Format unraveled

Richard Bonichon[1] & Pierre Weis[2]

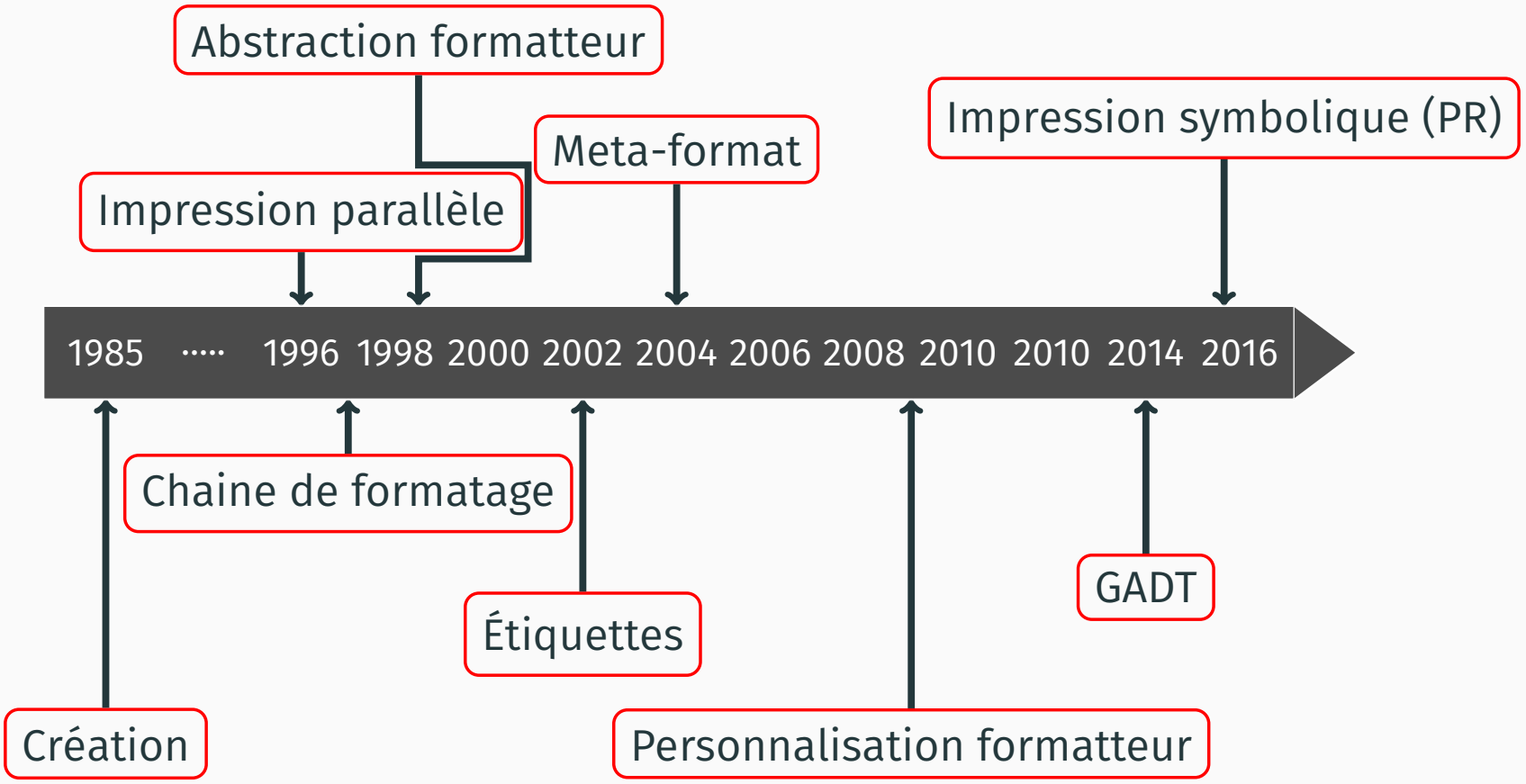
20170105

[1]CEA LIST

[2]INRIA Paris


Format
?

Un peu d'histoire



Indices de coupure

Boîtes

Indentation

Indice de coupure

Un indice de coupure marque un endroit où l'imprimeur peut couper le texte

```
print_break nspaces offset
```

```
print_space ()    =    print_break 1 0
print_cut ()      =    print_break 0 0
```

Boîtes

boîte = région
+ discipline cohérente
de passage à la ligne
et d'indentation

Les types de boîtes

Abbréviation	Type
h	horizontale
v	verticale
hv	horizontale ou bien verticale
hov	horizontale/verticale
b	basique

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte h

```
1-2-3-4-5-6
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte v

```
1
2
3
4
5
6
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte hv (v)

```
1
2
3
4
5
6
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte hv (h)

```
set_margin 12;;

1-2-3-4-5-6
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte hov

```
1-2-3-4
5-6
```

Rendu

```
(** Espaces rendues par '-' *)
let l = [ 1; 2; 3; 4; 5; 6; ] ;;
set_margin 8 ;;
```

Boîte b

```
1-2-3-4
5-6
```

Différencier les boîtes b & hov

Boîte b

```
[1-2-3
-4-5-6
]
```

Boîte hov

```
[1-2-3
-4-5-6]
```

```
set_margin 8;;
open_box 0;;          (* ou open_hovbox 0 *)
print_string "[";;
pp_format_hov l ;; (* imprime l dans boîte hov *)
print_cut () ;;     (* oh! la coupure! *)
print_string "]";;
close_box ();;
print_newline ();;
```

Impression élégante

$\neq$

Mise en page

Notation en chaîne de format

Fonctions	Chaîne
`open_hbox`	"@[<h>"
`open_vbox n`	"@[<v n>"
`open_hvbox n`	"@[<hv n>"
`open_hovbox n`	"@[<hov n>"
`open_box n`	"@[<n>"
`close_box`	"@]"
`print_break n o`	"@;<n o>"
`print_cut`	"@,"
`print_space`	"@ "

Triptyque

```
fprintf ppf "%a"
```

FPRINTF

```
Format.fprintf
:
Format.formatter
->
('a, Format.formatter, unit) format
->
'a
```

Abstraction du moteur d'impression

ppf

Construction & personnalisation

- `formatter_of_out_channel`
- `formatter_of_buffer`
- `make_formatter`
- `(pp_)set_formatter_out_functions`

%a

```
type expression =
  | Int of int
  | Add of expression * expression

let rec pp_expr ppf = function
  | Int n -> fprintf ppf "%i" n
  | Add (e1, e2) ->
    fprintf ppf "(%a@ +@ %a)"
      pp_expression e1 pp_expression e2

and pp_expression ppf =
  fprintf ppf "@[%a@]" pp_expr
```

Mécanisme avancé : les étiquettes

```
fprintf std_formatter "@{<error>Red is dead@}"
```

Défaut
Red is dead

Marque les étiquettes
Red is dead i.e. `"\027[0;31mRed is dead"`

Sortie Org

```
"@[<v 0>\
 @{<p>This paragraph precedes a list:@}@ \
 @{<ul>\
   @{<li>This@ first@ item@ might@ be@ too long@}\
   @{<li>Second item@}\
 @}\
 @]@.\
"
```

Org

```
This paragraph precedes a list:
- This first item might be
  too long
- Second item
```

Vers org

```
let org_tag_functions ppf =
  let mark_open_tag _ = "" and mark_close_tag _ = ""

  and print_open_tag  = function
    | "ul" -> fprintf ppf "@[<v>"
    | "li" -> fprintf ppf "- @[<hov>"
    | _    -> ()
  and print_close_tag  = function
    | "ul" -> fprintf ppf "@]"
    | "li" -> fprintf ppf "@]@ "
    | _    -> ()

  in { mark_open_tag;  mark_close_tag;
       print_open_tag; print_close_tag; }
```

Sortie HTML

```
"@[<v 0>\
 @{<p>This paragraph precedes a list:@}@ \
 @{<ul>\
   @{<li>This@ first@ item@ might@ be@ too long@}\
   @{<li>Second item@}\
 @}\
 @]@.\
"
```

HTML

```
<p>This paragraph precedes a list:</p>
<ul>
  <li>This first item might be
  too long</li>
  <li>Second item</li>
</ul>
```

Vers HTML

```
let html_tag_functions ppf =
  let mark_open_tag s =
    if s <> "ul" then "<" ^ s ^ ">" else ""
  and mark_close_tag s =
    if s <> "ul" then "</" ^ s ^ ">" else ""
  and print_open_tag  = function
    | "ul" -> fprintf ppf "@[<b>@<0><ul>@[<v 2>"
    | "li" -> fprintf ppf "@ @[<hov>"
    | "p"  -> fprintf ppf "@[<hov>"
    | _    -> ()
  and print_close_tag  = function
    | "ul" -> fprintf ppf "@]@,@<0></ul>@]"
    | "li"
    | "p"  -> fprintf ppf "@]"
    | _    -> ()
  in { mark_open_tag;  mark_close_tag;
       print_open_tag; print_close_tag; }
```

Emboîtez, coupez !

Emboîtez

Pas de boîte, pas de garantie ni de sémantique !

Aide-le et Format t'aidera

L'ajout d'indices de coupure et de boîte aide le formateur

Combinez, abstrayez

fprintf & %a

Utilisez `fprintf` et `%a` pour combiner vos fonctions d'impression.

Abstrayez

Ajoutez un formateur à vos fonctions d'impression pour les rendre génériques.

Attention ...


Don't Forget to Courtesy
FLUSH
Thank You!
SmartSign.com • 800-952-1457 • S2-0333

Bientôt ...

- Impression symbolique
- Impression tabulée
- Extension des chaînes de format