`invoice 0.4`
A Package for Writing Invoices

Oliver Corff

December 6th, 2001

Contents

1 Introduction

The `invoice` package was conceived in late 2000 when the author had to dig through a truly aweful pile of expense bills without having a spreadsheet featuring LaTeX 2_ε-compliant output (or any spreadsheet, for that purpose) available. After several miscalculations with a pocket calculator due to forgotten entries or double entries the idea came up to have LaTeX 2_ε do the calculation work. As such, the package in its present stage is highly specialized with regard to the documents it generates. The `invoice` package is basically a tailor-made solution for a consultant who charges fees and claims all sorts of expenses, sometimes working on different assignments for the same client.

The author expresses his gratitude to Robert Inder, Thilo Barth, Jacco Kok, Fred Donck, Ian Wormsbecker, Vincent Tougait and Robin Fairbairns who contributed ideas, corrections, bugfixes and caption translations after the first discussions on `comp.text.tex` and the initial release of `invoice`.

Given the current capabilities of `invoice`, it should well be possible to extend the capabilities of this package in the future or to rewrite it in a generalized fashion.

2 Software Requirements

The `invoice` environment runs under LaTeX 2_ε and relies on the `calc.sty` (providing infix arithmetic) and `realcalc` (providing real arithmetic) utilities to do its work which can be found at CTAN[1]. Compile and read `00README.tex` for further information if you are not sure whether these packages are installed at your site.

3 The `invoice` Environment

Within a given document, invoices are built with the `invoice` environment[2]. Figure 1 on page 3 shows the logical structure of an invoice as well as its basic commands.

Invoices contain one or more projects which in return contain the charged items, either fees (plus tax, if applicable) and/or expenses. An invoice with one project is announced by saying

```
\begin{invoice}{<Base Currency>}{<VAT>}
\ProjectTitle{...}%
\end{invoice}
```

There is no limit for the number of projects in an invoice, as there is no limit for the number of invoices per document.

The `invoice` environment requires two arguments:

1. `<Base Currency>` is the name of the currency in which the invoice is charged, e. g. DM, Euro, US$, RMB etc.

2. `<VAT>` is the amount of VAT which is charged; in Germany this is currently (autumn 2001) 16%. If no VAT is required, enter a `0` (*zero*) here. It is neither necessary nor permissible to use a percent sign here. As some countries (e. g. France) have fractions of percentages (like 16.9%), you would in this case enter `16.9` (without any percent sign).

3.1 Projects

An invoice contains items which are usually, in the case of e. g. consultancy fees and related expenses like hotel bills and air fares, attributed to a given case or *project*, or “Kostenstelle” (in German).

A project contains any of three different types of charged items:

1. **Fees**. A tax can be added, if applicable. Fees are always charged in the base currency of the invoice.

[1]The `realcalc` package is found at `CTAN:macros/generic/realcalc`, and `calc` is found at `CTAN:macros/latex/required/tools/`.

[2]Users of the KOMA-Script class `scrlettr.cls` are kindly requested to use `invoiceenv` instead; see also page 8.

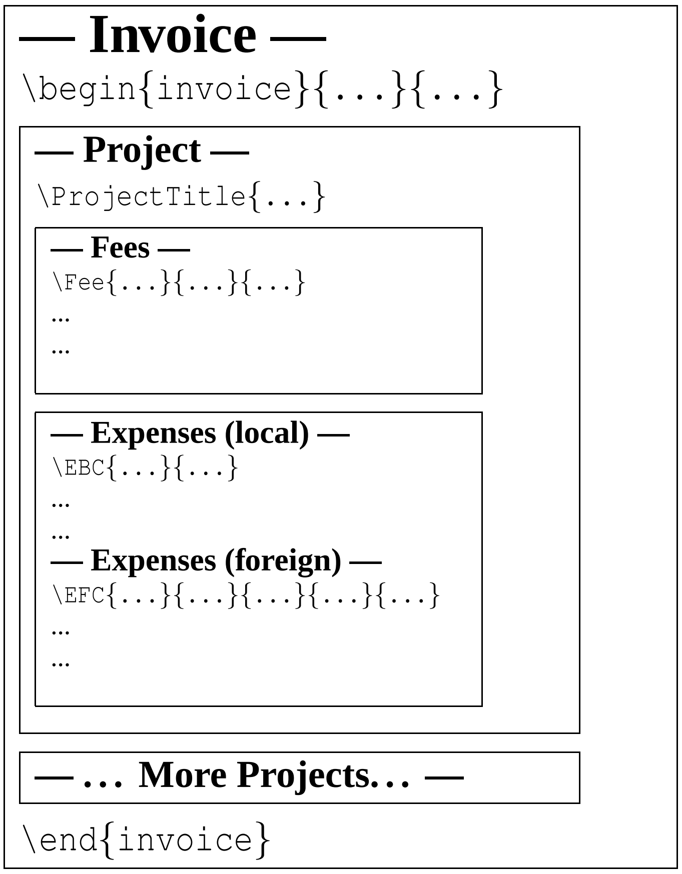

Figure 1: The `invoice` Environment and its Logical Structure

2. **Local Expenses**. Local expenses are charged in units of the base currency of the invoice.
3. **Foreign Expenses**. Foreign expenses are charged in units of any given foreign currency. Either the base currency equivalent is known (as taken from a credit card billing statement, for example), or, if not, an exchange rate between foreign currency and base currency has to be stated.

Nota bene: The order of fees and expenses is fixed. Either fees or expenses can be omitted, but expenses must be charged *after* fees.

4 The First Example: How to Charge Fees

A consultant charges fees per day, hour or any other unit. Usually this unit is agreed upon in a contract and there is no further need to refer to this unit but by its count. This is done by the `\Fee{}{}{}` command:

```
\Fee{<Contents>}{<Rate/Unit>}{<Count>}
```

Let’s assume an interim manager charges DM 1818.00 a day for 12 working days while negotiating a major project, nicknamed *Project Phenix*. He also charges DM 2750.00 a day for the analysing and negotiating the restructuring of the sales division, a work he spent 9 days with. This would be input in the invoice as follows:

```
\begin{invoice}{DM}{16}
   \ProjectTitle{Project Phenix}%
       %    Contents                    Rate/Unit   Count
       \Fee{Some really lengthy and utterly
        tedious egotiation}     {1818.00}   {12}

   \ProjectTitle{Sales Restructuring}%
       %    Contents                    Rate/Unit   Count
       \Fee{Sales Structure Analysis}   {2750.00}   { 6}
       \Fee{Negotiation with Agents}    {2750.00}   { 3}
\end{invoice}
```

And here is how the result looks like:

Project Phenix

Activity	Rate/Unit	Count	Amount (DM)
Some really lengthy and utterly te-dious negotiation	1818.00	12	21816.00
VAT (16%)			3490.56
Subtotal Project			25306.56

Sales Restructuring

Activity	Rate/Unit	Count	Amount (DM)
Sales Structure Analysis	2750.00	6	16500.00
Negotiation with Agents	2750.00	3	8250.00
Subtotal Fees			24750.00
VAT (16%)			3960.00
Subtotal Project			28710.00
Sum Fees			46566.00
Sum VAT			7450.56
Total			**54016.56**

Hints: If the base currency is to contain a dollar sign ($), then dollar sign must be entered in the form of `\string$` or otherwise the command writing the log file data will fail. The contents of each fee may be verbose; while the column width is limited, text contents longer than the column width wraps over several columns, if necessary.

The astute observer will note that a line beginning with “Subtotal Fees” appeared in the output of the Sales Restructuring Project without explicit input to this effect from the user's side. The full grammar of the Fee block requires that all fees are closed by a fee subtotal. Internally, `invoice` is defined as a finite state automaton providing mechanisms to insert a fee subtotal if logic requires it, and print its value if it makes sense to humans, which is the case if there is more than one fee.

Note that there is an explicit command `\STFee` which will produce a subtotal of the fees charged so far. This can be used if you want to show fee subtotals within the same project.

5 The Second Example: How to Claim Expenses

Expenses can be charged in base currency or in any foreign currency. The base currency's name should be announced once at the beginning of the invoice.

5.1 Expenses in Base Currency

The shape of an expense item in base currency is simple:

```
\EBC{<Contents>} {<Amount>}
```

Both fields contain mandatory arguments:

1. `<Contents>` contains a description of the charged item, e. g. "Hotel", "Airport Tax" or whatever.
2. `<Amount>` contains the amount in base currency units.

5.2 Expenses in Foreign Currency

Charging an expense in foreign currency is only slightly more complicated. The command is:

```
\EFC{<Contents>}
    {<Foreign Currency>}{<Amount>}
    {<Conversion Rate>}{<Base Currency Result>}
```

Arguments to the five fields are partially mandatory, partially optional:

1. `<Contents>` contains a description of the charged item, e. g. "Hotel", "Airport Tax" or whatever.
2. `<Foreign Currency>` contains the name of the foreign currency.
3. `<Amount>` contains the amount in foreign currency units.
4. `<Conversion Rate>` contains the factor by which the foreign currency amount has to be multiplied in order to achieve the base currency result. If the base currency result is stated, then, and only then, the Conversion Rate can be omitted.
5. `<Base Currency Result>` contains an optional amount in base currency units. Credit card billing statements show this amount which usually contains certain service charges of the credit card issuer; the base currency result is thus the true amount of money to be charged. If a `<Conversion Rate>` is given, stating a base currency result becomes optional. This is usually applied for expenses made with cash money.

Since some of the arguments given to `\EFC` are optional, there are basically two different forms of using this command. With the variant

```
\EFC{<Contents>}{<Foreign Currency>}{<Amount>}
        {<Conversion Rate>}              % Conversion rate
        {}                               % Base currency empty!
```

(amount in foreign currency given, as well as exchange rate stated), the command will automatically calculate the resulting amount in base currency.

```
\EFC{<Contents>}
        {<Foreign Currency>}{<Amount>}
        {}                               % Conversion rate empty!
        {<Base Currency Result>}         % Base currency
```

If, however, the exchange rate is omitted and the target amount in base currency is given, then this value is taken directly. Stating the resulting amount overrides the internal calculation mechanism. Examples are given below. We use our interim manager's invoice again, assuming this time that she spent working on Project Phenix 12 days in her home country while the Sales Restructuring effort took her to Hong Kong. The taxi bills are paid in cash, hence she enters the (fictive) conversion rate, whereas the hotel is paid by credit card. She can then take the final amount from her credit card billing statement; an example input would look as follows:

```
\begin{invoice}{DM}{16}
   \ProjectTitle{Project Phenix}%
       %    Contents                    Rate/Unit   Count
       \Fee{Negotiation}                {1818.00}   {12}
       %
       %    Contents                    Amount
       \EBC{Hotel, 12 nights}           {2400.00}

   \ProjectTitle{Sales Restructuring}%
       %    Contents                    Rate/Unit   Count
       \Fee{Sales Structure Analysis}   {2750.00}   { 6}
       \Fee{Negotiation with Agents}    {2750.00}   { 3}
       %
       %    Contents              Currency  Amount  Conv.Rate Result
       \EFC{Taxi Airport -- Hotel} {HK\$}  {325.00} {0.2354}   {}
       \EFC{Hotel, 9 nights}       {HK\$}  {9180.00}   {}    {2111.40}
\end{invoice}
```

And here is how the result looks like:

Project Phenix

Activity		Rate/Unit	Count	Amount (DM)
Negotiation		1818.00	12	21816.00
VAT (16%)				3490.56
Expense	Currency	Amount	Factor	DM

Hotel, 12 nights	DM			2400.00
Subtotal Project				27706.56

Sales Restructuring

Activity	Rate/Unit	Count	Amount (DM)
Sales Structure Analysis	2750.00	6	16500.00
Negotiation with Agents	2750.00	3	8250.00
Subtotal Fees			24750.00
VAT (16%)			3960.00

Expense	Currency	Amount	Factor	DM
Taxi Airport – Hotel	HK$	325.00	0.2354	76.50
Hotel, 9 nights	HK$	9180.00		2111.40
Subtotal Expenses				2187.90
Subtotal Project				30897.90

Sum Fees	46566.00
Sum VAT	7450.56
Sum Expenses	4587.90
Total	**58604.46**

Again, a subtotal of the expenses appears only if there is more than one expense item in a project.

6 Postprocessing

In order to allow the further processing of `invoice`-generated data, the log file contains the totals of fees, expenses and taxes in the form of `key:value` pairs. This information appears also on the terminal while LaTeX 2ε runs. Please note that the key appearing in the log file is expressed in the same language as that of the master document.

7 Document Language

All column headers appearing in `invoice` can be redefined in order to match the language of the master document. E. g., headings like “Total” will automatically appear as “Gesamtsumme” if the document language is set to German (either via `babel` or `\usepackage{german}`).

Please observe that the `invoice` package must be called *after* the document language has been selected.

This is correct:

```
\documentclass[10pt]{ltxdoc}
```

```
\usepackage{german}
\usepackage{invoice} % labels will now appear in German!
```

This will not work:

```
\documentclass[10pt]{ltxdoc}
\usepackage{invoice}
\usepackage{german} % labels will still appear in English!
```

7.1 Adding Labels in a New Language

At present, column labels produced by `invoice` can appear in four languages: German, English, Dutch and French. Other languages can be added easily by editing the file `invoice.def`. Language-dependent definitions are contained in the `\if`clause. Extending the file is simple:

1. Copy the labels template to the end of the file.
2. Remove the comments (`%`) in column 1.
3. Fill each label definition with the appropriate foreign language translation.
4. Put the correct internal name of the foreign language into the condition expression of the `\ifx`- and `\ifnum`-clauses.

Whenever you create your own foreign language extension please kindly consider sending your modified `invoice.def` file to the author (at `corff@zedat.fu-berlin.de`) so that it can be included in the next update. Future users can then share your work.

8 Bugs

Certainly there are bugs. After all, this is not Mars, but Earth (where life is supposed to exist). The author has not conducted extensive tests on the algebraic functions, and there may be rounding and truncating errors.

The author considers it a bug that the decimal point cannot be replaced by a comma at present, while ntering dollar signs requires the `\string$` notation.

Spurious spaces may distort the table layout. It is especially recommended to close all `\ProjectTitle{...}%` lines with a percent mark, as shown. If the percent mark is omitted, the first column header is not properly aligned to the left.

Breaking long invoice statements with many projects over several pages results in many orphans and widows.

8.1 Compatibility with KOMA-Script

Unfortunately, when first writing this package the author was not aware of the KOMA-Script classes, notably the class `scrlettr.cls` which defines its own `\invoice{#1}` command, albeit with completely different usage and syntax. KOMA-Script's `\invoice{#1}` command accepts a number and prints "Invoice no. #1" in a letter opening. Now that the names of the `invoice` package and environment have stuck it is easier to think of a workaround than to conceive a new name. Therefore, If the class `scrlettr.cls` is used, the environment `invoice` is automatically renamed to `invoiceenv`. Writing

```
\documentclass[10pt]{scrlettr}
\usepackage{invoice}
\begin{invoice}{DM}{16}
...
...
```

will result in an error message. Say `\begin{invoiceenv}{DM}{16}` instead. KOMA-Script's own `\invoice{#1}` command is renamed to `\invoiceno{#1}`, while `\invoice{#1}` now generates an error message reminding KOMA-Script users to apply the renamed commands.

9 Desiderata

The `invoice` package is far from complete. Future expansions should aim at making the package more flexible. How taxes are treated, etc., is at present a rather particular arrangement suitable for the author's immediate needs, but a more general solution can be designed as soon as more users reflect their needs to the author.