AWK

David Frey

Einleitung

AWK ist die nach *Aho*, *Weinberger*, und *Kernighan* benannte interpretierte Programmiersprache, die von `sed`, `egrep`, `sh` und *C* inspiriert wurde.

AWKs

Es gibt folgende freie, new `awk`s:

- `mawk` Mike Brennan's AWK. Klein und schnell.
- `gawk` [1] FSF's AWK. Einige Erweiterungen über das POSIX.2-AWK hinaus, z.B. Networking in den neuesten `awk`s.
- `awk` Original *new* `awk` von Aho, Kernighan und Weinberger.

Vorteile

AWK [2, 3] ist unter klassischen Unices — neben sed — das zweite Swiss-Army-Knife mit folgenden Vorteilen:

1. sicher auf jedem System vorhanden[a]
2. einfach zu erlernen (keine verschiedenen Variablenzugriffsmethoden à la Perl)
3. POSIX.2-standardisiert [4]
4. schnell
5. kleinerer Memory-Footprint als `perl`
6. lädt schneller als `perl`

Nachteile

Im Vergleich zu Perl hat AWK folgende Nachteile:

1. nicht die ganze C-Library eingebaut
2. “nur” `egrep` Regular-Expressions
3. kein Networking (Ausnahme gawk ≥ 3)
4. keinen so coolen Namen

[a] **Achtung:** auf Sun `nawk` resp. `/usr/xpg4/bin/awk`

Syntax

AWK ist nach dem Schema

pattern { *action* }

aufgebaut.

Falls das '*pattern*' wahr ist, wird die '*action*' ausgeführt; eine leere '*action*' ist gleichbedeutend mit '**{ print }**', ein leeres '*pattern*' mit **true**.

'*pattern*' können *bool'sche Ausdrücke* oder `egrep`-*Regular Expressions* sein.

Die Syntax der '*action*' ist C-ähnlich, d.h. Unix-Programmierer werden sich sofort daheim fühlen.

I/O-Paradigma

Input

`awk` trennt den Input in *Fields* und *Records* auf.

Fields sind durch den *Field Separator* (`FS`) getrennt. Die Anzahl der eingelesenen Felder befindet sich in der *Number of Fields*-Variable, `NF`.

Records sind durch den *Record Separator* (`RS`) getrennt. Die Anzahl der eingelesenen Records befindet sich in der *Number of Records*-Variable, `NR`.

Sowohl `FS` als auch `RS` können in neueren `awk`'s *Regular expressions* sein.

Output

Analog zu Input:

`OFS` wird nach jedem Feld eingefügt ('␣'), `ORS` nach jedem Record ('`\n`').

Flow-Control

if (*expr*) *statement* [**else** *statement*]
while (*expr*) *statement*
do *statement* **while** (*expr*)
for ([*expr*] ; [*expr*] ; [*expr*]) *statement*
for (*var* **in** *array*) *statement*
continue, **break**

eingebaute Funktionen

gsub, **index**, **length**, **match**, **split**, **sprintf**, **sub**, **substr**, **tolower**, **toupper**

atan2, **cos**, **exp**, **int**, **log**, **rand**, **sin**, **sqrt**, **srand**

benutzerdefinierte Funktionen

Neue `awk`s kennen Subroutinenaufrufe, *functions* genannt. Einziger Schwachpunkt ist die ein bisschen komische Deklarationsart für lokale Variaben: sie sehen aus wie zuviel übergebene Parameter und werden konventionsgemäss von diesen mit einem Tabulator getrennt.

Variablen

AWK kennt 2 Variablentypen: *Strings* und *Zahlen*.

Wie eine Variable interpretiert wird, ist kontextabhängig. Folgende AWK-Idiome erzwingen Zahlen resp. Strings:

- `var + 0`
- `var ""`

Selbstverständlich steht *sprintf* zur Verfügung.

Arrays

Unter AWK sind alle Arrays von Haus aus *assozativ*, was die Bildung von Listen und Hashes erlaubt.

Beispiel: gpg2callgraph

```
#!/usr/bin/awk -f

BEGIN { FS=":" }

function trim(a) {
        gsub(/ +<[^>]+>/,"",a);
        gsub(/[^A-Za-z0-9@ ]+/,"",a);
        gsub(/ +/,"_",a);
        return a;
}

$1 ~ /^pub/        { if (pubno > 0) { printf(")\n"); }
                     # Enforce a lone node.
                     printf("%s: ",trim($10));
                     printf("(%s:",trim($10));
                     pubno++;
                   }
$1 ~ /^sig/ && $10 !~ /\[User id not found\]/ {
                     if (!sigs[$10])
                       printf(" %s",trim($10));
                     sigs[$10]++;
                   }
END                { if (pubno > 0) { printf(")\n"); } }
```

Zusammenfassung

`awk` ist für kleine, schnelle und effiziente Programme gedacht, welche häufig als Glue-Programmen in Pipelines verwendet werden. Dies sind häufig Einzeiler.

Kleine Konversionsprogramme sind häufig, z.B. pgp/gpg-Output ins NoSQL-Format.

Man hört zwar, dass Buchhaltungen, Assembler und Troff-Clones in `awk` geschrieben wurden, aber dies ist nicht der ursprünglich gedachte Verwendungszweck.

Literatur

[1] Arnold D. Robbins. *Effective AWK Programming.* Specialized Systems Consultants, Inc., 7723 24th N.W., Seattle, WA 98117, USA, 1.0 edition, January 1996. ISBN 0-916151-88-3.

[2] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *Awk — A Pattern Scanning and Processing Language (Second Edition)*, Seventh edition, September 1978.

[3] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language.* Addison Wesley Longman Publishing Company, Inc., Reading Massachusetts, MA 01867, USA, October 1988. ISBN 0-201-07981-X.

[4] IEEE, editor. *Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (Volume 1).* Information technology—Portable Operating System Interface (POSIX). IEEE Computer Society, 345 East 47th St, New York, NY 10017, USA, December 1993. ISBN 1-55937-406-3.