Language C

Université de Lorraine - Télécom Nancy

Omar CHIDA

Chapitre 1

A propos de moi

- Premier ligne de code à l'age de 14 ans.
- Hardline C++ Fanboy : 6 ans de C/C++.
- De nombreux projets dont un moteur de rendu, une application mobile entre autres codés en C/C++.

Organisation

Comment ca va se passer ?

- Cours, exercices, solutions et projets sertont sur Github.
- Serveur Discord dédié pour les questions, aide et autre.
- TD, TP et Projets seront en présentiel.
- N'hésitez pas à m'interrompre à tout moment pour poser des questions.

L’objectif du Tutorat

- Vous familiariser avec la Langage C.
- Connaître les bonnes pratiques de programmation en C.
- Réussir les examens mais ça va aussi plus loin que ça.
- Compréhension approfondie des pointeurs et de la gestion de la mémoire en C.
- Bien comprendre l’outillage (Compilateur, Débogueur, autre).

L’objectif du Tutorat

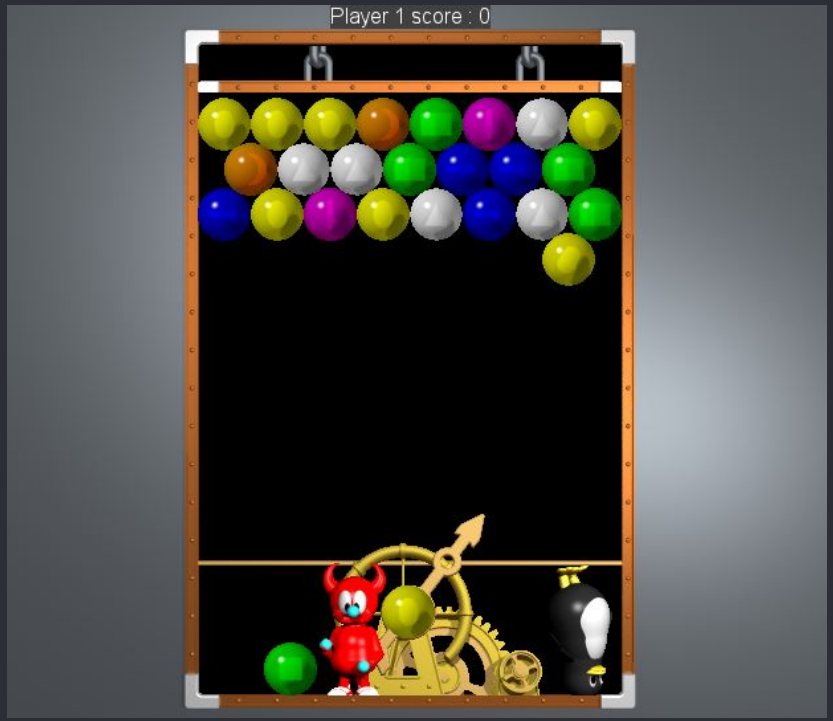
Ce que vous pourrez faire à la fin

```c
#include <stdio.h>

int main()
{
        printf("Hello world!\n");
        return 0;
}
```

```
omar@Omar:~$ gcc main.c
omar@Omar:~$ ./a.out
Hello world!
```

⟹



L'objectif du Tutorat

Ce que nous allons faire ensemble

- Plein d'exercices (même style que les TD).
  - Exercices liés aux structures de données.
  - Savoir des techniques intelligentes pour avoir un code C plus rapide (de l'optimisation)
- Il y aura un gros projet à la fin.
  - Un jeu vidéo du style (Puzzle Bobble ou Mario).
  - Jeu sur le terminal (style Snake).
  - Émulateur de processeur ARM.
  - Quelque chose de plus simple que ça ? (n'hésitez pas à déposer vos idées).

À propos de C

Un peu de connaissances générales

- Langage conçu par Dennis Ritchie et développé par lui et Bell labs.
- Sortie en 1972 (Il y a 49 ans).
- Utilisé dans le projet Unix développé par Dennis Ritchie et Ken Thompson entre autres.
- A vu une évolution relativement petite.
  - K&R C, ANSI C, C99, C11, C17, C2x.
- Aujourd'hui, C est considéré comme un langage de bas niveau.

Motivation : Pouquoi apprendre le C en 2021 ?

C c'est cool !

- C est toujours pertinent et utile aujourd'hui pour beaucoup de choses.
- Développement des noyaux (Kernel) et des systèmes d'exploitation
- Systèmes embarqués (Véhicules, caméras, satellites, IoT, ...)
- Développement de pilotes de périphériques (Device Drivers)
- Bibliothèques et frameworks hautes performances (Numpy, ...)
- Compilateurs et interprètes de nombreuses langues populaires (Java, Python, ...).
- Moteurs de rendu et jeux vidéo.
- Bref... partout où la performance est essentielle.

Chapitre 2

Compilation

- TODO ! ! !

Phase 1 : Preprocessing

Preprocessing

Le Preprocessing (prétraitement) est la première étape du pipeline de compilation. Au cours de laquelle :

- Les commentaires sont supprimés.
- Les macros sont développées.
- Les fichiers inclus sont développés.

Exemple :

Un `#include <stdio.h>` sera remplacé à l'execution de la phase du preprocessing par le contenu du fichier `stdio.h`

Phase 2 : Compiling

La Compilation

La Compilation est la deuxième étape. Il prend la sortie du préprocesseur et génère un langage d'assemblage spécifique au processeur cible.

Exemple :

- La commande "`gcc -S main.c`" arrête le pipeline de compilation avant l'étape d'assemblage.
- Utilisez l'option "`-masm=intel`" pour obtenir l'assembleur en syntaxe Intel.

Phase 2 : Compiling

```c
#include <stdio.h>

int main()
{
        printf("Hello world!\n");
        return 0;
}
```

```
omar@Omar:~$ arm-none-eabi-gcc -S main.c
omar@Omar:~$ cat main.s
```

⟹

```
        .cpu arm7tdmi
        .eabi_attribute 20, 1
        .eabi_attribute 21, 1
        .eabi_attribute 23, 3
        .eabi_attribute 24, 1
        .eabi_attribute 25, 1
        .eabi_attribute 26, 1
        .eabi_attribute 30, 6
        .eabi_attribute 34, 0
        .eabi_attribute 18, 4
        .file   "main.c"
        .text
        .section        .rodata
        .align  2
.LC0:
        .ascii  "Hello world!\000"
        .text
        .align  2
        .global main
        .arch armv4t
        .syntax unified
        .arm
        .fpu softvfp
        .type   main, %function
main:
        @ Function supports interworking.
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 1, uses_anonymous_args = 0
        push    {fp, lr}
        add     fp, sp, #4
        ldr     r0, .L3
        bl      puts
        mov     r3, #0
        mov     r0, r3
        sub     sp, fp, #4
        @ sp needed
        pop     {fp, lr}
        bx      lr
.L4:
        .align  2
.L3:
        .word   .LC0
        .size   main, .-main
        .ident  "GCC: (15:9-2019-q4-0ubuntu1) 9.2.1 20191025 (release) [ARM/arm-9-branch revision 277599]"
```

Phase 3 : Assemblage

L'Assemblage

L'assemblage est la troisième étape de la compilation. L'assembleur convertira le code d'assemblage en code binaire (code machine [1]). Ce code est également appelé code objet.

Exemple :

- La commande "`gcc -c main.c`" arrête le pipeline de compilation à l'étape de l'assemblage.

1. Des zéros et uns

Phase 3 : Assemblage

```
omar@Omar:~$ gcc -c main.c
omar@Omar:~$ hexdump -C main.o
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  01 00 3e 00 01 00 00 00  00 00 00 00 00 00 00 00  |..>.............|
00000020  00 00 00 00 00 00 00 00  10 03 00 00 00 00 00 00  |................|
00000030  00 00 00 00 40 00 00 00  00 00 40 00 0e 00 0d 00  |....@.....@.....|
00000040  f3 0f 1e fa 55 48 89 e5  48 8d 3d 00 00 00 00 e8  |....UH..H.=.....|
00000050  00 00 00 00 b8 00 00 00  00 5d c3 48 65 6c 6c 6f  |.........].Hello|
00000060  20 77 6f 72 6c 64 21 00  00 47 43 43 3a 20 28 55  | world!..GCC: (U|
00000070  62 75 6e 74 75 20 39 2e  33 2e 30 2d 31 37 75 62  |buntu 9.3.0-17ub|
00000080  75 6e 74 75 31 7e 32 30  2e 30 34 29 20 39 2e 33  |untu1~20.04) 9.3|
00000090  2e 30 00 00 00 00 00 00  04 00 00 00 10 00 00 00  |.0..............|
000000a0  05 00 00 00 47 4e 55 00  02 00 00 c0 04 00 00 00  |....GNU.........|
000000b0  03 00 00 00 00 00 00 00  14 00 00 00 00 00 00 00  |................|
000000c0  01 7a 52 00 01 78 10 01  1b 0c 07 08 90 01 00 00  |.zR..x..........|
```

Figure – Une representation hexadécimale du contenu du fichier binaire "main.o"

Phase 4 : Linking

Édition du lien

L'édition du lien est la dernière étape de la compilation. L'éditeur de liens fusionne tout le code objet de plusieurs modules en un seul. Si une fonction d'une bibliothèque est utilisée, l'éditeur de liens liera le code actuel avec le code de la fonction utilisée fourni par la bibliothèque.

N.B :

Il existe deux types de liaison :

- La liaison statique.
- La liaison dynamique.

Phase 4 : Linking

N.B :

Il existe deux types de liaison :

- Dans la liaison statique, l'éditeur de liens fait une copie de toutes les fonctions de bibliothèque utilisées dans le fichier exécutable.
  - Windows : l'extension '`.lib`'
  - Linux & MacOS : l'extension '`.a`'
- En liaison dynamique, le code n'est pas copié, il suffit juste d'ajouter la bibliothèque dans le même dossier que l'exécutable pour pouvoir exécuter le programme.
  - Windows : l'extension '`.dll`'
  - Linux : l'extension '`.so`'
  - MacOS : l'extension '`.dylib`'

Chapitre 3

In the beginning there was main

La fonction main

La fonction main est le point d'entrée du programme [2].

Profiles possibles :

- `int main()`
- `int main(int argc, char** argv)`

Profiles qui compile mais avec un Warning :

- `void main()`
- `void main(int argc, char** argv)`

2. l'exécutable

In the beginning there was main

Les arguments de main

- argc : Indique le nombre d'arguments passés au programme. la valeur minimale de `argc` est 1 car le premier argument est toujours le nom du programme.
- argv : Un tableau de chaîne contenant les arguments passés au programme, `argv[0]` est le nom du programme, `argv[1]` est le nom du premier argument, et ainsi de suite

In the beginning there was main

Exemple :

Soit la commande suivante : `"./a.out abc w 23 1"`

- `argc` : vaut 5
- `argv[0]` : est la chaine "./a.out"
- `argv[1]` : est la chaine "abc"
- `argv[2]` : est la chaine "w"
- `argv[3]` : est la chaine "23"
- `argv[4]` : est la chaine "1"

Les conditions

Syntax : Première possibilité

```
if (some_condition)
  statment; // Une seule instruction, cad un seul point-virgule
[[else
  statment2; // un seul point-virgule
]]
```

N.B :

Ce qui est entre mis entre [[…]] est facultatif

Les conditions

Syntax : Deuxième possibilité

```
if (some_condition1) {
  statment_1;
  // ...
  statment_N;
} [[ else if (some_condition2) {
  statment_1;
  // ...
  statment_N;
// Possibilite d'ajouter plusieurs blocs else if
} ]] [[ else {
  statment_1;
  // ...
  statment_N;
} ]]
```

Les conditions

Comment une condition est évaluée ?

Le type booléen n'existe pas en C. Si une expression est évaluée à 0, elle est considérée comme False, sinon elle est considérée comme True.

Les conditions : Exemple

Example 1 :

```
int i = 0;
if (i--)
  puts("Hello World");
```

Example 2 :

```
int i = -1;
if (i++)
  puts("Hello World");
```

Example 3 :

```
int i = -1;
if (i++)
  if (++i)
    if ('c')
      puts("Hello World");
```

Les conditions : Exemple

Example 1 : (N'affiche rien)

```
int i = 0;
if (i--)
  puts("Hello World");
```

Example 2 : (Affiche "Hello World")

```
int i = -1;
if (i++)
  puts("Hello World");
```

Example 3 : (Affiche "Hello World")

```
int i = -1;
if (i++)
  if (++i)
    if ('c')
      puts("Hello World");
```

Les boucles

Syntax : boucle pour

```
for (initialisation; condition; increment) {
  // Une seule instruction, cad un seul point-virgule
}
```

L'instruction d'initialisation n'est exécutée qu'au début de la boucle. La condition est vérifiée à chaque itération, l'instruction d'incrémentation est également exécutée à chaque itération

Une boucle for peut être écrite comme une boucle while

```
initialisation;
while (condition) {
  // ...
  increment;
}
```

Les boucles

Syntax : boucle pour

Comme la syntaxe du `if`, la boucle pour peut être écrite de cette manière :

```
for (initialisation; condition; increment)
  statment;
```

Example 1 :

```
for (int i = 0; i < 10; i++)
  for (int j = 0; j < 20; j++)
    puts("Hello World");
```

Example 2 :

```
for (;;)
  puts("Hello World");
```

Les boucles

Example 1 : (Affiche 200 "Hello World")

```
for (int i = 0; i < 10; i++)
  for (int j = 0; j < 20; j++)
    puts("Hello World");
```

Example 2 : (Affiche une infinité de "Hello World")

```
for (;;)
  puts("Hello World");
```

Example 3 :

```
for (int i = -1; i < 10; i++) {
  break;
  printf("Hello World\n");
}
```

Les boucles

Example 3 : (N'affiche rien)

```
for (int i = -1; i < 10; i++) {
  break;
  printf("Hello World\n");
}
```

Example 4 :

```
for (int i = -1; i < 10; i++) {
  if (i > 3) continue;
  printf("Hello World\n");
}
```

Example 5 :

```
for (int i = -1; i < 10; i++) {
  continue;
  printf("Hello World\n");
}
```

Les boucles

Example 4 : (Affiche 5 "Hello World")

```
for (int i = -1; i < 10; i++) {
  if (i > 3) continue;
  printf("Hello World\n");
}
```

Example 5 : (N'affiche rien)

```
for (int i = -1; i < 10; i++) {
  continue;
  printf("Hello World\n");
}
```

Les boucles

Syntax : boucle faire ... tantque

```
do {
  // ..
} while(condition);
```

La boucle continue de s'exécuter jusqu'à ce que la condition soit fausse. Il est similaire à une boucle tantque, sauf le fait qu'elle est garantie de s'exécuter au moins une fois.

Les structs

Définition et Syntax :

Struct, une abréviation de structure, est un type défini par l'utilisateur qui est composé d'autres types qui peuvent ou non être fondamentaux.

```
struct struct_name
{
  TypeA field1_name;
  TypeB field2_name;
  TypeC field3_name;
  // ...
};
```

Les structs

Quelques remarques :

- La taille d'une structure est la somme de la taille de ses champs.
- La taille est accessible en utilisant `sizeof(struct_name)`.

Example :

```
struct A
{
  int a; // sizeof(int) = 4
  short b; // sizeof(short) = 2
  double b; // sizeof(double) = 8
  char str[256]; // sizeof(char) = 1 * 256 elements
};
```

La taille est : `sizeof(struct A)` = 4 + 2 + 8 + 256 = 270 octets.

Les unions

Définition et Syntax :

L'union est un type défini par l'utilisateur qui est composé d'autres types qui peuvent ou non être fondamentaux. La mémoire réelle allouée à une union est égale au maximum de ses champs. Tous les champs d'un union partagent donc la même mémoire sous-jacente.

```
union union_name
{
  TypeA field1_name;
  TypeB field2_name;
  TypeC field3_name;
  // ...
};
```

Les unions

Quelques remarques :

- La taille d'une union est le maximum des tailles de ses champs.
- La taille est accessible en utilisant `sizeof(union_name)`.

Example :

```
union A
{
  int a; // sizeof(int) = 4
  short b; // sizeof(short) = 2
  double b; // sizeof(double) = 8
  char str[256]; // sizeof(char) = 1 * 256 elements
};
```

La taille est : `sizeof(union A)` = $max(4, 2, 8, 256) = 256$ octets.

Les unions

ATTENTION : Soyez prudent lorsque vous accédez aux champs d'union. Écrit dans n'importe quel champ d'union peut écraser la mémoire déjà écrite par un autre champ.

```
union B {
  int a;
  short b;
  char str[4];
};
union B var;
var.str[0] = 'T';
var.str[1] = 'N';
var.str[2] = 'C';
var.str[3] = 'Y';
var.b = 256; // ATTENTION: var.str ne vaut plus TNCY !!!
```

Les enums

Chapitre 4

Chapitre 5

Jabberwocky

Lewis Carroll

’Twas brillig, and the slithy toves
Did gyre and gimble in the wabe ;
All mimsy were the borogoves,
And the mome raths outgrabe.

“Beware the Jabberwock, my son !
The jaws that bite, the claws that catch !
Beware the Jubjub bird, and shun
The frumious Bandersnatch !”

Lists and locales

Lorem ipsum dolor sit amet

- Nulla nec lacinia odio. Curabitur urna tellus.
  - Fusce id sodales dolor. Sed id metus dui.
    - Cupio virtus licet mi vel feugiat.

1. Donec porta, risus porttitor egestas scelerisque video.
   1.1 Nunc non ante fringilla, manus potentis cario.
   1.1.1 Pellentesque servus morbi tristique.

Nechť již hříšné saxofony ďáblů rozzvučí síň úděsnými tóny waltzu, tanga a quickstepu! Nezvyčajné kŕdle šťastných figliarskych ďatľov učia pri kótovanom ústí Váhu mĺkveho koňa Waldemara obžierať väčšie kusy exkluzívnej kôry. The quick, brown fox jumps over a lazy dog. DJs flock by when MTV ax quiz prog. “Now fax quiz Jack!”

Text blocks

In plain, example, and alert flavour

This text is highlighted.

A plain block

This is a plain block containing some highlighted text.

An example block

This is an example block containing some highlighted text.

An alert block

This is an alert block containing some highlighted text.

Definitions, theorems, and proofs

All integers divide zero

Definition

$\forall a, b \in \mathbb{Z} : a \mid b \iff \exists c \in \mathbb{Z} : a \cdot c = b$

Theorem

$\forall a \in \mathbb{Z} : a \mid 0$

Proof

$\forall a \in \mathbb{Z} : a \cdot 0 = 0$ □

Numerals and Mathematics

Formulae, equations, and expressions

$$1234567890 \quad 1234567890 \quad \hat{x}, \check{x}, \tilde{a}, \bar{a}, \dot{y}, \ddot{y} \quad \iint f(x, y, z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z$$

$$\frac{1}{1+\cfrac{1}{2+\cfrac{1}{3+x}}} + \frac{1}{1+\cfrac{1}{2+\cfrac{1}{3+x}}}$$

$$F : \begin{vmatrix} F''_{xx} & F''_{xy} & F'_{x} \\ F''_{yx} & F''_{yy} & F'_{y} \\ F'_{x} & F'_{y} & 0 \end{vmatrix} = 0$$

$$\iint_{\mathbf{x}\in\mathbb{R}^2} \langle \mathbf{x}, \mathbf{y} \rangle \,\mathrm{d}\mathbf{x}$$

$$\overline{\overline{a\alpha}^2 + \underline{b\beta} + \overline{\overline{d\delta}}}$$

$$]0, 1[+ \lceil x \rceil - \langle x, y \rangle$$

$$e^x \approx 1 + x + x^2/2! + \\ + x^3/3! + x^4/4!$$

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

Figures

Tables, graphs, and images

Faculty	With TEX	Total	%
Faculty of Informatics	1 716	2 904	59.09
Faculty of Science	786	5 275	14.90
Faculty of Economics and Administration	64	4 591	1.39
Faculty of Arts	69	10 000	0.69
Faculty of Medicine	8	2 014	0.40
Faculty of Law	15	4 824	0.31
Faculty of Education	19	8 219	0.23
Faculty of Social Studies	12	5 599	0.21
Faculty of Sports Studies	3	2 062	0.15

Table – The distribution of theses written using TEX during 2010–15 at MU

Figures

Tables, graphs, and images

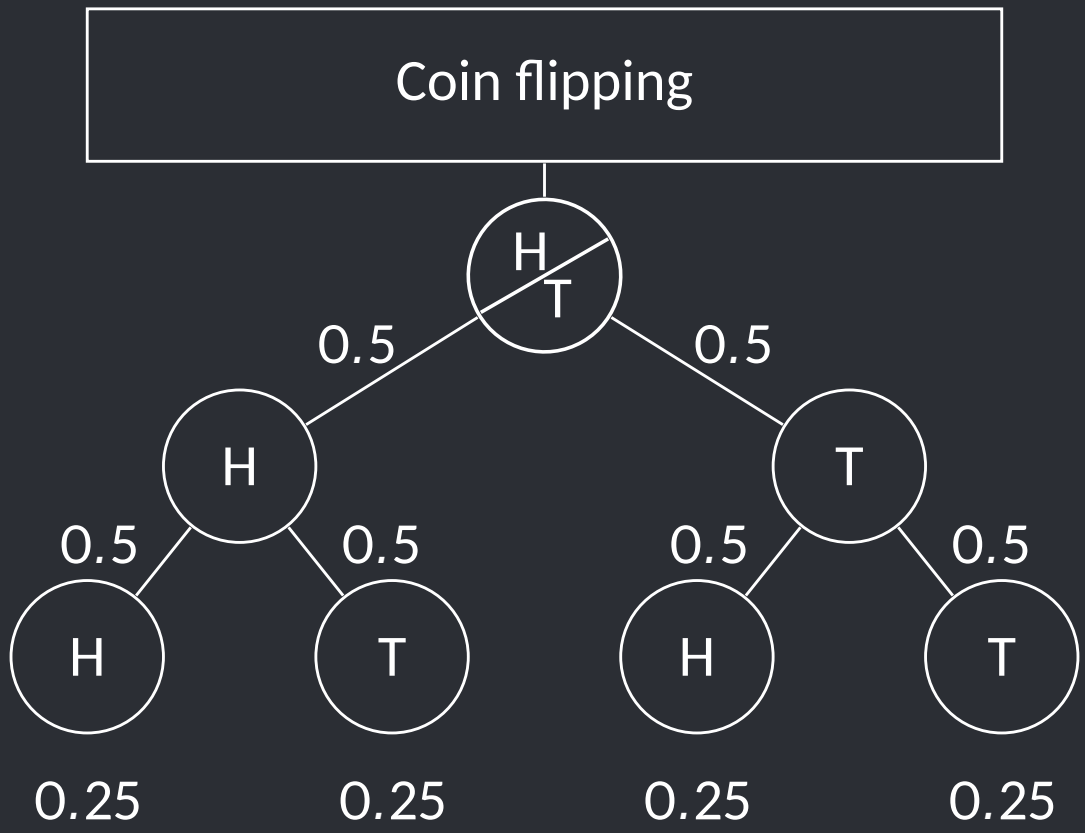


Figure – Tree of probabilities – Flipping a coin [3]

3. A derivative of a diagram from `texample.net` by cis, CC BY 2.5 licensed

Code listings

An example source code in C

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

// This is a comment
int main(int argc, char **argv)
{
        while (--c > 1 && !fork());
        sleep(c = atoi(v[c]));
        printf("%d\n", c);
        wait(0);
        return 0;
}
```

Citations

TEX, LATEX, and Beamer

TEX is a programming language for the typesetting of documents. It was created by Donald Erwin Knuth in the late 1970s and it is documented in *The TEXbook* [1].
In the early 1980s, Leslie Lamport created the initial version of LATEX, a high-level language on top of TEX, which is documented in *LATEX : A Document Preparation System* [2]. There exists a healthy ecosystem of packages that extend the base functionality of LATEX ; *The LATEX Companion* [3] acts as a guide through the ecosystem.
In 2003, Till Tantau created the initial version of Beamer, a LATEX package for the creation of presentations. Beamer is documented in the *User's Guide to the Beamer Class* [4].

Bibliography

TEX, LATEX, and Beamer

[1] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984.

[2] Leslie Lamport. *LATEX : A Document Preparation System*. Addison-Wesley, 1986.

[3] M. Goossens, F. Mittelbach, and A. Samarin. *The LATEX Companion*. Addison-Wesley, 1994.

[4] Till Tantau. *User's Guide to the Beamer Class Version 3.01*. Available at `http://latex-beamer.sourceforge.net`.

[5] A. Mertz and W. Slough. Edited by B. Beeton and K. Berry. *Beamer by example* In TUGboat, Vol. 26, No. 1., pp. 68-73.