NAME

csvmode – direct processing of Comma-Separated-Values (CSV) data files with gawk.

SYNOPSIS

```
@include "csv"
BEGIN { CSVMODE = 1 }
  ... rules with $0, $1, ... $NF, CSVRECORD, ...

csvfield(name, default)
csvprint(record, option...)
csvprint0()
```

DESCRIPTION

The *gawk-csv* extension can directly process CSV data files. Uses some specific variables:

CSVMODE

Setting **CSVMODE=1** lets CSV formatted input data records to be automatically converted to regular awk records with fixed field separators, and delivered as **$0**. And **$1** .. **$NF** are also set accordingly. Setting **CSVMODE=0** disables the conversion, and input files are processed the usual way. See NOTE 1.The conversion can be customized by some control variables:

CSVFS

The resulting field separator, that temporarily overrides the `FS` and `OFS` predefined variables. If not set, a null char '\0' is used. See NOTE 1.

CSVCOMMA

The input CSV field delimiter. Default comma ','.

CSVQUOTE

The input CSV quoting character. Default double quote '"'.

CSVRECORD

The original CSV input record.

If the CSV file has a header record, the fields can also be accessed by name:

csvfield(*name* **[,** *missing***])**

Returns the named field of the current record. If there is no column named *name*, then return *missing*, or a null value if not given.

csvprint([*record***, [***fs* **[,** *comma* **[,** *quote***]]]])**

A convenience function to format and print the given record with a single call. If called without arguments it prints either $0 formatted as CSV or `CSVRECORD`, depending on `CSVMODE`. Arguments are like `csvformat()`.

csvprint0()

A convenience function to print the original input record as such. Prints either $0 or `CSVRECORD`, depending on `CSVMODE`.

`CSVMODE`, `CSVFS`, `CSVCOMMA` and `CSVQUOTE` are checked only at `BEGINFILE` time. Changing them in the middle of a file processing takes no effect.

`CSVRECORD` is updated for each CSV input record.

The CSV input mode accepts fields with embedded newlines, tabs and other control characters, except null characters ('\0').

EXAMPLES

Extract CSV records with some specific value in the second field:

```
BEGIN {CSVMODE = 1}
$2=="some value" {print CSVRECORD}
```

Process CSV files with fields separated by semicolons instead of commas:

```
BEGIN {CSVMODE = 1; CSVFS = ";"}
```

```
  ... processing rules ...
```

Print a specific named field of every record:

```
BEGIN {CSVMODE = 1;}
{ print csvfield("City") }
```

Print records that contain commas as data, in both normal and CSV modes:

```
grepcommas.awk:
BEGINFILE {
    CSVMODE = (FILENAME ~ /\.csv$/)
}
/,/ { csvprint0() }

Sample invocation:
gawk -f grepcommas.awk a.txt, b.csv, c.txt
```

NOTES

(1) If the user code has a `BEGINFILE` action that sets `CSV`*-mode variables depending on the current file, this action must appear before the* `@include "csv"` *clause:*

```
BEGINFILE {
    CSVMODE = (FILENAME ~ /\.csv$/)  # switch mode depending on the file type
}
@include "csv"
```

BUGS

Null characters are not allowed in fields. A null character terminates the record processing.

AUTHOR

Manuel Collado, **m-collado@users.sourceforge.net**.

SEE ALSO

XML Processing With gawk, *csvparse(3am)*.

COPYING PERMISSIONS