Date Driven Lists

(Examples)

Willi Egger

How can we produce a list with dates in a specific interval over a defined timeframe? The number of columns in the list and their headers should be configurable. The solution is to use the operating system's timestamp. All calculations are performed as a time unit of seconds. This allows for arbitrary time intervals of the dates in the list. This small module is setup as a CLD (ConTEXt Lua Document).

1. Introduction

During the preparations for the ConTEXt meeting 2022, Taco Hoekwater asked me whether the PocketDiary module could also calculate lists of dates in specific intervals. Furthermore it should be possible to configure the number of columns of the list including the column headers. Analyzing the interval question soon revealed that this is not a task for the calendar calculations of the PocketDiary module. The approach for such date-driven lists is to use the `os.timestamp`, which is the representation of a date as the number of seconds since a fixed date, and this depends on the operating system.

It is possible to choose arbitrary intervals. However if we talk about a monthly interval, we must decide what we mean by this: If the interval is 30 days – then the year falls 5 or 6 days short. The year divided into 13 equal periods would result in an interval length of 28 days.

2. Intervals Defined as Modes

The module can be used for the production of different lists. For each list we define a mode. None of the modes are activated.

```
\definemode[weekly][keep]
\definemode[biweekly][keep]
\definemode[monthly][keep]
```

3. General Setups

```
\setupbodyfont[dejavu,ss,12pt]
\setuppagenumbering[location=]
```

```
\setuppapersize[A4,portrait]%[A4,portrait]

\setuplayout
 [topspace=20mm,
  backspace=12mm,
  header=10mm,
  footer=2\bodyfontsize,
  height=middle,
  width=middle]

\setupfootertexts
  [\jobname .pdf]
  [\pagenumber\ / \totalnumberofpages]
```

4. Lua Code

```
\startluacode

  thirddata          = thirddata or { }
  thirddata.calendar = { }
  local calendar     = thirddata.calendar

  local report = logs.reporter("Interval Checks")

  local interval_dates = {}

  function calendar.interval_checks(startdate,stopdate,cols,intval
    )

    local startdate_string = startdate
    local stopdate_string  = stopdate
    local columns = cols
    local intervaldays = intval

    local year,month,day = calendar.date(startdate_string)
    local start_time_stamp = os.time({year=year,month=month,day=
      day})

    --report("Starttimestamp: %s", start_time_stamp)

    local year,month,day = calendar.date(stopdate_string)
    local end_time_stamp = os.time{year=year,month=month,day=day}

    local interval = intervaldays * 24 * 60 * 60

    for i = start_time_stamp, end_time_stamp, interval do
      local interval_date = os.date("%d-%m-%Y",i)
      table.insert(interval_dates,interval_date)
```

```
    report("%s",interval_date)
  end

  calendar.print_interval_checks(columns)
end

function calendar.print_interval_checks(cols)

  local columns = cols

  context.bTABLE({setups="table:interval_check"})
    context.bTABLEhead()
      context.bTR()
        for i = 1,columns do
          context.bTH()
            context.labeltext("c"..i)
          context.eTH()
        end
      context.eTR()
    context.eTABLEhead()
    context.bTABLEbody()
    for k,v in ipairs(interval_dates) do
      context.bTR()
        context.bTD()
          context(v)
        context.eTD()
        for i=1,columns-1 do
          context.bTD()
            context.strut()
          context.eTD()
        end
      context.eTR()
    end
  context.eTABLEbody()
  context.eTABLE()
end

function calendar.date(inputstr)

  --report("Input : %s",inputstr)

  local sep = "%-%s/"
  if sep == nil then
    sep = "%s"
  end
  local t={}
  i=1
```

```
    for str in string.gmatch(inputstr, "([^"..sep.."]+)") do
      t[i] = str
      --report("Actual string %s", str)
      i = i + 1
    end

    --report("Datum strings: %s, %s, %s", t[1],t[2],t[3])

    return t[1],t[2],t[3]
  end
\stopluacode
```

The main function is `calendar.interval_ckecks` with four parameters. The function needs a start date, an end date, the number of columns for the table in the output and the interval in days. The function calculates all dates fitting into the start and end dates based on the epoch time. It converts the timestamp into a human readable date format and puts the values into a table `interval_dates`.

The second function is called from within the first function and gets the number of columns as parameter. This function creates the output table. It is a flexible setup that is based on the number of the columns. Label texts are used for the column heads, and the column heads are repeated at the top of each page.

The third function deals with the preparation of the dates, which are sent to Lua as strings. It extracts year (yyyy), month (m) and day (d). The values are stored in a table. Dates can be entered in ISO format as '2022-8-21' or '2022/8/21'.

5. Macro for Calling the List Generator

The command needs four parameters: start date, stop date, number of columns in the table and interval in days.

```
\define[4]\Checklist{\ctxlua{thirddata.calendar.interval_checks(#1
  ,#2,#3,#4)}}
```

6. Setups for the Output Table

The layout of the output table is organized at the TEX end. If the table design changes in terms of more or fewer columns, this setup has to be adapted:

```
\startsetups table:interval_check
  \setupTABLE[split=repeat]
  \setupTABLE[r][each][height=12mm,align=lohi]
  \setupTABLE[r][1][style=bold,align={lohi,middle}]
  \setupTABLE[c][1][width=0.2\textwidth]
  \setupTABLE[c][2,3,4][width=0.15\textwidth]
  \setupTABLE[c][5][width=0.3\textwidth]
```

```
\stopsetups
```

7. Column Head Texts

The labeltext mechanism has been selected to implement the head texts of the columns. Depending on the design of the output table the names can be adapted, the list can be made longer or shorter according to needs.

```
\setuplabeltext[en][c1=Date]
\setuplabeltext[en][c2=Gas]
\setuplabeltext[en][c3=Electricity]
\setuplabeltext[en][c4=Water]
\setuplabeltext[en][c5=Observation]
```

8. Header Texts of Output

The labeltext mechanism is also used for the header texts. For now, only three intervals are defined, but this mechanism allows us to extend the list of possible header texts easily.

It is also suitable for implementing other languages. Depending on the `\mainlanguage[]` setting, the right labels are picked up and typeset.

```
\setuplabeltext[en][weekly={Weekly Checks}]
\setuplabeltext[en][biweekly={Biweekly Checks}]
\setuplabeltext[en][monthly={Monthly Checks}]

\setuplabeltext[de][weekly={Wochen Checks}]
\setuplabeltext[de][biweekly={Zwei-Wochen-Checks}]
\setuplabeltext[de][monthly={Monatliche Checks}]

\setuplabeltext[nl][weekly={Weekelijkse Checks}]
\setuplabeltext[nl][biweekly={Twee-weekelijkse Checks}]
\setuplabeltext[nl][monthly={Maandelijkse Checks}]
```

9. List Header Text Setup

Each list gets a header text fitting the purpose of the list:

```
\startmode[weekly]
  \setupheadertexts
    [\midaligned{\bfc \labeltext{weekly}}]
    []
\stopmode

\startmode[biweekly]
```

```
  \setupheadertexts
    [\midaligned{\bfc \labeltext{biweekly}}]
    []
\stopmode

\startmode[monthly]
  \setupheadertexts
    [\midaligned{\bfc \labeltext{monthly}}]
    []
\stopmode
```

10. Interval Calendar User File

For each available interval a mode is created. These modes can be activated here:

```
% \enablemode[weekly]
\enablemode[biweekly]
%\enablemode[monthly]
```

We load the module:

```
\usemodule[intervalcalendar]
```

Because the module has a multilingual interface, we need to set the main language:

```
\mainlanguage[nl]
```

The output table can be organized with some flexibility, so the necessary setups are placed in the user's file:

```
\startsetups table:interval_check
  \setupTABLE[split=repeat]
  \setupTABLE[r][each][height=12mm,align=lohi]
  \setupTABLE[r][1][style=bold,align={lohi,middle}]
  \setupTABLE[c][1][width=0.2\textwidth]
  \setupTABLE[c][2,3,4][width=0.15\textwidth]
  \setupTABLE[c][5][width=0.3\textwidth]
\stopsetups
```

The column heads get an identification:

```
\setuplabeltext[en][c1=Datum]
\setuplabeltext[en][c2=Gas]
\setuplabeltext[en][c3=Electricity]
\setuplabeltext[en][c4=Water]
\setuplabeltext[en][c5=Observation]
```

Now we can let ConTEXt and Lua do their job:

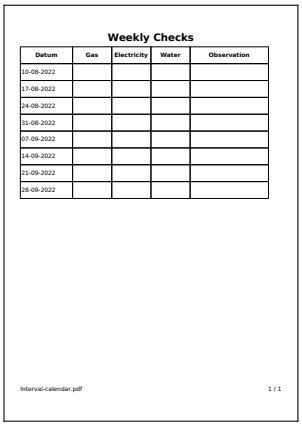
```
\starttext
```

```
\startmode[weekly]
  \Checklist{"2022-8-10"}{"2022-9-30"}{5}{7} %
    startdate,enddate,columns,interval
\stopmode
\startmode[biweekly]
  \Checklist{"2022/9/10"}{"2023/01/31"}{5}{14} %
    startdate,enddate,columns,interval
\stopmode
\startmode[monthly]
  \Checklist{"2022/8/10"}{"2023/05/31"}{5}{30} %
    startdate,enddate,columns,interval
\stopmode

\stoptext
```

11. Results

The information placed in the user file in the previous section gives the following tables:

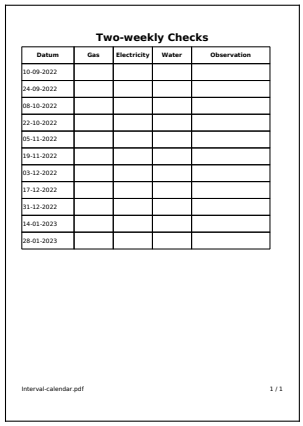


Weekly Checks

Datum	Gas	Electricity	Water	Observation
10-08-2022				
17-08-2022				
24-08-2022				
31-08-2022				
07-09-2022				
14-09-2022				
21-09-2022				
28-09-2022				

interval-calendar.pdf 1 / 1

Weekly

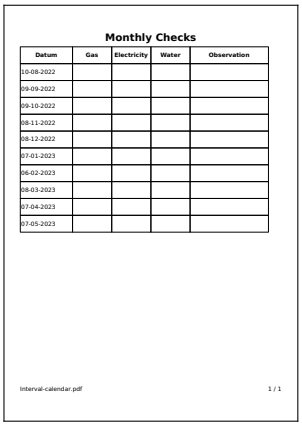


Two-weekly Checks

Datum	Gas	Electricity	Water	Observation
10-09-2022				
24-09-2022				
08-10-2022				
22-10-2022				
05-11-2022				
19-11-2022				
03-12-2022				
17-12-2022				
31-12-2022				
14-01-2023				
28-01-2023				

interval-calendar.pdf 1 / 1

Biweekly



Monthly Checks

Datum	Gas	Electricity	Water	Observation
10-08-2022				
09-09-2022				
09-10-2022				
08-11-2022				
08-12-2022				
07-01-2023				
06-02-2023				
08-03-2023				
07-04-2023				
07-05-2023				

interval-calendar.pdf 1 / 1

Monthly