Betriebssysteme

Entwicklung mit LaTeX und Literate Programming

Slide set 4:
LaTeX and
Literate programming

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Winter semester 2015/16

Dr. Hans-Georg Eßer

hgesser@cs.fau.de

http://ohm.hgesser.de/

v1.1, 10/27/2014

LaTeX: Overview (1)

- LaTeX (pronounced: La-Tech) is a markup language for documents, roughly comparable to HTML
- Two essential syntax elements:
  - Command:

    \ **command** { argument1 } { argument2 } ...
  - Surroundings:

    \ **begin** { *Surroundings* }

    ...

    \ **end** { *Surroundings* }

LaTeX: Overview (2)

- "Hello World" document:

```
\documentclass{article}                  % different classes, e.g. B.% book,
                                         report, letter% start of document%
\begin{document}                         "body"
    Hello, World.
\end{document}                           % End of document
```

- headlines

```
\section{Heading 1}                      % Step 1
text

\subsection{Heading 1.1}                 % Level 2
test

\subsubsection{Heading 1.1.1}            % Level 3
```

LaTeX: Overview (3)

```
% testfile1.tex

\documentclass{article}
\usepackage[latin1]{inputenc}

\begin{document}

\tableofcontents                % Content list

\section{example}

This is a small example. \subsection{Subsection

}

More in the example document \dots

\subsubsection{Heading 1.1.1}

That is how it goes \emph{italic} and \textbf{
fat} .

\end{document}
```

$ **pdflatex testfile1.tex**
$ **pdflatex testfile1.tex**

→ generated testfile1.pdf:

Contents

1 Beispiel

Das ist ein kleines Beispiel.

1.1 Unterabschnitt

Mehr im Beispieldokument ...

1.1.1 Überschrift 1.1.1

So geht es *kursiv* und **fett**.

Comparison LaTeX / HTML (1)

```
\ documentclass { article }
\ usepackage [ latin1] { inputenc }

\ begin { document }

\ tableofcontents                % Content list

\ section { example }

This is a small example. \ subsection { Subsection

}

More in the example document \ dots

\ subsubsection { Heading 1.1.1 }

That is how it goes \ emph { italic } and \ textbf {
fat } .

\ end { document }
```

```
<html>
<head> <meta charset = "iso-8859-1"> </ head>

<body>

<! - There is no content ->

<h1> 1st example </ h1>

This is a small example.

<h2> 1.1 subsection </ h2>

More in the example document ...

<h3> 1.1.1 Heading 1.1.1 </ h3>

This is how it works < i> italic </ i> and
<b> bold </ b>.

</body>
</html>
```

Comparison LaTeX / HTML (2)

```
\ begin { enumerate }
    % Enumeration (1, 2, 3 ...)
    \ item First
    \ item Second
    \ item Third
\ end { enumerate }
```

```
<ol>
    <! - Ordered list ->
    <li> First </ li>
    <li> Second </ li>
    <li> Third </ li>
</ol>
```

1. First
2. Second
3. Third

```
\ begin { itemize }
    % Listing (bullets)

    \ item First
    \ item Second
    \ item Third
\ end { itemize }
```

```
<ul>
    <! - Unordered list,
                bullets ->
    <li> First </ li>
    <li> Second </ li>
    <li> Third </ li>
</ul>
```

- First
- Second
- Third

```
\ includegraphics [ width =
    10 centimeters ] { bild.png }
```

```
<img src = "image.png"
    width = "100px">
```

(Image)

Comparison LaTeX / HTML (3)

First paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph.

Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph.

Text in the third paragraph. Text in the third paragraph. Text in the third paragraph. Text in the third paragraph. Text in the third paragraph.

Hard upheaval \\
New line \\
One more line

<p> First paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. Text in the first paragraph. **</ p>**

<p> Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. Text in the second paragraph. **</ p>**

<p> Text in the third paragraph. Text in the third paragraph. Text in the third paragraph. Text in the third paragraph. Text in the third paragraph. **</ p>**

<p> Hard upheaval **< br>**
New line **< br>**
One more line **</ p>**

LaTeX: commands

- pdflatex: transforms tex- File in pdf- File around
  - → Call it several times if necessary (e.g. for content)
  - → in the event of errors: press [Enter]
  - → repeated errors: type X (and press [Enter])

- xpdf: PDF viewer (Evince, Gnome app)
- bibtex: helps with reference management
  - → used bib- file
  - → more on this later if necessary

LaTeX: Successful execution (1)

ulix @ ulixdevel : ~ / tex $ **pdflatex test2.tex**
This is pdfTeX, version 3.1415926-1.40.10 (TeX Live 2009 / Debian) entering extended mode

(./test2.tex
LaTeX2e <2009/09/24>
Babel <v3.8l> and hyphenation patterns for english, usenglishmax, dumylang,
nohyphenation, loaded. (/usr/share/texmf-texlive/tex/latex/base/article.cls

Document Class: article 2007/10/19 v1.4h Standard LaTeX document class

(/usr/share/texmf-texlive/tex/latex/base/size10.clo))
No file test2.aux.
[1 {/var/lib/texmf/fonts/map/pdftex/updmap/pdftex.map}]
(./test2.aux)) </ usr / sha
re / texmf-texlive / fonts / type1 / public / amsfonts / cm / cmr10.pfb> Output written on
test2.pdf (1 page, 12056 bytes). Transcript written on test2.log.

LaTeX: Successful execution (2)

```
ulix @ ulixdevel : ~ / tex $ ls -l test2. *
- rw-r - r-- 1 ulix ulix                79 Oct 13 22:39 test2.aux
- rw-r - r-- 1 ulix ulix 2488 Oct 13 22:39 test2.log
- rw-r - r-- 1 ulix ulix 32965 Oct 13 22:39 test2.pdf
- rw-r - r-- 1 ulix ulix               112 Oct 13 22:39 test2.tex
- rw-r - r-- 1 ulix ulix                53 Oct 13 22:39 test2.toc
```

- test2.log: Log, with warnings
- test2.pdf: Created PDF file
- test2.toc: Data for table of contents (toc = table of contents)

LaTeX: termination on error

```
ulix @ ulixdevel : ~ / tex $ pdflatex test1.tex
This is pdfTeX, version 3.1415926-1.40.10 (TeX Live 2009 / Debian) entering extended mode

(./test1.tex
LaTeX2e <2009/09/24>
Babel <v3.8l> and hyphenation patterns for english, usenglishmax, dumylang, nohyphenation,
loaded.
(/usr/share/texmf-texlive/tex/latex/base/article.cls
Document Class: article 2007/10/19 v1.4h Standard LaTeX document class
(/usr/share/texmf-texlive/tex/latex/base/size10.clo)) (./test1.aux)

! LaTeX Error: Environment empyhd undefined.

See the LaTeX manual or LaTeX Companion for explanation. Type H <return> for
immediate help.
  . . .

l.4         \ begin {empyhd}

? x
No pa   ges of outpu        t.
Trans   cript writte        n on test1.log.
```

Literate programming

- Basic idea: Don't comment on the code, but rather “tell” how the program works
- Code and documentation swapped
- Example of getting started: Bubblesort

  (Original code taken from http://de.wikipedia.org/wiki/Bubblesort; ported to C)

Bubble sort (1): without comments

```
void bubblesort () {
    int SIZE = 10;
    int i, j, tmp;
    for (j = SIZE; j> 1; j--) {
        for (i = 0; i <j-1; i ++) {
            if (arr [i]> arr [i + 1]) {
                tmp = arr [i + 1];
                arr [i + 1] = arr [i];
                arr [i] = tmp;
            }
        }
    }
}
```

Bubble sort (2): with comments

```
// bubblesort: sort elements of an array arr [] of size SIZE
void bubblesort () {
    // declarations
    int SIZE = 10; // size of the array
    int i, j;                    // loop variables
    int tmp;                     // temporary variable for swapping elements
    // outer loop
    for (j = SIZE; j> 1; j--) {
        // check all neighbors in arr [0..j-1] and swap them if their // order is wrong

        for (i = 0; i <j-1; i ++) {
            if (arr [i]> arr [i + 1]) {
                // swap neighbors i, i + 1 (using tmp as temporary variable)
                tmp = arr [i + 1];
                arr [i + 1] = arr [i];
                arr [i] = tmp;
            }
        }
    }
}
```

Bubblesort as Literate Program (1)

Bubblesort: The Literate Program

To sort a field arr [], we first need to initialize a SIZE variable and declare some local variables, such as i other j which are used as loop counters, as well as a temporary variable tmp:

———

- ***bubblesort: declarations*** 15th • ≡

```
int SIZE = 10;
int i, j, tmp;
```

Bubblesort as Literate Program (2)

The main routine of the bubblesort algorithm compares each field element with its direct (right) neighbor and corrects their order if it is wrong. After doing this once the biggest element will be at the end of the list. It then repeats these steps with a smaller field (ignoring the right-most element). Thus, with each step in a loop, the unsorted array becomes one element smaller until there is nothing left to sort:

• ***bubblesort program*** 16 • ≡

```
void bubblesort () {
  • bubblesort: declarations 15th •
  for (j = SIZE; j> 1; j--) {}
    • bubblesort: check neighbors in range 0 .. j 17th •
}
```

Bubblesort as Literate Program (3)

In order to do the neighbor checks, a second loop inside the outer loop is necessary:

• ***bubblesort: check neighbors in range 0 ..*** j 17th • ≡ _

```
for (i = 0; i <j-1; i ++) {
    if (arr [i]> arr [i + 1]) {
        • bubble location: swap elements 17th •
    }
}
```

For swapping, three commands and usage of a temporary variable are necessary in a C program:

• ***bubble location: swap elements*** 17th • ≡

```
tmp = arr [i + 1];
arr [i + 1] = arr [i];
arr [i] = tmp;
```

Code Chunks (1)

- Elements of form • ***Surname*** • be called **Code chunks.**
- Code chunks, like macros, are replaced where they occur, e.g. B.

```
void bubblesort () {
    • bubblesort: declarations 15th •

    for (j = SIZE; j> 1; j--) {
        . . .
```

```
void bubblesort () {
    int SIZE = 10;
    int i, j, tmp;
    for (j = SIZE; j> 1; j--) {
        . . .
```

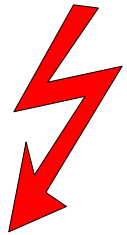
Code Chunks (2)

- Code chunks cannot be recursive:

  • ***nonsense*** 19th • ≡

  ```
  for (i = 0; i <10; i ++) {}
  ```

  • ***nonsense*** 19th •



- Chunk can be defined before or after use

- Chunk can also be completely absent

  → only causes a warning

Code Chunks (3)

- Code chunks can be defined in several places

- later occurrences continue the definition

• ***long definition*** 20th • ≡
```
for (i = 0; i <10; i ++) {
    // first loop
}
```

Text, other chunks, ...

• ***long definition*** 20th • + ≡
```
for (i = 0; i <10; i ++) {
    // second loop
}
```

later becomes:

```
for (i = 0; i <10; i ++) {
    // first loop
}
for (i = 0; i <10; i ++) {
    // second loop
}
```

≡ vs. + ≡!

LaTeX and NoWeb (1)

Syntax in LaTeX documents:

- << chunk name >> For • ***chunk name*** •
- Definition: << name >> = ... @
- [[variable]] For variable _____

```
<< bubblesort program >> =
void bubblesort () {
    << bubblesort: declarations >>
    for (j = SIZE; j> 1; j--) {
        << bubblesort: check neighbors in range 0 .. [[j]] >>}

}
@
```

LaTeX and NoWeb (2)

Complete Literate Program in LaTeX / NoWeb:

\ subsubsection {Bubblesort: The Literate Program}

To sort a field, we first need to initialize a [[SIZE]] variable and declare some local variables, such as [[i]] and [[j]] which are used as loop counters, as well as a temporary variable [[tmp]]:

<< bubblesort: declarations >> =
int SIZE = 10;
int i, j, tmp;
@

The main routine of the bubblesort algorithm compares each field element with its direct (right) neighbor and corrects their order if it is wrong. After doing this once the biggest element will be at the end of the list. It then repeats these steps with

a smaller field (ignoring the right-most element). Thus, with each step in a loop, the unsorted array becomes one element smaller until there is nothing left to sort:

LaTeX and NoWeb (3)

```
<< bubblesort program >> =
void bubblesort () {
    << bubblesort: declarations >>
    for (j = SIZE; j> 1; j--) {
        << bubblesort: check neighbors in range 0 .. [[j]] >>}

}
@
```

In order to do the neighbor checks, a second loop inside the outer loop is necessary:

```
<< bubblesort: check neighbors in range 0 .. [[j]] >> = for (i = 0; i <j-1; i ++) {

    if (arr [i]> arr [i + 1]) {
        << bubblesort: swap elements >>
    }
}
@
```

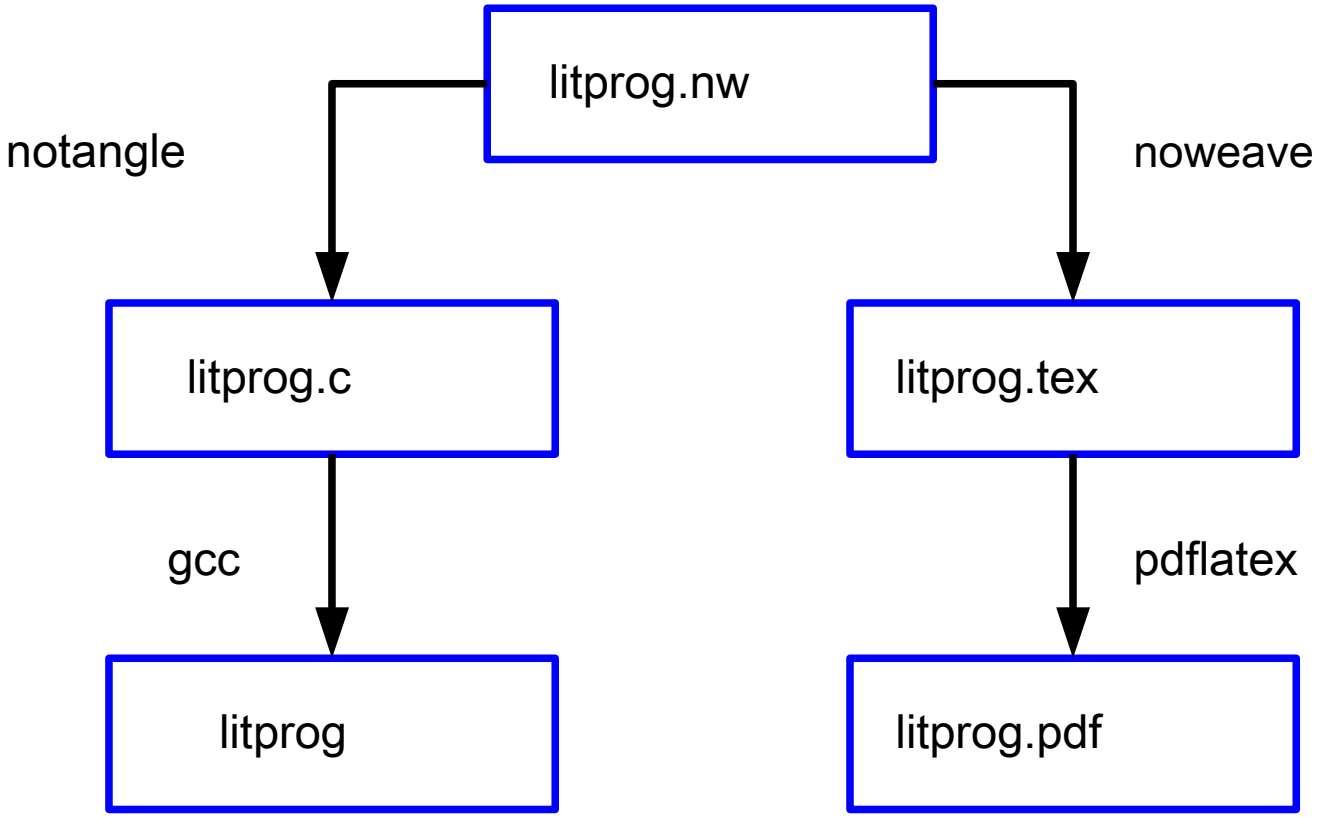
LaTeX and NoWeb (4)

For swapping, three commands and usage of a temporary variable are necessary in a C program:

```
<< bubblesort: swap elements >> =
tmp = arr [i + 1];
arr [i + 1] = arr [i];
arr [i] = tmp;
@
```

LaTeX and NoWeb (5)

- NoWeb tools: noweave, notangle



noweave

- Syntax of noweave:

  noweave file.nw> file.tex

- nw- The file always contains a complete literate program

notangle

- Syntax of notangle:

  notangle -R *chunkname* file.nw> code.c

- nw- File contains many code chunks - which you want to extract, insert with the option

  - R *chunkname* firmly.

- Additional option - L: gives information about line numbers in nw- File from (for the compiler)

Example from Ulix Makefile

- Excerpt from ulix / bin-build / Makefile:

```
TEXSRC_FILE = .. / ulix-book.nw
TEXSRC_MODULE_FILE = .. / student.nw


extract:
        notangle -L -Rulix.c $ (TEXSRC_FILE)> ulix.c; true notangle -L -Rprintf.c $
        (TEXSRC_FILE)> printf.c notangle -Rstart.asm $ (TEXSRC_FILE)> start.asm
        notangle -Rulix.ld $ (TEXSRC_FILE)> ulix.ld

        notangle -L -Rmodule.c $ (TEXSRC_MODULE_FILE)> module.c notangle -L
        -Rmodule.h $ (TEXSRC_MODULE_FILE)> module.h
```

(Line numbers only for C files)

Top-down vs. bottom-up (1)

"In computer science, a software development process is called
Top-down, when the design begins with abstract objects that are then concretized; the process is Bottom-up if individual detailed tasks are assumed that are required to handle higher-level processes. "

(Source: http://de.wikipedia.org/wiki/Top-down_and_Bottom-up)

Top-down vs. bottom-up (2)

- Literate programming supports both approaches and also mixed variants
- The order of presentation (the code chunks) determines whether the code is developed top-down or bottom-up

- Example: operating system

Top-down: operating system

- Our operating system has to initialize the memory and the disk and then load and start the shell from disk:

```
<<ulix>> =
    << initialize memory >>
    << initialize harddisk >>
    << load shell program from disk >> << run shell
    >>
@
```

- How can you initialize the memory now?

```
<< initialize memory >> =
    << check available memory >>
    << create initial page table >>
@
```

Bottom-up: operating system

- Our operating system must first initialize the memory; a page table is required for this:

```
<< page table declaration >> =
typedef struct {
    unsigned int present                    : 1; // 0
    unsigned int writeable                  : 1; // 1
    . . .
    unsigned int frame_addr                 : 20; // 31..12
} page_desc;

typedef struct {
      page_desc pds [1024];
} page_table;
@
```