



CTUslides — jednoduché slidy ve stylu CTUstyle

Petr Olšák
petr@olsak.net

<http://petr.olsak.net/ctustyle.html>



Zahájení dokumentu

2

- Dokument vložíme do souboru (například `soubor.tex`) a zpracujeme příkazem `pdfcsplain soubor`.

- Do záhlaví dokumentu je třeba napsat:

```
\input ctuslides2 % načtení maker pro slídy (ve verzi 2)
\worktype[B/CZ]  % nastavení typu práce (B,M,D,O) a jazyka (CZ,SK,EN)
\faculty{F3}     % označení fakulty pro titulní stránku
\department {Katedra matematiky} % označení katedry

\slideshow      % zahájení dokumentu
... dokument ...
\pg.
```

- Dokument se ukončí sekvencí `\pg` následovanou tečkou.
- Pro bezproblémové zpracování je nutné mít OPmac ve verzi aspoň May 2015. Dostupné z <http://petr.olsak.net/opmac.html>.
- Nastavení typu práce a jazyka probíhá stejně jako v **CTUstyle**.
- Na rozdíl od **CTUstyle** je možné použít jen deklarační příkazy `\worktype`, `\faculty` a `\department`.



Základní struktura

3

- V celém dokumentu je možné psát * pro zahájení odrážky.
- Vložené odrážky (druhé a další úrovně) vzniknou v prostředí `\beginitems... \enditems`.
- Nadpisy slíd řešíme pomocí sekvence `\sec Nadpis` následované prázdným řádkem. Podobně lze použít `\secc Nadpis`.
- Pro titulní stranu (první slída) použijeme sekvenci `\tit Nadpis` následovanou prázdným řádkem.
- Po nadpisu pomocí `\tit` může následovat sekvence `\subtit Jméno autora apod.` rovněž následovaná prázdným řádkem.
- Texty v odstavci jsou zarovnány jen vlevo (na prapor).
- Pokud chceme odřádkovat, je možné použít sekvenci `\nl`.
- Odstránkování a konec dokumentu provedeme pomocí sekvence `\pg` následované znaky `+` nebo `,` nebo `.`
- Pásek s naznačenými stránkami vpravo je klikací a upraví se správně po druhém průchodu zpracování T_EXem.



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),
 - znakem `;`, pak se jedná o „normální“ odstránkování,



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),
 - znakem `;`, pak se jedná o „normální“ odstránkování,
 - znakem `.` což se *musí* použít na konci dokumentu.



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),
 - znakem `;`, pak se jedná o „normální“ odstránkování,
 - znakem `.` což se *musí* použít na konci dokumentu.

Shrnutí:

```
\pg+    ... pokračuj od stejného místa  
\pg;    ... nová strana  
\pg.    ... konec dokumentu
```



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),
 - znakem `;`, pak se jedná o „normální“ odstránkování,
 - znakem `.` což se *musí* použít na konci dokumentu.
- Shrnutí:
 - `\pg+` ... pokračuj od stejného místa
 - `\pg;` ... nová strana
 - `\pg.` ... konec dokumentu
- Jakmile odstraníme nebo zakomentujeme `\slideshow` ze záhlaví dokumentu, příkazy `\pg+` se deaktivují. To je vhodné pro verzi dokumentu pro tisk.



Způsoby odstránkování

- Na odstránkování se použije sekvence `\pg` následovaná:
 - znakem `+`, pak po odstránkování stávající text zůstává a přidává se k němu nový (postupné odhalování myšlenek),
 - znakem `;`, pak se jedná o „normální“ odstránkování,
 - znakem `.` což se *musí* použít na konci dokumentu.
- Shrnutí:
 - `\pg+` ... pokračuj od stejného místa
 - `\pg;` ... nová strana
 - `\pg.` ... konec dokumentu
- Jakmile odstraníme nebo zakomentujeme `\slideshow` ze záhlaví dokumentu, příkazy `\pg+` se deaktivují. To je vhodné pro verzi dokumentu pro tisk.
- Další zde nezmíněnou variantou je sekvence `\pg=`, která nezpůsobí odstránkování, ale používá se pro verbatim texty (viz dále).



Verbatim (tedy doslovné) texty

Verbatim texty v odstavci

- V textu odstavce *nelze* používat "... " pro verbatim úseky textu.
- Místo toho použijeme sekvenci `\code{...}` popsanou v OPmac triku 0102 na <http://petr.olsak.net/opmac-tricks.html#code>.
- Argument příkazu `\code{...}` se vypíše doslova, ale před problémové znaky je třeba psát backslash. Takže znak backslash se vytiskne jen tehdy, pokud je zdvojený.



Verbatim (tedy doslovné) texty

Verbatim texty v odstavci

- V textu odstavce *nelze* používat "... " pro verbatim úseky textu.
- Místo toho použijeme sekvenci `\code{...}` popsanou v OPmac triku 0102 na <http://petr.olsak.net/opmac-tricks.html#code>.
- Argument příkazu `\code{...}` se vypíše doslova, ale před problémové znaky je třeba psát backslash. Takže znak backslash se vytiskne jen tehdy, pokud je zdvojený.

Víceřádkové verbatim texty

- Pro výpisy víceřádkových kódů je nutné před `\begtt` použít `\pg=` takto:

```
\pg=\begtt  
... livovolný verbatim text ...  
\endtt
```

Následuje příklad...



Příklad výpisu víceřádkového kódu

Do zdrojového dokumentu napíšeme:

```
\pg=\typosize[13/15]\Red\begtt
#include <stdio.h>
int main();
{
    printf("Hello world!\n");
}
\endtt
```



Příklad výpisu víceřádkového kódu

Do zdrojového dokumentu napíšeme:

```
\pg=\typosize[13/15]\Red\begtt
#include <stdio.h>
int main();
{
    printf("Hello world!\n");
}
\endtt
```

A na výstupu dostaneme:

```
#include <stdio.h>
int main();
{
    printf("Hello world!\n");
}
```

Vidíme, že mezi `\pg=` a `\begtt` je možné vložit lokální nastavení sazby.



Menší potíže se sekvencí `\pg+`

- Sekvenci `\pg+` nelze použít uvnitř skupiny.
- Výjimkou je skupina vnořeného prostředí `\begitems... \enditems`.



Menší potíže se sekvencí `\pg+`

- Sekvenci `\pg+` nelze použít uvnitř skupiny.
- Výjimkou je skupina vnořeného prostředí `\begitems... \enditems`.

Jak se s tím vyrovnat?

- Přejít na jinou velikost fontu pomocí `\typosize` nebo `\typoscale` provedeme globálně, pak můžeme v této nové velikosti použít `\pg+` a pak se vrátíme k původní velikosti pomocí sekvence `\normalsize`.
- Chceme-li postupně poodhalovat jednotlivé řádky kódu, je možné použít:

```
\pg=\begtt  
... první řádek kódu ...  
\endtt \pg+ \pg=\begtt  
... druhý řádek kódu ...  
\endtt \pg+
```

- Pro odhalování „na přeskáčku“ a odhalování uvnitř skupin je možné použít makra `\use` a `\pshow...`



Odhalování pomocí `\use a \pshow`

- Makro `\use{podmínka}\povel` použije `\povel`, jen pokud číslo postupně odhalené slídy splňuje podmínku.
- Makro `\pshow num` (partially show) zobrazí následující text až po konec skupiny
 - neviditelně, je-li číslo odhalené slídy menší než `num`,
 - červeně, je-li číslo odhalené slídy rovno `num`,
 - černě, je-li číslo odhalené slídy větší.
- Číslo odhalené slídy se po každém `\pg`; resetuje na jedničku a po každém `\pg+` se zvětšuje o jedničku.
- Makro `\pshow` využívá `\use a` je definováno takto

```
\def\pshow#1{\use{=#1}\Red \use{<#1}\White \ignorespaces}
```




Příklad použití \pshow

\secc Myšlenky na přeskáčku

- * {\pshow1 První myšlenka}
- * {\pshow3 Druhá myšlenka}
- * {\pshow2 Třetí myšlenka}

\pg+\pg+\pg+

\secc Vzorec

Zabývejme se vzorcem

\$\$

$$E = {\pshow5 m}{\pshow6 c^2}$$

\$\$

\pg+\pg+\pg+

A to je vše.

\pg;

Myšlenky na přeskáčku

■ První myšlenka

■

■



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

■ První myšlenka



■ Třetí myšlenka



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

■ První myšlenka

■ Druhá myšlenka

■ Třetí myšlenka



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

- První myšlenka
- Druhá myšlenka
- Třetí myšlenka

Vzorec

Zabývejme se vzorcem

$E =$



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

- První myšlenka
- Druhá myšlenka
- Třetí myšlenka

Vzorec

Zabývejme se vzorcem

$$E = m$$



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

- První myšlenka
- Druhá myšlenka
- Třetí myšlenka

Vzorec

Zabývejme se vzorcem

$$E = mc^2$$



Příklad použití `\pshow`

`\secc` Myšlenky na přeskáčku

```
* {\pshow1 První myšlenka}  
* {\pshow3 Druhá myšlenka}  
* {\pshow2 Třetí myšlenka}
```

`\pg+\pg+\pg+`

`\secc` Vzorec

Zabývejme se vzorcem

```
$$  
E = {\pshow5 m}{\pshow6 c^2}  
$$
```

`\pg+\pg+\pg+`

A to je vše.

`\pg;`

Myšlenky na přeskáčku

- První myšlenka
- Druhá myšlenka
- Třetí myšlenka

Vzorec

Zabývejme se vzorcem

$$E = mc^2$$

A to je vše.



Tabulky, obrázky

- Tabulky lze udělat příkazem `\table` nebo `\ctable`.
- Obrázky lze vložit příkazem `\inpic`.
- Podrobněji viz dokumentaci k OPmac.
- Umístění na střed je možné zařídit pomocí `\centerline{}`.
- Příklad:



Tabulky, obrázky

- Tabulky lze udělat příkazem `\table` nebo `\ctable`.
- Obrázky lze vložit příkazem `\inpic`.
- Podrobněji viz dokumentaci k OPmac.
- Umístění na střed je možné zařídit pomocí `\centerline{}`.
- Příklad:

```
\centerline{\picw=5cm \inpic cmelak1.jpg }
```





Srovnání CTUslides a Beamer*

L^AT_EXový balíček Beamer umí mnohonásobně více věcí a nabízí množství předpřipravených typografických řešení, **ale**

- Beamer nutí (stejně jako L^AT_EX) dokument *programovat* za použití velkého množství nejrůznějších `\begin{něco}` a `\end{něco}` a dalších programátorských konstrukcí,
- zatímco plainT_EX umožňuje autorovi dokument *psát* s minimálním množstvím značek. Výsledný zdrojový kód je daleko přehlednější.

11+

* <http://www.ctan.org/pkg/beamer>



Srovnání CTUslides a Beamer*

L^AT_EXový balíček Beamer umí mnohonásobně více věcí a nabízí množství předpřipravených typografických řešení, **ale**

- Beamer nutí (stejně jako L^AT_EX) dokument *programovat* za použití velkého množství nejrůznějších `\begin{něco}` a `\end{něco}` a dalších programátorských konstrukcí,
- zatímco plainT_EX umožňuje autorovi dokument *psát* s minimálním množstvím značek. Výsledný zdrojový kód je daleko přehlednější.
- Beamer se naučíme používat po přečtení 250 stránkové dokumentace,
- zatímco v případě **CTUslides** stačí pročíst deset slíd**.

11+

* <http://www.ctan.org/pkg/beamer>

** tuto jedenáctou už nepočítáme



Srovnání CTUslides a Beameru*

L^AT_EXový balíček Beamer umí mnohonásobně více věcí a nabízí množství předpřipravených typografických řešení, **ale**

- Beamer nutí (stejně jako L^AT_EX) dokument *programovat* za použití velkého množství nejrůznějších `\begin{něco}` a `\end{něco}` a dalších programátorských konstrukcí,
- zatímco plainT_EX umožňuje autorovi dokument *psát* s minimálním množstvím značek. Výsledný zdrojový kód je daleko přehlednější.
- Beamer se naučíme používat po přečtení 250 stránkové dokumentace,
- zatímco v případě **CTUslides** stačí pročíst deset slíd**.
- Vzkaz pro programátory: naprogramovat další typografické řešení pro L^AT_EX je daleko komplikovanější, než implementovat typografický návrh v plainT_EXu. A abychom se v L^AT_EXu opravdu vyznali, stejně nejprve musíme pořádně ovládat T_EX.

* <http://www.ctan.org/pkg/beamer>

** tuto jedenáctou už nepočítáme



Děkuji za pozornost





Děkuji za pozornost

Dotazy?

