Usage of \LaTeX\ module \texttt{luaindex} and \texttt{luaindex.sty} for Generating Indexes

Markus Kohm*

v0.1b

With \LaTeX it would not be a problem to call an index processor like MakeIndex while running \LaTeX. So the user would not longer require to call the index processor on his own. But on the other side Lua hat enough power to process the index itself. Package \texttt{luaindex} was made to do this. It consists primary of a Lua module: \texttt{luaindex.lua}. This provides functions to generate a new index (or several new indexes), add entries to it and print the index. To make the world easier there’s an additional \LaTeX package: \texttt{luaindex.sty}.

Contents

1 Idea

2 General Options

3 Generating Index Entries

4 Print an Index

5 Known Issues

6 Implementation of Lua Module \texttt{luaindex.lua}

\*komascript@gmx.info
1 Idea

We will explain this in a future release.

2 General Options

See implementation documentation.

3 Generating Index Entries

See implementation documentation.

4 Print an Index

See implementation documentation.

5 Known Issues

Currently the user documentation is not existing. Please use the implementation documentation and the example instead of. This will be changed in a future release but maybe not at a near future.

Currently there are no attributes to give the different indexes different headings. You may redefine \indexname before printing an index to do so. Future releases will do this simply by option.

Currently repeated pre-sort-replaces are not supported. Maybe they will in a future release.

Currently page ranges are not supported. They will in a future release.

Note: This is not even a beta version. It’s only a proof of concept. Almost everything my be designed and implemented in a better kind. The author himself is just learning Lua\TeX.

Nevertheless you may report bugs and patches to komascript@gmx.info.
6 Implementation of Lua Module luaindex.lua

First of all we define a new module named luaindex. All variables and functions will be local to this module.

```lua
module("luaindex", package.seeall)
```

To handle all indexes we have a variable named indexes. This is a table of index tables associated by the name of the index table:

- Each index table has at least two elements associated to `presortreplaces` and `sortorderbychar`.
- There may be additional numerically associated elements, the index entries.
  - Each index entry has at least two elements associated to `sort` and `value`. Element `sort` is the sort key of the index entry. Element `value` is the print value of the index entry.
  - Each index entry may have an element associated to `pages`. This is a table of print values, that will be used as page number of the entry. It need not to be numeric. This table has numeric associations. Later added pages will be appended to the end of the table.
  - Each index entry may have an element associated to `subindex`. This is an index table too, but do not have elements `presortreplaces` or `sortorderbychar`.

```lua
indexes={
    name={
        presortreplaces={
            {[pattern]=replace, ...}, ...
        },
        sortorderbychar={
            [char]=position, ...
        },
        sort="...",
        value="...",
        pages={...},
        subindex={...}
    }
}
```

Next we have a function to generate a new index table at indexes:

```lua
newindex(index name)
```

```lua
local indexes = {}
newindex( indexname )
indexes[indexname]={
    presortreplaces = {},
    sortorderbychar = {} }
end
```

The function parameter is the name of the index. This is not really a print name, but a simple association name.

Don’t be impressed because of empty initialization of `presortreplaces` and `sortorderbychar`. We will have functions to change this.

First of all, we have a function to add a new sort order.

```lua
sortorder(index name, sort-order)
```

```lua
function sortorder( indexname, sortorder )
    local i, value
    The first parameter of the function is the name if the index table. If an index table with the given name does not exist, \TeX should release an error message with some optional help.
    local index = indexes[indexname]
    if index == nil then
    ```
You've tried to add a new sortorder to an index, but there's no index with the given name.
You should define the index using Lua function "luaindex.newindex(" .. indexname .. ")"," before.

The second parameter of the function may be a string. The string simply is a concatenation of the character in the order that should be used to sort the index entries of this index. The index table association `sortorderbychar` is a table. The characters are the association and the wanted sort position is the associated value.

```
local value
i = 1
repeat
  value = unicode.utf8.sub( sortorder, i, i )
  print( i, value )
  if value then
    index.sortorderbychar[value] = i
  end
  i = i + 1
until value == ""
```

The second parameter of the function may also be a table with numerical associations.

```
for i, value in ipairs( sortorder ) do
  index.sortorderbychar[value] = i
end
```

Function `presortreplace(index name, pass, pattern, replace)` is to add presort entries to a presort pass of an index. `pattern` and `replace` are strings. See Lua function `unicode.utf8.sub` for more information about these.

```
function presortreplace( indexname, pass, pattern, replace )
local n
```

The first parameter of the function is the name of the index table. If an index table with the given name does not exist, TeX should release an error message with some optional help.

```
local index = indexes[indexname]
if index == nil then
tex.error( "Unknown index " .. indexname .. ",",
  { "You've tried to add a new presort-replace to an index, but the index with the same name doesn't exist." } )
else
  if type(sortorder) == "string" then
    local value
    i = 1
    repeat
      value = unicode.utf8.sub( sortorder, i, i )
      print( i, value )
      if value then
        index.sortorderbychar[value] = i
      end
      i = i + 1
    until value == ""
  else -- should be table
    The second parameter of the function may also be a table with numerical assoziationes.
    ```
    for i, value in ipairs( sortorder ) do
      index.sortorderbychar[value] = i
    end
    ```
```

Second manipulation function is to add presort entries to a presort pass of an index. `pattern` and `replace` are strings. See Lua function `unicode.utf8.sub` for more information about these.
"with the given name."
"You should define the index using lua function ",
  = `'luaindex.newindex('%' .. indexname .. '%');'",
  "before."
}

else

if the index exists, we have to create replace tables for every pass until the
given.
for n = table.maxn(index.presortreplaces), pass, 1 do
  if ( index.presortreplaces[n] == nil ) then
    index.presortreplaces[n] = {}
  end
end

Last but not least we have to add a new replace to the pass:
index.presortreplaces[pass][pattern]=replace  

Indexes are normally separated into single letters, all numbers and all
other symbols. To do so, we have a new function that returns 1 for all
other symbols, 2 for all numbers and 3 for all letters. Whether an UTF-8
character is a letter or not depends on the locale type “collate”. You may
set it using os.setlocale("locale", "collate").

local function getclass( utfc )
  local i
  for i in unicode.utf8.gmatch( utfc, "%n" ) do
    ⟨debug⟩
    print( utfc .. " is a number" )
    return 2
  end
  for i in unicode.utf8.gmatch( utfc, "%a" ) do
    ⟨debug⟩
    print( utfc .. " is a letter" )
    return 3
  end
  ⟨debug⟩
  print( utfc .. " is a symbol" )
  return 1
end

Before printing or sorting we may want to replace some strings. We have
a table of those. At the string each occurrence of the association should be
replaced by the associated value.
local function do_presortreplaces( srcstr, presortreplace )
  if presortreplace then
    local pat, rep
    for pat, rep in pairs( presortreplace ) do
      srcstr = unicode.utf8.gsub( srcstr, pat, rep )
    end
  end

end
local printsubindex( level, index, presortreplace_zero )

Now let’s print the index. There aren’t much differences in printing an index or a sub-index to an index entry. We only need to know the level of the (sub-) index. level 0 is the main index.

local function printsubindex( level, index, presortreplace_zero )
local i,t,n,p,l
local group=""
local class=-1

We build the \TeX index item command: \item, \subitem, \subsubitem etc. depending on the level. So level is simply the number of sub at the index item command.

local item="\"
for l = 1, level, 1 do
  item = item .. "sub"
end
item = item .. "item \\
Walk through all index items.
for i,t in ipairs( index ) do
If level is 0, we are at the root index. We want to group this Index into numbers, symbols and single letters. To do so, we detect the class of the first character at the sort string and add \indexgroup commands if neccessary.

if ( level == 0 ) then
  local sort=do_presortreplaces( t["sort"], presortreplace_zero )
  local firstchar=unicode.utf8.upper( unicode.utf8.sub( sort, 1, 1 ) )
  if ( firstchar ~= group ) then
    local newclass
    The character differ, but we have to print the group only if the groups of the characters differ.
    newclass=getclass( firstchar )
    if ( newclass == 1 and class ~= newclass ) then
      tex.print( "\indexgroup{\symbolsname}" )
    elseif ( newclass == 3 ) then
      tex.print( "\indexgroup{\charname}" )
    elseif ( newclass == 2 and class ~= newclass ) then
      tex.print( "\indexgroup{\numbersname}" )
    end
    group=firstchar
    class=newclass
  end
end
Now we have to print the index item. We use the value to be printed. If one or more pagenumbers are stored, we print them too. If the index entry has a sub index, we call printsubindex for this one with increased level.
108  tex.sprint( item, t["value"] )
109  if t["pages"] then
110      tex.sprint( "\\indexpagenumbers{" )
111      for n,p in ipairs( t["pages"] ) do
112          tex.sprint( "\\indexpagenumber{", p, "}" )
113      end
114      tex.print( "}" )
115  end
116  if t["subindex"] then
117      printsubindex( level+1, t["subindex"], presortreplaces_zero )
118  end
119  end
120 end

printindex(index name)  Printing a whole index is simply the same like printing a sub index, but
before printing the index, we have to test, wether the named index exists
or not.
121 function printindex( indexname )
122     local index=indexes[indexname]
123     if index == nil then
124         tex.error( "Unknown index `", indexname, ": You've tried to print an index, but there's no index with the
given name.", "You should define the index using lua function \texttt{luaindex.newindex("", indexname, \texttt{"\")',
125             "before."
126         )
127     } )
128     else
129         print( "Index: \"", indexname, ": \" \" . table.maxn( index ) \" . " )
130         tex.print( "\\begin{theindex}" )
131         printsubindex(0,indexes[indexname],indexes[indexname].presortreplaces[0])
132         tex.print( "\\end{theindex}" )
133      end
134 end

local getsubclass(utf8-char)  To sort the index character classes numbers, letters and other are not
enough. So we build sub-classes inside these three classes.
139 local function getsubclass( utfc )
140     local i
141     Inside letters we want so sort upper case before lower case.
142     for i in unicode.utf8.gmatch( utfc, "%l" ) do
143         return 1
144     end
145     for i in unicode.utf8.gmatch( utfc, "%u" ) do
146         return 2
147     end
148 end

Inside other symbols we want so sort controls before spaces before punctuations before numbers before unknown.
for i in unicode.utf8.gmatch(utfc, "%c") do
    return 1
end
for i in unicode.utf8.gmatch(utfc, "%s") do
    return 2
end
for i in unicode.utf8.gmatch(utfc, "%p") do
    return 3
end
for i in unicode.utf8.gmatch(utfc, "%n") do
    return 4
end
return 10 -- unkown is the biggest sub class
end

To compare two UTF8-strings we could simply use the string compare of Lua. But for our purpose this is not enough. So we've added a configurable sort order and now have to compare character by character depending on this sort order.

local function do_strcmp( first, second, sortorderbychar )
  local secondtable = string.explode( second, "" )
  local firstutf
  local n = 1
  (debug) print( first .. ", " .. second );
  for firstutf in string.utfcharacters( first ) do
    local secondutf = unicode.utf8.sub( second, n, n )
    n = n + 1;
    if firstutf then
      if secondutf ~= "" then
        (debug) print( " " .. firstutf .. ", " .. secondutf )
        if firstutf ~= secondutf then
          local firstn, secondn
          if sortorderbychar then
            firstn = sortorderbychar[firstutf]
            secondn = sortorderbychar[secondutf]
          end
          if firstn and secondn then
            (debug) print( " n: " .. firstn .. ", " .. secondn )
            if firstn < secondn then
              return -1
            elseif firstn > secondn then
              return 1
            end
          else
            end
        else
          end
      end
    end
  end
end

If both characters were in the sort order table with different index we may return -1, if the index of first was lower than second, and 1, if the index of first was higher than second.
If one character was not in the sort order table, we compare the classes and
if same the sub-classes.

```lua
local firstclass = getclass( firstutf )
local secondclass = getclass( secondutf )
if firstclass < secondclass then
    return -1
elseif firstclass == secondclass then
    local firstsubclass = getsubclass( firstutf)
    local secondsubclass = getsubclass( secondutf )
    if firstsubclass < secondsubclass then
        return -1
    elseif firstsubclass == secondsubclass then
        if firstutf < secondutf then
            return -1
        else
            return 1
        end
    end
else
    return 1
end
else
    return 1
end
```

If the first string was longer than the second, it is greater.

```
return 1
end
else
```

If the first string was shorter than the second, it is lower.

```
if seconduf ~= "" then
    return -1
else
    return 0 -- This should never happen!
end
```

If the first string was shorter than the second, it is lower. If not they are
same.

```
if unicode.utf8.sub( second, n, n ) ~= "" then
    return -1
else
    return 0
end
```

Now we are able to compare the sort value of two index entries. Before

```lua
local do_indexcmp( first string, second string, replace tables, sort order table)
```
the first compare we do the first pre-sort replace. All other pre-sort replaces will be done only, if the sort entries are not same!

```lua
local function do_indexcmp( firstsort, secondsort, 
    presortreplaces, sortorderbychar )
local pass = 0
local ncmp = 0
repeat
    if presortreplaces and presortreplaces[pass] then
        firstsort = do_presortreplaces( firstsort, presortreplaces[pass] )
        secondsort = do_presortreplaces( secondsort, presortreplaces[pass] )
        (debug) print( "Replace-Pass " .. pass .. ", " .. firstsort .. ", " .. secondsort .. ")
        end
        pass = pass + 1
        ncmp = do_strcmp( firstsort, secondsort, sortorderbychar )
    until ( ncmp ~= 0 ) or ( pass > table.maxn( presortreplaces ) )
    (+debug)
    if ncmp < 0 then
        print( firstsort .. "<" .. secondsort )
    elseif ncmp == 0 then
        print ( firstsort .. "=" .. secondsort )
    else
        print( firstsort .. ">" .. secondsort )
    end
    return ncmp
end
```

**local subinsert( index table, replace tables, sort order table, page string, sort value, print value, ... )**

Inserting a new entry to an index is same like inserting a new entry to a sub-index of an already existing entry. So we have only one local function for this. A new entry consists of a page string, that should be added to the page list of the entry, a sort value, that should be used to find the correct entry and a print value, that should be shown at the index. Entries are only same, if the compare of the sort value is 0 and the print values are same. A new entry may be not only a new entry to the top level but to sub levels. Because of this, there may be several pairs of sort- and print values. We use bisection search to find the insert position.

```lua
local function subinsert( index, presortreplaces, sortorderbychar, 
    pagestring, sortvalue, outputvalue, ... )
local min = 1
local max = table.maxn(index)
local updown = 0
local n = math.ceil(( min + max ) / 2)
while min <= max do
    updown = do_indexcmp( sortvalue, index[n].sort, 
        presortreplaces, sortorderbychar )
    if updown == 0 then
```
The sort values are compared to be same (after serveral replaces). But only if the print values are (without any replaces) same, we have to use this entry. In this case we add a new sub-entry to this entry and if no new sub entry was given the page string to the page table.

```plaintext
if outputvalue == index[n].value then
  print( "The entries are same." )
  if ( ... ) then
    print( " Adding subentry to already existing entry" )
    if ( index[n].subindex == nil ) then
      index[n].subindex = {}
    end
    subinsert( index[n].subindex, presortreplaces, sortorderbychar,
     pagestring, ... )
  end
else
  print( " Is the pagestring already at the pages table?" )
  local i, p
  for i, p in ipairs( index[n].pages ) do
    if pagestring == p then
      print( "The pagestring is already at the pages table." )
      print( " We have nothing to do." )
      return
    end
  end
end
print( pagestring, "!=" , p )
end
print( " The pagestring was not at the pages table," ),
"Add the new pagestring to the pages table",
"and stop processing." )
end
table.insert( index[n].pages, pagestring )
end
else
  print( "The entries are not same.",
"Search for the last entry, with same sort." )
  repeat
    n = n + 1
    if n <= max then
      updown = do_indexcmp( sortvalue, index[min].sort,
       presortreplaces, sortorderbychar )
    end
    until n > max or updown == 0
    min = n
    max = n-1
  end
else if updown > 0 then
  min = n+1
end
```
else
    max = n - 1
end
n = math.ceil((min + max) / 2)

(debug) print(min, max, n)
end

if we have a new sub entry we add this to the new position. If not we simply add the new entry with the page table.
if (...) then
    (debug) print("Generating new entry without page but subindex")
table.insert(index, n,
    { sort=sortvalue, value=outputvalue, subindex={} })
    (debug) print("Add subindex to new generated entry")
    subinsert(index[n].subindex, presortreplaces, sortorderbychar,
    pagestring, ... )
else
    (debug) print("Generating new entry with page")
table.insert(index, n,
    { sort=sortvalue, value=outputvalue, pages={pagestring} })
end
end

We’ve explained before, that inserting a new entry is same like inserting an entry to a sub entry. There’s only one tiny difference: the replace tables and sort order are members of the index table.

function insert(indexname, pagestring, sortvalue, outputvalue, ...)
    local index=indexes[indexname]
    subinsert( index, index.presortreplaces, index.sortorderbychar,
    pagestring, sortvalue, outputvalue, ... )
end

Last we will need a function, that only removes all index entries but not presortreplaces or sortorderbychar.

function removeentries(indexname)
    local p = indexes[indexname].presortreplaces
    local s = indexes[indexname].sortorderbychar
    indexes[indexname]={
        presortreplaces = p,
        sortorderbychar = s }
end

7 Implementation of \LaTeX Package \texttt{luaindex.sty}

The \LaTeX package is user’s candy but not necessary. You may use \texttt{luaindex.lua} directly, but \LaTeX users will expect a \LaTeX interface.

7.1 Package Startup

\texttt{LuaLaTeX} must be used to use the package.
We need some LuaTEX primitives:
\directlua{\luatexbase@ensure@primitive{luaescapestring}}

We need some Lua functions:
\directlua{\if not tex.error then\luatexbase.module_error('luaindex',\'undefined function!\string\n\ LuaTeX function tex.error() needed but not defined.\string\n\ Maybe you are using the wrong version of LuaTeX.'\)\end\if\if not tex.print then\luatexbase.module_error('luaindex',\'undefined function!\string\n\ LuaTeX function tex.print() needed but not defined.\string\n\ Maybe you are using the wrong version of LuaTeX.'\)\end\if\if not tex.sprint then\luatexbase.module_error('luaindex',\'undefined function!\string\n\ LuaTeX function tex.sprint() needed but not defined.\string\n\ Maybe you are using the wrong version of LuaTeX.'\)\end\if\}}

Load an initialize lua module. We could do this much later, but it is very, very important, so we do so as soon as possible.
\RequireLuaModule{luaindex}

With luaindex we use a temporary index file, too. This is necessary, because page numbers are only valid while output routine. So usage of a temporary index file is a good solution to have correct page numbers. If this file exists, we load it simply while \begin{document} and then produce a new one. But loading the old one is not simply an \input. Out temporary index file is a Lua file, so we use Lua function dofile to load it.
\newwrite\@indexfile\AtBeginDocument{\%
7.2 Options

We use a key-value interface even for options. Because of this we’re using KOMA-Script package `scrbase`.

```latex
\RequirePackage{scrbase}
\DefineFamily{luaindex}
\DefineFamilyMember{luaindex}
\newcommand*{\luaindex@sortorder}{\protect\ref{luaindex@sortorder}}
\DefineFamilyKey{luaindex}{sortorder}{\edef\luaindex@sortorder{#1}}
```

**locale**  
If no individual sort order is given, the `collate` locale would cause the sort order. So we add an option make this locale changable. Note, that changing this locale may also affect to other Lua functions!

```latex
\DefineFamilyKey{luaindex}{locale}{\if@atdocument\expandafter\@firstofone\else\AtBeginDocument\fi\protect\write\@indexfile{}{\os.setlocale\@protect\ref{locale}\@protect\ref{collate})\protect\fi\@protect\write\@indexfile{}{}}
```

**pageformat**  
The page format is an attribute of every index entry. But you may define a primary page format to be used, if no individual page format will be given.

```latex
\newcommand*{\luaindex@pageformat}{\protect\ref{luaindex@pageformat}}
\DefineFamilyKey{luaindex}{pageformat}{\def\luaindex@pageformat{#1}}
```

**singlepass**  
This option changes the general behavior of `\printindex`. See definition of `\printindex` for more information about.

```latex
\FamilyBoolKey{luaindex}{singlepass}{\@luaindexsinglepass}
```
Processing all the options while loading the package.
\setupluaindex This is only an convenience command for run time setup of luadindex options.

7.3 Some Usual Index Commands
\see \seealso are common commands used at the page number format. They are defined for compatibility.
\see and \seealso are used by \see and \seealso and needed to be defined also.

7.4 Generation of Indexes and Index Entries
\newindex We can handle not only one index but several indexes. To do so, we have to create a new lua index table for each index. Just use

\newindex{(index name)}

to do so. Additional features may be set up using:

\newindex[(index options)]{(index name)}

Currently all global options are supported for (index options), but some will be ignored.

\newcommand*{\newindex}[2][2]{}% \directlua{luaindex.newindex('\luatexluaescapestring{#2}')}% \begingroup \setupluaindex[#1] % \ifx\luaindex@sortorder@empty\else \AtBeginDocument{ % \protected@write@indexfile{}{% \luaindex@sortorder('\luatexluaescapestring{#2}', \luaindex@sortorder') }% \fi }% \endgroup

You may use \newindex at the document preamble only.

\@onlypreamble\newindex

15
This command will be used to add a new root level entry to an index:

\luaindex{⟨index name⟩}{⟨options⟩}{⟨entry⟩}

⟨index name⟩ – the name of the index to be used. This has to be the same like you’ve used to create the new index using \newindex.

⟨options⟩ – several options for the index entry. Currently supported are:

locale=⟨locale specifier⟩ – just calls \luaindexsetup{⟨locale specifier⟩}.

Note, that this is a global action!

pageformat=⟨command⟩ – is a command with at most one argument to format the page number of the index entry. You may, e.g., use sort=\see{⟨reference⟩} or sort=\seealso{⟨reference⟩} to produce a “see” or “see also” cross reference to ⟨reference⟩ instead of showing a real page number.

sort=⟨sort entry⟩ – destines the sort position of the index entry. If it is omitted ⟨entry⟩ will be used instead.

⟨entry⟩ – this will be shown in the index.

Note: An index entry is only same, if ⟨sort entry⟩ is same (after several presort replaces) and ⟨entry⟩ is same. Index entries with same ⟨sort entry⟩ but different ⟨entry⟩ will be placed at the current end of the entries with same ⟨sort entry⟩.

413 \newcommand*{\luaindex}[1]{%
414 \@bsphack
415 \begin{group}
416 \edef\luaindex@name{#1} %
417 \lua@index
418 }
419 \newcommand*{\lua@index}[2][]{%
420 \set@display@protect
421 \edef\luaindex@sort{#2} %
422 \define@key{luaindex.setindex}{sort}{\edef\luaindex@sort{##1}}%
423 \define@key{luaindex.setindex}{pageformat}{\def\luaindex@pageformat{##1}}%
424 \define@key{luaindex.setindex}{locale}{\luaindexsetup{locale=#1}}%
425 \setkeys{luaindex.setindex}{#1} %
426 \protected@write\@indexfile{\let\luatexluaescapestring\relax}{%
427 \luaindex.insert(',\luatexluaescapestring{\luaindex@name}',
428 '{\luatexluaescapestring{\luaindex@pageformat{\thedpage}}}',
429 '{\luatexluaescapestring{\luaindex@sort}}',
430 '{\luatexluaescapestring{#2}}')
431 )%
432 \endgroup
433 \@esphack
434 }
Same like \texttt{\luaindex} but to produce a sub entry:

\begin{verbatim}
\luaindex{⟨index name⟩}{⟨options⟩}{⟨entry⟩}{⟨options⟩}{⟨sub-entry⟩}
\end{verbatim}

Note, that the \texttt{⟨options⟩} for the \texttt{⟨sub-entry⟩} only allows a sub-set of the options shown for \texttt{\luaindex}. Currently only \texttt{sort=⟨sort entry⟩}.

\begin{verbatim}
\newcommand*{\luaindex}{[1]{%\@bsphack\begingroup\edef\luaindex@name{#1}\lua@subindex}}
\newcommand*{\lua@subindex}{[2][]{%\set@display@protect\edef\luaindex@sort{#2}\%\define@key{luaindex.setindex}{sort}{\edef\luaindex@sort{##1}}\%\define@key{luaindex.setindex}{pageformat}{\edef\luaindex@pageformat{##1}}\%\define@key{luaindex.setindex}{locale}{\luaindexsetup{locale=#1}}\%\setkeys{luaindex.setindex}{#1}\%\protected@write\@indexfile{\let\luatexluaescapestring\relax}{\luaindex.insert{\luatexluaescapestring{\luaindex@name},}'\luatexluaescapestring{\luaindex@pageformat{\the\thepage}},'\luatexluaescapestring{\luaindex@sort}','\luatexluaescapestring{#2}',\@spaces}\%\aftergroup\lua@subindex\endgroup}}
\end{verbatim}

Same like \texttt{\luaindex} but to produce a sub-sub-entry, that is a sub-entry to a sub-entry:

\begin{verbatim}
\luaindex{⟨index name⟩}{⟨options⟩}{⟨entry⟩}{⟨options⟩}{⟨sub-entry⟩}{⟨options⟩}{⟨sub-sub-entry⟩}
\end{verbatim}
Note, that the \textit{options} for the \textit{sub-entry} and the \textit{sub-sub-entry} only allows a sub-set of the options shown for \texttt{\index}. Currently only \texttt{sort=\textit{sort entry}}.

\newcommand*{\luasubsindex}[1]{\%}
\@bsphack
\begingroup
\edef\luaindex@name{#1}\%
\lua@subsubindex
\}
\newcommand*{\lua@subsubindex}[2][{}]{\%}
\set@display@protect
\edef\luaindex@sort{#2}\%
\define@key{luaindex.setindex}{sort}{\edef\luaindex@sort{##1}}\%
\define@key{luaindex.setindex}{pageformat}{\def\luaindex@pageformat{##1}}\%
\define@key{luaindex.setindex}{locale}{\%
\luaindexsetup{locale=#1}\%
\setkeys{luaindex.setindex}{#1}\%
\protected@write\@indexfile{\let\luatexluaescapestring\relax}{\%
\luaindex.insert('{\luatexluaescapestring{\luaindex@name}',
\'
\luatexluaescapestring{\luaindex@pageformat{\thechapter}',
\'
\luatexluaescapestring{\luaindex@sort}',
\'
\luatexluaescapestring{#2}',
\}
\aftergroup\lua@@@subindex
\endgroup
\newcommand*{\lua@@@subindex}[2][{}]{\%}
\begingroup
\set@display@protect
\edef\luaindex@sort{#2}\%
\define@key{luaindex.setindex}{sort}{\edef\luaindex@sort{##1}}\%
\setkeys{luaindex.setindex}{#1}\%
\protected@write\@indexfile{\let\luatexluaescapestring\relax}{\%
\@spaces
\luatexluaescapestring{\luaindex@sort}',
\'
\luatexluaescapestring{#2}',
\}
\aftergroup\lua@@subindex
\endgroup
\makeindex
\index
\subindex
\subsubindex

These are defined to increase compatibility to old index packages only. Command \texttt{\makeindex} simply generates the new index named \texttt{general} and the other commands to add entries to that index. Note, that adding a sub-entry or sub-sub-entry is not yet compatible to other index packages. You need to use the command \texttt{\subindex} and \texttt{\subsubindex} instead of something like \texttt{\index{⟨entry⟩!⟨sub-entry⟩!⟨sub-sub-entry⟩}}. Note also,
Table 1: Implications of option `singlepass` to `\printindex`

<table>
<thead>
<tr>
<th>singlepass=\text{false}</th>
<th>singlepass=\text{true}</th>
</tr>
</thead>
<tbody>
<tr>
<td>index of previous LuaLaTeX run will be printed</td>
<td>index of current LuaLaTeX run will be printed</td>
</tr>
<tr>
<td>start of index depends on the class</td>
<td>start of the index at next page earliest</td>
</tr>
<tr>
<td>index entries may be added to an index even after it has been printed</td>
<td>no more index entries may be added to the index after it has been printed</td>
</tr>
</tbody>
</table>

that changing the format of the page number is not compatible with other index packages. You have to use `\index[\text{page format}=⟨page format⟩]{...}` instead of something like `\index{⟨entry⟩|⟨page format⟩}`.

\begin{verbatim}
\renewcommand*{\makeindex}{%
  \newindex{general}%
  \renewcommand*{\index}{\luaindex{general}}%
  \newcommand*{\subindex}{\luasubindex{general}}%
  \newcommand*{\subsubindex}{\luasubsubindex{general}}%
}%
\end{verbatim}

\section*{7.5 Printing an Index}

We do not only want to create an index, we also need to print it.

\begin{verbatim}
\printindex With
\end{verbatim}

`\printindex[⟨options⟩]`

you can print an index. The known options are

\begin{description}
\item[index=⟨index name⟩] – print the index with the given name as declared at `\newindex`. If you omit this option, index “general” will be printed.
\item[singlepass=⟨boolean value⟩] – you may switch on and off the single pass feature. For the differences of single pass feature on and off, see table 1
\end{description}

\begin{verbatim}
\newcommand*{\\printindex}[1]{%\begingroup
  \edef\luaindex@name{general}%
  \define@key{\luaindex.setindex}{index}{\edef\luaindex@name{##1}}%
  \define@key{\luaindex.setindex}{singlepass}{true}%
  \setkeys{\luaindex.setindex}{#1}%
}%
\end{verbatim}

19
luaindex.lua uses several macros while printing the index. First of all it uses the environment \textit{theindex}. But several additional macros will be used:

\begin{itemize}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexgroup}}}}}}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexspace}}}}}}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$symbolsname}}}}}}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$numbersname}}}}}}}
\end{itemize}

Each index is grouped. Index groups are symbols, numbers and each first letter. Each group starts with \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexgroup}{$\langle$group$\rangle$}}}}}} with group is either \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$symbolsname}}}}}}, \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$numbersname}}}}}}}} or a upper case letter. In difference to other index processors no automatic \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexspace}}}}}}} will be added before each group. So we define \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexgroup}}}}}}} to add it.

\begin{itemize}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$providemacro{$\langle$indexgroup$\rangle$\{1\}{\%}}}}}}}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$indexspace$\backslash$textheft$\{1\}$\{\noexpand\pagebreak\}}}}}}}}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\texttt{\texttt{$\backslash$def$\backslash$indexspace$\{\vskip\baselineskip\}}}}}}}}
\end{itemize}

The page numbers of an entry are printed all together as argument
of \indexpagenumbers\{(page number)\}. Each single page number is printed as argument of \indexpagenumber\{(page number)\}. So separate the single page numbers \indexpagenumber is predefined to add internal macro \index@pagenumbersep before the page number. This will add \indexpagenumbersep before each page number but the first one.

\providecommand*{\indexpagenumbers}[1]{% 
def\index@pagenumbersep{\let\index@pagenumbersep\indexpagenumbersep}% 
nobreakspace-- #1}% 
\providecommand*{\indexpagenumber}[1]{\index@pagenumbersep #1}% 
\providecommand*{\indexpagenumbersep}{, }

8 Examples

Currently only one example file will be produced:

\texttt{luaindex-example} – This should show index entries, index sub-entries, index sub-sub-entries.

\documentclass{article}
\usepackage[ngerman]{babel}
\usepackage{blindtext}
\usepackage{fontspec}

We load package \texttt{luaindex} with option \texttt{locale=de\_DE}. At least at Linux this will add Ä, Ö, Ü, ä, ö, ü, and ß to the letters and even set a valid sort order for those.

We load package \texttt{luaindex} with option \texttt{singlepass} to produce a valid index with one Lua\TeX{} run instead of two or more. But with this printing of the index will produce a new page.

\texttt{\usepackage[locale=de\_DE, singlepass % Wenn der Index ohnehin eine neue Seite produziert, % dann kann er direkt beim ersten Lauf ein korrektes % Ergebnis liefern. ]{luaindex}}

We use the compatibility command \texttt{\makeindex} to generate the “general” index and the further compatibility commands, e.g., \texttt{\index}.

\makeindex

We want \texttt{\textbf} to be ignored at the sort:

\directlua{luaindex.presortreplace('general',0, 
    '\luatexluaescapestring{\string\textbf}\space\string\{(\string\^\string\}\})}

Now we can start our document. This consist of some text and several index entries.
Now, let’s do something different. Let’s show that babel shorthands may be used inside index entries:

C\index{C ist "der" dritte Buchstabe}
X\index{X ist der drittletzte Buchstabe}

And macros may also be used but change the sort sequence of the index!

D\index{D ist der Buchstabe nach C}
Y\index{Y ist der \textbf{vorletzte} Buchstabe}
Z\index{Z ist der letzte Buchstabe}
A\index{Ä ist auch ein Buchstabe}

We may change the sort sequence manually by adding the sort option. The page number format may also be changed using the pageformat option.

Ä\index{Ä ist aber auch ein Buchstabe}, %
\textit{Ä ist wirklich auch ein Buchstabe (und hier stimmt die Sortierung nicht -- \textit{aber eigentlich doch})}

Let’s add one more page with some more index entries:

A\index{A ist der erste Buchstabe}
Ae\index{Ae ist kein Buchstabe, sondern zwei}

And now, let’s have some sub-entries and even a sub-sub-entry. One of the sub-entries will become a different sort position and will be marked with an emphasized page number.

Kompliziert\subindex{Diverses}{Untereintrag}
Noch komplizierter\subindex{Diverses}{Übereintrag}
Noch komplizierter\%
subindex{Diverses}{sort=Obereintra,pageformat=\textit}{Untereintrag}
Noch komplizierter%
\subsubindex{Diverses}{Untereintrag}{Unterunteintrag}
That’s enough. Time time to print the index. Remember, that this is already a valid index, because we are using option `singlepass`.

\printindex
\end{document}

### Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>A</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>\alsoname</td>
<td>\newindex</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\index</td>
<td></td>
</tr>
<tr>
<td>\index@pagenumbersep</td>
<td></td>
</tr>
<tr>
<td>\indexgroup</td>
<td></td>
</tr>
<tr>
<td>\indexpagenumber</td>
<td></td>
</tr>
<tr>
<td>\indexpagenumbers</td>
<td></td>
</tr>
<tr>
<td>\indexpagenumbersep</td>
<td></td>
</tr>
<tr>
<td>\indexspace</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>locale</td>
<td></td>
</tr>
<tr>
<td>\lua@@@subindex</td>
<td></td>
</tr>
<tr>
<td>\lua@@subindex</td>
<td></td>
</tr>
<tr>
<td>\lua@subindex</td>
<td></td>
</tr>
<tr>
<td>\lua@subsubindex</td>
<td></td>
</tr>
<tr>
<td>\luaaindex</td>
<td></td>
</tr>
<tr>
<td>\luaindex@pageformat</td>
<td></td>
</tr>
<tr>
<td>\luaindex@sortorder</td>
<td></td>
</tr>
<tr>
<td>\luasubindex</td>
<td></td>
</tr>
<tr>
<td>\luasubsubindex</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\makeindex</td>
<td></td>
</tr>
</tbody>
</table>

### Change History

v0.1
- General: start of new package 1

v0.1b
- General: prefix ‘koma’ removed from Lua module 1

Using package `luatexbase-compat` 13
Using package `luatexbase-modutils` 13