. Sõnekonstante eraldatakse ja tähistatakse jutumärkidega.

Programmilausete lõppu ei pea panema semikoolonit.

Kommentaare võib programmi lisada, kasutades #-märki rea alguses.

Kasutaja defineeritud funktsioonid

Funktsioone defineeritakse nagu C-keeles. Funktsiooni definitsioon koosneb võtmesõnast `function`, funktsiooni nimest, argumentide nimedest ja funktsiooni kehast ehk sisust. Järgneb funkstiooni näide.

```
function lisa_kolm (number) {
  return number +3
}
```

Selle funktsiooni väljakutsumiseks sobib käsk

print lisa_kolm(36),

mis väljastab **39**.

Funktsioonides saavad olla kohalikud muutujad. Nende nimed lisatakse parameetrite loendi lõppu, kuid neid ei ole tarvis funktsiooni väljakutsumisel väärtustada. Tavaks on lisada argumentide loendisse enne kohalikke muutujaid rohkem tühikuid, et märkida kohta, kus argumendid lõpevad ja kohalikud muutujad algavad.

Funktsiooni definitsioonis võivad olla liigsed tühikud funktsiooni nime ja algussulu vahel, kuid funktsiooni väljakutses ei tohi liigseid tühikuid olla.

Lihtsad rakendused

Hello world

Programm "Tere, maailm!" kirjutatatakse AWKis nii:

BEGIN { print "Tere, maailm!" }

Siin ei ole vaja ilmset `exit`-käsku, kuna ainus muster on `BEGIN`, ühtegi käsurea argumenti ei töödelda.

Pikemate kui 80 sümboli pikkuste ridade väljastamine

Järgmist käsku kasutatakse 80 sümbolist pikemate ridade väljastamiseks. Vaikimisi väljastatakse käesolev rida.

length($0) > 80

Sisendist loetud sõnade arvu väljastamine

Loendada sisendist tulevaid sõnu ning väljastada ridade, sõnade ja sümbolite arv (nagu "wc" ehk "*word count*" käsk UNIXis):

```
{
    w += NF
    c += length + 1
}
END { print NR, w, c }
```

Kuna programmi esimeses reas pole antud mustrit, siis vaikimisi sobib iga sisendrida ja järgnevad käsud täidetakse kõikidel ridadel.

`w += NF` on lühendatud variant ja tähendab sama mis `w = w + NF`.

Viimaste sõnade liitmine

```
{ s += $NF }
END { print s + 0 }
```

s-i suurendatakse NF-i võrra, mis on sisendirea viimase sõna (välja) järjekorranumber selles reas. Näiteks $4 tähistab 4. välja väärtust, siis $NF tähistab viimast välja olenemata sellest, mitu välja reas on (võib olla nii rohkem kui vähem kui eelnevates/järgnevates ridades). Dollarimärk $ on unaarne tehtemärk kõrgeima prioriteediga. Kui sisendreas pole ühtegi välja, siis NF väärtus on 0 ning $0, mis tähistab tervet rida, on tühi ja selle väärtus on samuti 0.

Pärast kogu sisendi lugemist sobitub END-muster ja seejärel väljastatakse s. Kuid sisendis ei pruukinud olla ühtegi rida, mis tähendab, et s ei väärtustunud ja seega s väärtus on tühi sõne. Nulli liitmine muutuja väärtusele on AWKi eripära sõne numbriliseks muutmiseks. Tühja sõne ja muutujate liitmist kasutatakse numbrilise väärtuse sõneks muutmiseks, näiteks s "". Sõnede liitmiseks pole operaatorit vaja, nad asetatakse lihtsalt järjest. Nulli liitmise tõttu väljastab programm 0, vastasel juhul oleks väljastatud tühi rida.

Sõnade sageduste arvutamine

Sõnade sageduste leidmiseks kasutatakse assotsiatiivseid liste:

```
BEGIN {
    FS="[^a-zA-Z]+"
}

{
     for (i=1; i<=NF; i++)
          words[tolower($i)]++
}

END {
    for (i in words)
```

```
        print i, words[i]
}
```

BEGIN-plokk seab väljaeraldajaks suvalise sümbolijada, mis ei sisalda tähti. Sealjuures võivad eraldajad olla regulaaravaldised. Pärast seda järgneb tegevus, mille käigus vaadatakse läbi iga sisendrida. Iga sisendrea sõna sagedust suurendatakse 1 võrra. Enne seda muudetakse iga sõna väiketähtedest koosnevaks, et sama sõna ei loetaks erinevaks olenevalt suurtähtede esinemisest sõnas. Lõpuks väljastatakse END-plokis sõnad koos vastavate esinemissagedustega. Rida

for (i in words)

loob tsükli, mis käib läbi sõnade loendi, muutes i-d igaks loendi alaindeksiks, mis tähendab, et sõnade loendi pikkust pole vaja teada. See erineb enamikust keeltest, kus tsükkel käib läbi iga väärtuse etteantud loendis.

Käsurealt mustri sobitamine

Seda programmi võib esitada mitut moodi. Esimene kasutab Bourne'i shelli, et luua kestaskript, mis teeb kõik ära. See on neist meetodeist lühim:

```
$ cat grepinawk
pattern=$1
shift
awk '/'$pattern'/ { print FILENAME ":" $0 }' $*
$
```

`$pattern` muster ei ole AWK käsus jutumärkidega kaitstud. Muster kontrollib tavalisel viisil, kas terve rida (`$0`) sobitub. `FILENAME` sisaldab käesoleva faili nime. Kaks järjestikku asuvat sõnet liidetakse automaatselt. AWKis puudub selleks spetsiaalne tehe. `$0` viitab algsele muutmata sisendreale.

Sama asja kirjutamiseks on teisigi mooduseid. Järgnev kestaskript pöördub keskkonna poole otse awk'ist:

```
$ cat grepinawk
muster=$1
shift
awk '$0 ~ ENVIRON["muster"] { print FILENAME ":" $0 }' $*
$
```

See on kestaskript, mis kasutab käsku `ENVIRON` - järjend, mis lisandus uuemas awki versioonis. `ENVIRON` alaindeks on keskkonnamuutuja nimi. Tulemuseks on muutuja väärtus. See on nagu *getenv* funktsioon standardteekides ja POSIXis. Kestaskript loob keskkonnamuutuja `pattern`, mis sisaldab esimest parameetrit, siis kaotab selle parameetri ja laseb awkil otsida mustrit igast failist.

~ võrdleb oma vasakut operandi parema operandiga (`!~` on selle pöördtehe). Pane tähele, et regulaaravaldis on lihtsalt sõne ja seda saab hoida muutujas.

Järgmine variant kasutab käsureal muutuja väärtustamist, milles awki parameetrit võib näha muutuja väärtustamisena:

```
$ cat grepinawk
muster=$1
shift
```

```
    awk '$0 ~ muster { print FILENAME ":" $0 }' "muster=$muster" $*
    $
```

Viimane variant on kirjutatud puhtalt AWKis ilma kestaskriptide abita ja ilma vajaduseta teada palju AWKi skriptide teostusest (nagu on vaja teada käsurealt muutuja väärtustamise näite korral), kuid see on eeltoodutest pikem:

```
BEGIN {
    muster = ARGV[1]
    for (i = 1; i < ARGC; i++) # eemalda esimene argument
        ARGV[i] = ARGV[i + 1]
    ARGC--
    if (ARGC == 1) { # muster oli ainus argument, seega tuleb sisendit lugeda standardsisendist
        ARGC = 2
        ARGV[1] = "-"
    }
}
$0 ~ muster { print FILENAME ":" $0 }
```

`BEGIN` on lisaks esimese argumendi eraldamisele vajalik ka selleks, et takistada programmil seda failinimena tõlgendada, kui `BEGIN`-plokk lõpeb. Parameetrite arv `ARGC` on alati ≥1, sest `ARGV[0]` on see käsk, mis käivitas selle skripti - enamasti sõne `"awk"`. Sealjuures `ARGV[ARGC]` on tühisõne `""`. `#` algatab kommentaari, mis ulatub kuni rea lõpuni.

`if`-plokis awk kontrollib, kas on vaja lugeda standardsisendist enne käskude täitmist. See tähendab, et

awk 'prog'

töötab üksnes sellepärast, et fakti, et failinimed puuduvad, kontrollitakse ainult enne `prog` käivitamist. Kui seada otseselt `ARGC` väärtuseks 1, mis näitab, et argumente ei ole, siis awk lõpetab töö, sest talle tundub, et sisendfaile pole. Sellepärast tuleb otseselt öelda, et on vaja lugeda standardsisendist, erilise failinimega `-`.

Seesmised AWK skriptid

Nagu mitmes teises programmeerimiskeeles, saab ka AWKis konstrueerida seesmisi skripte nn "*shebang*" (UNIXi) süntaksi abil.

Näiteks UNIXi käsu `hello.awk`, mis väljastab sõne "Tere, maailm!", võib tekitada, luues faili nimega `hello.awk`, mis sisaldab järgnevaid ridu:

```
#!/usr/bin/awk -f
BEGIN { print "Tere, maailm!" }
```

`-f` ütleb awkile, et järgnev argument on fail, millest tuleb lugeda AWK programm.

AWKi versioonid ja realisatsioonid

AWK kirjutati levitati algselt UNIXi 7. versiooniga. 1985 hakkasid autorid keelt laiendama, põhiliselt kasutaja defineeritud funktsioonide lisamisega. AWKi kirjeldatakse raamatus "*The AWK Programming Language*", mis avaldati 1988. Selle keele realisatsioonid olid kättesaadavad alates UNIXi System V-st. Et vältida arusaamatusi mitteühilduva vanema versiooniga, teatakse seda versiooni nime all "uus awk" ehk *nawk*. See realisatsioon oli kättesaadav vabavaraliselt alates 1996 ja seda haldab veel ikka Brian Kernighan.

Vanemad versioonid UNIXist (nagu näiteks UNIX/32V), koosseisus oli `awkcc`, mis teisendasid AWKi C-sse. Kernighan kirjutas programmi, millega sai teisendada AWKi C++-sse, kuid selle seisukord ei ole teada.[2]

BWK awk viitab Brian Kernighani versioonile. Seda on nimetatud ainsaks tõeliseks AWKiks (*One True AWK*), sest seda mõistet seostatakse raamatuga, mis algselt kirjeldas AWKi, ja ka fakti pärast, et Kernighan oli üks AWKi loojatest [3]. FreeBSD viitab kõnealusele versioonile sama nimega (*one-true-awk* [4]. Selles versioonis on ka võimalusi, mida raamatus pole kirjeldatud, näiteks `tolower` ja `ENVIRON`, mis on eelnevalt selgitatud. Lisateavet vaata failist FIXES allikate loetelus. BWK awki kasutavad näiteks FreeBSD, NetBSD ja MacOS X.

gawk (GNU awk) on järjekordne tasuta tarkvara realisatsioon. See võimaldab kasutajal funktsionaalsust laiendada kasutaja kirjutatud teekide abil. See kirjutati enne kui algne realisatsioon vabalt kättesaadavaks sai ja on siiani laialdaselt kasutusel. Mitmel Linuxil on uusim gawki versioon ja seda tunnustatakse awki *de facto* realisatsioonina Linuxi maailmas. gawki versioon 3.0 oli awki nime all FreeBSDs enne versiooni 5.0. Järgnevad FreeBSD versioonid kasutasid BWK awki, et vältida GNU GPLi [5].

mawk on väga kiire AWKi realisatsioon, mille autor on Mike Brennan. See põhineb baitkoodi interprataatoril.

awka, mille algusosa on kirjutatud programmile mawk, on veel üks translaator, mis teisendab AWKi skripte C-sse. Pärast kompileerimist (autori *libawk.a* lisamist) saadud käivitatavad failid on oluliselt kiiremad ja autori testide põhjal võrreldavad AWKi teiste versioonide, Perli ja Tcliga. Väikesed skriptid teisenevad programmideks suurusega 160-170 kB.

tawk (Thompson AWK) on AWKi kompilaator Solarisele, MS-DOSile, OS/2le ja Windowsile, mida varem müüs Thompson Automation Software. See ettevõte on oma tegevuse lõpetanud.

Jawk on SourceForge'i projekt AWKi realiseerimiseks Javas.[6] Keelele on lisatud laiendused, et tagada AWKi skriptidest juurdepääs Java võimalustele (näiteks Java lõimed, *Collections* jne).

xgawk on SourceForge'i projekt, mis põhineb gawkil [7]. See laiendab gawki dünaamiliselt laetavate teekidega.

QSEAWK on sisene AWKi interpretaatori realisatsioon, mis sisaldub QSE teegis, mis omakorda tagab API C ja C++ jaoks.[8]

BusyBox sisaldab hõredalt dokumenteeritud AWKi realisatsiooni, mis paistab olevat täielik, autoriks Dmitri Zahharov. See on väga väike realisatsioon, mis sobib sisestele süsteemidele.

Raamatud

- (1988-01-01) *The AWK Programming Language (http://cm.bell-labs.com/cm/cs/awkbook/)* . New York, NY: Addison-Wesley. ISBN 0-201-07981-X. Retrieved on 2009-04-16. Raamatu veebilehel on lingid awki versioonidele ja võimalus alla laadida viimast versiooni.
- (2001-05-15) *Effective awk Programming (http://www.oreilly.com/catalog/awkprog3/)* , 3rd, Sebastopol, CA: O'Reilly Media. ISBN 0-596-00070-7. Retrieved on 2009-04-16.
- (1997-03-01) *sed & awk (http://www.oreilly.com/catalog/sed2/)* , 2nd, Sebastopol, CA: O'Reilly Media. ISBN 1-56592-225-5. Retrieved on 2009-04-16.
- (2000) *Effective Awk Programming: A User's Guide for Gnu Awk (http://www.gnu.org/software/gawk/manual/)* , 1.0.3, Bloomington, IN: iUniverse. ISBN 0595100341. Retrieved on 2009-04-16. Arnold Robbins arendas GNU Awki üle 10 aasta. O'Reilly

avaldas 2001. aasta mais vaba GNU awki käsiraamatu. Seda saab raamatu saidilt tasuta alla laadida.

Vaata ka

- Skriptimiskeel

Viited

1. ↑ http://www.computerworld.com.au/index.php/id;1726534212;pp;2 The A-Z of Programming Languages: AWK
2. ↑ An AWK to C++ Translator (http://cm.bell-labs.com/cm/cs/who/bwk/awkc++.ps)
3. ↑ **The AWK Programming Language**, ISBN 0-201-07981-X. (http://cm.bell-labs.com/cm/cs/awkbook/)
4. ↑ FreeBSD's work log for importing BWK awk into FreeBSD's core (http://www.freebsd.org/cgi/cvsweb.cgi/src/contrib/one-true-awk/FREEBSD-upgrade?rev=1.9&content-type=text/x-cvsweb-markup) , avaldatud 16. mail 2005, vaadatud 20. septembril 2006
5. ↑ FreeBSD arvamus GPL eelistest ja puudustest (http://www.freebsd.org/doc/en_US.ISO8859-1/articles/bsdl-gpl/)
6. ↑ *Jawk at SourceForge* (http://sourceforge.net/projects/jawk/)
7. ↑ *xgawk Home Page* (http://home.vrweb.de/~juergen.kahrs/gawk/XML/)
8. ↑ QSEAWK at Google Code (http://qse.googlecode.com/)

Välislingid

- http://awk.info/
- http://www.awktutorial.com/
- http://www.gnu.org/software/gawk/manual/gawkinet/
- http://www.ibm.com/developerworks/linux/library/l-awk1.html
- http://www.ibm.com/developerworks/linux/library/l-awk2.html
- http://www.ibm.com/developerworks/linux/library/l-awk3.html
- http://www.computerworld.com.au/article/216844
- http://www.think-lamp.com/2008/10/awk-a-boon-for-cli-enthusiasts/

Välja otsitud andmebaasist "http://et.wikipedia.org/wiki/AWK"
Kategooria: Programmeerimiskeeled
