

# G-TeamPostScriptCreatorDeLuxe Referenční příručka

1

Generováno programem Doxygen 1.3.7

Tue May 25 13:38:04 2004



# Obsah



# Kapitola 1

## G-TeamPostScriptCreatorDeLuxe Rejstřík datových struktur

### 1.1 G-TeamPostScriptCreatorDeLuxe Datové struktury

Následující seznam obsahuje identifikace datových struktur a jejich stručné popisy:

<a href="#">afm_base_char</a> (Spojový seznam, který drží informace o základních (tj. s ASCII-kódem < 128) znacích) . . . . .	??
<a href="#">afm_composite</a> (Spojový seznam všech znaků, které jsou složené) . . . . .	??
<a href="#">afm_composite_char</a> (Spojový seznam jednotlivých znaků pro složení nějakého písmene) . . . . .	??
<a href="#">afm_font</a> (Struktura udržující kompletní informace o fontu) . . . . .	??
<a href="#">afm_glyph</a> (Struktura pro převod z post-scriptových názvů znaků na jejich hodnotu) . . . . .	??
<a href="#">afm_kerning</a> (Spojový seznam udržující informace o kreningu (v x-ové ose) ) . . . . .	??
<a href="#">cfg_config</a> (Struktura pro načtení konfiguračního souboru) . . . . .	??
<a href="#">enc_table</a> (Seznam pro převod z národního kódování na post-scriptové názvy znaků) . . . . .	??
<a href="#">err_queue</a> (Interní struktura, pro uchovávání posloupnosti chyb) . . . . .	??
<a href="#">ps_code_element</a> (Struktura popisující řádek zdrojového kódu) . . . . .	??
<a href="#">ps_headline_element</a> (Struktura popisující nadpis v textu) . . . . .	??
<a href="#">ps_image_element</a> (Struktura popisující vložený obrázek (ve formátu .eps)) . . . . .	??
<a href="#">ps_list_element</a> (Struktura popisující jednu odrážku v textu) . . . . .	??
<a href="#">ps_note_element</a> (Struktura popisující poznámku v textu) . . . . .	??
<a href="#">ps_paragraph</a> (Struktura popisující odstavec textu) . . . . .	??
<a href="#">ps_text_element</a> (Struktura popisující standardní text) . . . . .	??



## Kapitola 2

# G-TeamPostScriptCreatorDeLuxe

## Rejstøík souborù

### 2.1 G-TeamPostScriptCreatorDeLuxe Seznam souborù

Zde naleznete seznam všech dokumentovaných souborù se stručnými popisy:

<a href="#">afm.h</a> (Naèítání metrik fontù ) . . . . .	??
<a href="#">cfg.h</a> (Naèítání konfiguraèního souboru ) . . . . .	??
<a href="#">enc.h</a> (Naèítání kódovací tabulky ) . . . . .	??
<a href="#">err.h</a> (Obsluha výjimek ) . . . . .	??
<a href="#">ps.h</a> (Generování post-scriptu ) . . . . .	??
<a href="#">txt.h</a> (Naèítání vstupního textového souboru ) . . . . .	??
<a href="#">txt_def.h</a> . . . . .	??





## Kapitola 3

# G-TeamPostScriptCreatorDeLuxe

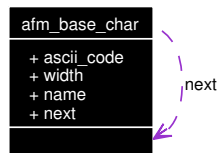
## Dokumentace datových struktur

### 3.1 Dokumentace struktury afm\_base\_char

spojový seznam, který drží informace o základních (tj. s ASCII-kódem < 128) znacích

```
#include <afm.h>
```

Diagram tříd pro afm\_base\_char:



#### Datové položky

- int `ascii_code`  
*ascii hodnota*
- float `width`  
*šířka znaku*
- char \* `name`  
*post-scriptové jméno*
- `afm_base_char` \* `next`  
*ukazatel na další položku v seznamu*

#### 3.1.1 Detailní popis

spojový seznam, který drží informace o základních (tj. s ASCII-kódem < 128) znacích

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

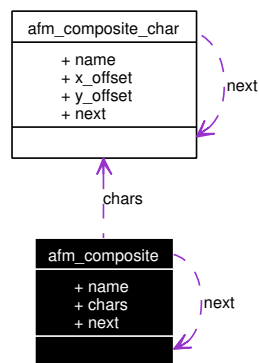
- [afm.h](#)

## 3.2 Dokumentace struktury afm\_composite

spojový seznam všech znaků, které jsou složené

```
#include <afm.h>
```

Diagram tříd pro afm\_composite:



### Datové položky

- `char * name`  
*jméno kompozitu (např. aacute)*
- `afm_composite_char * chars`  
*jednotlivé znaky kompozitu*
- `afm_composite * next`  
*další položka v seznamu*

### 3.2.1 Detailní popis

spojový seznam všech znaků, které jsou složené

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

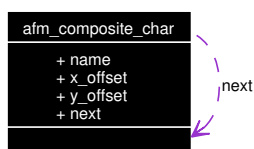
- `afm.h`

### 3.3 Dokumentace struktury `afm_composite_char`

spojový seznam jednotlivých znaků pro složení nějakého písmene

```
#include <afm.h>
```

Diagram třídy pro `afm_composite_char`:



#### Datové položky

- `char * name`  
*jméno znaku (např. o nebo acute)*
- `int x_offset`  
*x-posun*
- `int y_offset`  
*y-posun*
- `afm_composite_char * next`  
*další prvek v seznamu*

#### 3.3.1 Detailní popis

spojový seznam jednotlivých znaků pro složení nějakého písmene

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

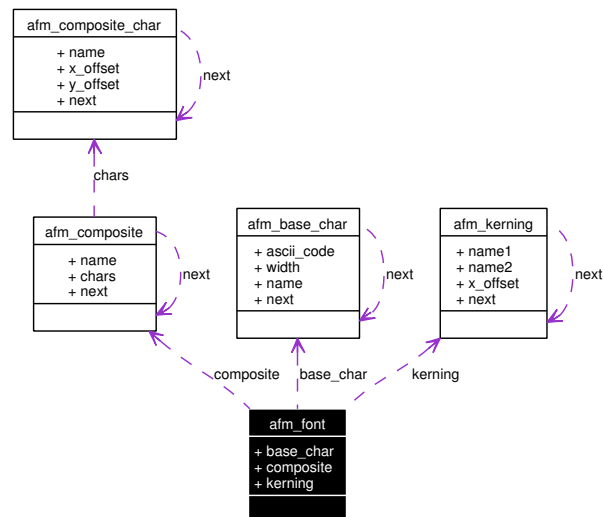
- [afm.h](#)

## 3.4 Dokumentace struktury afm\_font

struktura udržující kompletní informace o fontu

```
#include <afm.h>
```

Diagram tříd pro afm\_font:



### Datové položky

- [afm\\_base\\_char](#) \* [base\\_char](#)  
*seznam informací o základních znacích (tj. s ASCII-kódem < 128)*
- [afm\\_composite](#) \* [composite](#)  
*seznam kompozit*
- [afm\\_kerning](#) \* [kerning](#)  
*seznam kerningových dat*

### 3.4.1 Detailní popis

struktura udržující kompletní informace o fontu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

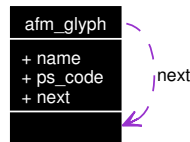
- [afm.h](#)

## 3.5 Dokumentace struktury afm\_glyph

struktura pro převod z post-scriptových názvů znaků na jejich hodnotu

```
#include <afm.h>
```

Diagram tříd pro afm\_glyph:



### Datové položky

- `char * name`  
*post-scriptové jméno znaku*
- `int ps_code`  
*post-scriptový kód znaku*
- `afm_glyph * next`  
*další položka v seznamu*

### 3.5.1 Detailní popis

struktura pro převod z post-scriptových názvů znaků na jejich hodnotu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

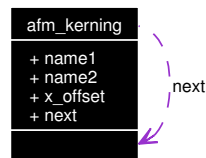
- `afm.h`

## 3.6 Dokumentace struktury afm\_kerning

spojový seznam udržující informace o kreningu (v x-ové ose)

```
#include <afm.h>
```

Diagram tříd pro afm\_kerning:



### Datové položky

- char \* [name1](#)  
*jméno prvního znaku*
- char \* [name2](#)  
*jméno druhého znaku*
- int [x\\_offset](#)  
*posun v x-ové ose*
- [afm\\_kerning](#) \* [next](#)  
*další položka seznamu*

### 3.6.1 Detailní popis

spojový seznam udržující informace o kreningu (v x-ové ose)

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [afm.h](#)

## 3.7 Dokumentace struktury `cfg_config`

Struktura pro načtení konfiguračního souboru.

```
#include <cfg.h>
```

### Datové položky

- `int border`  
*okraje papíru*
- `char * base_font`  
*jméno fontu základního textu*
- `float base_size`  
*velikost fontu základního textu*
- `float base_line`  
*řádkování základního textu*
- `char * head_font [3]`  
*jména fontu nadpisu*
- `float head_size [3]`  
*velikost fontu nadpisu*
- `float courier_size`  
*velikost courieru*
- `float courier_line`  
*řádkování courieru*

### 3.7.1 Detailní popis

Struktura pro načtení konfiguračního souboru.

Do této struktury se načte obsah konfiguračního souboru

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- `cfg.h`

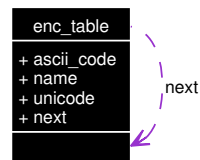


## 3.8 Dokumentace struktury enc\_table

seznam pro pøevod z národního kódování na post-scriptové názvy znakù

```
#include <enc.h>
```

Diagram tøíd pro enc\_table:



### Datové položky

- int [ascii\\_code](#)  
*ascii hodnota znakù*
- char \* [name](#)  
*post-scriptové jméno*
- int [unicode](#)  
*unicodova hodnta zanku*
- [enc\\_table](#) \* [next](#)  
*další položka v seznamu*

### 3.8.1 Detailní popis

seznam pro pøevod z národního kódování na post-scriptové názvy znakù

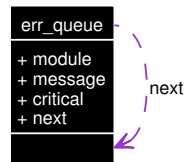
Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [enc.h](#)

## 3.9 Dokumentace struktury err\_queue

interní struktura, pro uchovávání posloupnosti chyb

Diagram třídy pro err\_queue:



### Datové položky

- `char * module`  
*jméno modulu, ve kterém chyba nastala*
- `char * message`  
*zpráva s popisem chyby*
- `int critical`  
*je chyba kritická?*
- `err_queue * next`  
*ukazatel na následující položku ve frontě*

### 3.9.1 Detailní popis

interní struktura, pro uchovávání posloupnosti chyb

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- `err.c`

## 3.10 Dokumentace struktury `ps_code_element`

struktura popisující řádek zdrojového kódu

```
#include <ps.h>
```

### Datové položky

- `char * text`  
*text kódu*

### 3.10.1 Detailní popis

struktura popisující řádek zdrojového kódu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [ps.h](#)

## 3.11 Dokumentace struktury `ps_headline_element`

struktura popisující nadpis v textu

```
#include <ps.h>
```

### Datové položky

- `int level`  
*uroveň nadpisu - přípustné hodnoty 1..3*
- `char * text`  
*text nadpisu*

### 3.11.1 Detailní popis

struktura popisující nadpis v textu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- `ps.h`

## 3.12 Dokumentace struktury `ps_image_element`

struktura popisující vložený obrázek (ve formátu .eps)

```
#include <ps.h>
```

### Datové položky

- `char * path`  
*cesta k obrázku*

### 3.12.1 Detailní popis

struktura popisující vložený obrázek (ve formátu .eps)

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [ps.h](#)

### 3.13 Dokumentace struktury `ps_list_element`

struktura popisující jednu odrážku v textu

```
#include <ps.h>
```

#### Datové položky

- `int level`  
*uroveň odrážky - přípustné hodnoty 1..3*
- `char * text`  
*text odrážky*

#### 3.13.1 Detailní popis

struktura popisující jednu odrážku v textu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- `ps.h`

## 3.14 Dokumentace struktury `ps_note_element`

struktura popisující poznamku v textu

```
#include <ps.h>
```

### Datové položky

- `char * text`  
*text poznámky*

### 3.14.1 Detailní popis

struktura popisující poznamku v textu

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

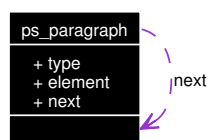
- [ps.h](#)

## 3.15 Dokumentace struktury ps\_paragraph

struktura popisující odstavec textu

```
#include <ps.h>
```

Diagram tříd pro ps\_paragraph:



### Datové položky

- enum [ps\\_paragraph\\_type](#) type  
*typ odstavce*
- void \* [element](#)  
*element(y)*
- [ps\\_paragraph](#) \* next  
*další odstavec*

### 3.15.1 Detailní popis

struktura popisující odstavec textu

odstavec je chápán jako řádka textu(!), která může přetéci na více řádek. takto je třeba dávat pozor na interpretaci odrážek, kdy se ve struktuře [ps\\_list\\_element](#) udržuje pouze text jedné(!) odrážky. obdobně se je třeba dívat na text kódu, kdy opit ve struktuře [ps\\_code\\_element](#) se udržují informace pouze o jednom řádku kódu.

- PT\_TEXT - odkazuje na strukturu [ps\\_text\\_element](#) (obyčejný text)
- PT\_HEADLINE - odkazuje na strukturu [ps\\_headline\\_element](#) (nadpis)
- PT\_LIST - odkazuje na strukturu [ps\\_list\\_element](#) (odrážka)
- PT\_NOTE - odkazuje na strukturu [ps\\_note\\_element](#) (poznámka)
- PT\_CODE - odkazuje na strukturu [ps\\_code\\_element](#) (řádek kódu)

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [ps.h](#)

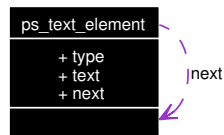


## 3.16 Dokumentace struktury ps\_text\_element

struktura popisující standardní text

```
#include <ps.h>
```

Diagram tříd pro ps\_text\_element:



### Datové položky

- enum [ps\\_text\\_type](#) type  
*typ textu*
- char \* [text](#)  
*znaky textu*
- [ps\\_text\\_element](#) \* next  
*další element textu*

### 3.16.1 Detailní popis

struktura popisující standardní text

Dokumentace pro tuto strukturu (struct) byla generována z následujícího souboru:

- [ps.h](#)



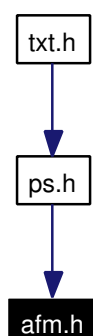
## Kapitola 4

# G-TeamPostScriptCreatorDeLuxe Dokumentace souborù

### 4.1 Dokumentace souboru afm.h

načítání metrik fontù

Následující graf ukazuje, které soubory přímo nebo nepřímo vkládají tento soubor:



#### Datové struktury

- struct [afm\\_base\\_char](#)  
*spojový seznam, který drží informace o základních (tj. s ASCII-kódem < 128) znacích*
- struct [afm\\_composite\\_char](#)  
*spojový seznam jednotlivých znaků pro složení nějakého písmene*
- struct [afm\\_composite](#)  
*spojový seznam všech znaků, které jsou složené*
- struct [afm\\_kerning](#)  
*spojový seznam udržující informace o kerningu (v x-ové ose)*

- struct [afm\\_font](#)  
*struktura udržující kompletní informace o fontu*
- struct [afm\\_glyph](#)  
*struktura pro převod z post-scriptových názvů znaků na jejich hodnotu*

## Definice typů

- typedef [afm\\_base\\_char](#) [afm\\_base\\_char](#)  
*spojový seznam, který drží informace o základních (tj. s ASCII-kódem < 128) znacích*
- typedef [afm\\_composite\\_char](#) [afm\\_composite\\_char](#)  
*spojový seznam jednotlivých znaků pro složení nějakého písmene*
- typedef [afm\\_composite](#) [afm\\_composite](#)  
*spojový seznam všech znaků, které jsou složené*
- typedef [afm\\_kerning](#) [afm\\_kerning](#)  
*spojový seznam udržující informace o kreningu (v x-ové ose)*
- typedef [afm\\_font](#) [afm\\_font](#)  
*struktura udržující kompletní informace o fontu*
- typedef [afm\\_glyph](#) [afm\\_glyph](#)  
*struktura pro převod z post-scriptových názvů znaků na jejich hodnotu*

## Funkce

- void [afm\\_load](#) (const char \*file\_name, [afm\\_font](#) \*\*font)  
*alokuje a nahraje informace o metrikách fontu*
- void [afm\\_del](#) ([afm\\_font](#) \*\*font)  
*dealokuje strukturu informací o fontu*
- void [afm\\_glyph\\_load](#) (const char \*file\_name, [afm\\_glyph](#) \*\*glyph)  
*alokuje a nahraje tabulku pro převod z post-scriptových názvů na post-scriptový kód*
- void [afm\\_glyph\\_del](#) ([afm\\_glyph](#) \*\*glyph)  
*dealokuje tabulku pro převod z post-scriptových názvů na post-scriptový kód*

### 4.1.1 Detailní popis

načítání metrik fontů

modul, který se stará o načítání informací o metrikách fontů

**Autor:**

pipa &amp; mrozu

**4.1.2 Dokumentace funkcí****4.1.2.1 void afm\_del (afm\_font \*\*font)**

dealokuje strukturu informací o fontu

**Parametry:***font* struktura určena ke zničení**4.1.2.2 void afm\_glyph\_del (afm\_glyph \*\*glyph)**

dealokuje tabulku pro převod z post-scriptových názvů na post-scriptový kód

**Parametry:***glyph* struktura ke zničení určená**4.1.2.3 void afm\_glyph\_load (const char \*file\_name, afm\_glyph \*\*glyph)**

alokuje a nahraje tabulku pro převod z post-scriptových názvů na post-scriptový kód

**Parametry:***file\_name* jméno vstupního souboru*glyph* výstupní struktura s informacemi pro převod**4.1.2.4 void afm\_load (const char \*file\_name, afm\_font \*\*font)**

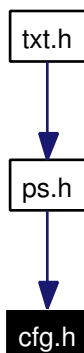
alokuje a nahraje informace o metrikách fontu

**Parametry:***file\_name* jméno vstupního souboru*font* výstupní struktura informací o fontu

## 4.2 Dokumentace souboru cfg.h

načítání konfiguračního souboru

Následující graf ukazuje, které soubory přímo nebo nepřímo vkládají tento soubor:



### Datové struktury

- struct [cfg\\_config](#)

*Struktura pro načtení konfiguračního souboru.*

### Definice typů

- typedef [cfg\\_config](#) [cfg\\_config](#)

*Struktura pro načtení konfiguračního souboru.*

### Funkce

- void [cfg\\_new](#) ([cfg\\_config](#) \*\*cfg)  
*vytvoří strukturu (alokace) a nahraje do ní defaultní nastavení*
- void [cfg\\_load](#) (const char \*file\_name, [cfg\\_config](#) \*cfg)  
*nahrání informací z konfiguračního souboru*
- void [cfg\\_del](#) ([cfg\\_config](#) \*\*cfg)  
*(v)zrušení struktury config*

#### 4.2.1 Detailní popis

načítání konfiguračního souboru

modul, který se stará o načítání konfigurace z textového souboru

**Autor:**

peshees

**4.2.2 Dokumentace definic typů****4.2.2.1 typedef struct [cfg\\_config](#) [cfg\\_config](#)**

Struktura pro naětení konfiguraěního souboru.

Do této struktury se naěte obsah konfiguraěního souboru

**4.2.3 Dokumentace funkcí****4.2.3.1 void [cfg\\_del](#) ([cfg\\_config](#) \*\* *cfg*)**

(v)zruěení struktury config

**Parametry:**

*cfg* konfiguraění struktura urěena ke zniěení

**4.2.3.2 void [cfg\\_load](#) (const char \**file\_name*, [cfg\\_config](#) \* *cfg*)**

nahrání informací z konfiguraěního souboru

nacte do struktury data z konfiguraěního souboru testuje správnost otevrení a uzavrení souboru

**4.2.3.3 void [cfg\\_new](#) ([cfg\\_config](#) \*\* *cfg*)**

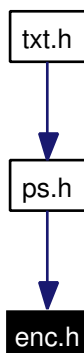
vytvoří strukturu (alokace) a nahraje do ní defaultní nastavení

vytvorí strukturu pro data z konfiguraěního souboru a uloží do ní defaultní nastavení testuje přidělování paměti

## 4.3 Dokumentace souboru enc.h

načítání kódovací tabulky

Následující graf ukazuje, které soubory přímo nebo nepřímo vkládají tento soubor:



### Datové struktury

- struct [enc\\_table](#)  
*seznam pro převod z národního kódování na post-scriptové názvy znaků*

### Definice typů

- typedef [enc\\_table](#) [enc\\_table](#)  
*seznam pro převod z národního kódování na post-scriptové názvy znaků*

### Funkce

- void [enc\\_load](#) (const char \*file\_name, [enc\\_table](#) \*\*table)  
*alokuje a nahraje tabulku pro převod národních znaků*
- void [enc\\_del](#) ([enc\\_table](#) \*\*table)  
*dealokuje tabulku pro převod národních znaků*

#### 4.3.1 Detailní popis

načítání kódovací tabulky

modul, který se stará o načítání kódovací tabulky, pro převod národních znaků

**Autor:**

pipa & mrozu



## 4.3.2 Dokumentace funkcí

### 4.3.2.1 void enc\_del ([enc\\_table](#) \*\* *table*)

dealokuje tabulku pro převod národních znaků

**Parametry:**

*table* struktura určená k zničení

### 4.3.2.2 void enc\_load (const char \**file\_name*, [enc\\_table](#) \*\* *table*)

alokuje a nahraje tabulku pro převod národních znaků

**Parametry:**

*file\_name* jméno vstupního souboru

*table* kódovací tabulka

## 4.4 Dokumentace souboru err.h

obsluha výjimek

### Funkce

- void `err_clear` ()  
*vyčistí všechny předchozí chyby v bufferu*
- void `err_add` (const char \*module, int critical, const char \*fmt,...)  
*přidá chybu do bufferu*
- void `err_print` ()  
*vypíše všechny chyby, které jsou v bufferu*
- int `err_critical` ()  
*test, zda nastala kritická chyba*

### 4.4.1 Detailní popis

obsluha výjimek

modul, který se stará o obsluhu výjimek v programu

#### Autor:

netnoise

### 4.4.2 Dokumentace funkcí

#### 4.4.2.1 void `err_add` (const char \* *module*, int *critical*, const char \* *fmt*, ...)

přidá chybu do bufferu

#### Parametry:

*module* jméno modulu, ve kterém vznikla chyba

*critical* zda chyba je kritická, pokud ano, znamená to, že se program ukončí

*fmt* formátovací řetězec (stejně jako u printf)

#### 4.4.2.2 int `err_critical` ()

test, zda nastala kritická chyba

#### Návratová hodnota:

1 pokud nastala kritická chyba, jinak 0

## 4.5 Dokumentace souboru ps.h

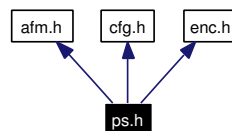
generování post-scriptu

```
#include "afm.h"
```

```
#include "cfg.h"
```

```
#include "enc.h"
```

Graf závislostí na vkládaných souborech pro ps.h:



Následující graf ukazuje, které soubory přímo nebo nepřímo vkládají tento soubor:



### Datové struktury

- struct [ps\\_text\\_element](#)  
*struktura popisující standardní text*
- struct [ps\\_headline\\_element](#)  
*struktura popisující nadpis v textu*
- struct [ps\\_list\\_element](#)  
*struktura popisující jednu odrážku v textu*
- struct [ps\\_note\\_element](#)  
*struktura popisující poznámku v textu*
- struct [ps\\_code\\_element](#)  
*struktura popisující řádek zdrojového kódu*
- struct [ps\\_image\\_element](#)  
*struktura popisující vložený obrázek (ve formátu .eps)*
- struct [ps\\_paragraph](#)  
*struktura popisující odstavec textu*

## Definice typù

- typedef [ps\\_text\\_element](#) [ps\\_text\\_element](#)  
*struktura popisující standardní text*
- typedef [ps\\_headline\\_element](#) [ps\\_headline\\_element](#)  
*struktura popisující nadpis v textu*
- typedef [ps\\_list\\_element](#) [ps\\_list\\_element](#)  
*struktura popisující jednu odrážku v textu*
- typedef [ps\\_note\\_element](#) [ps\\_note\\_element](#)  
*struktura popisující poznamku v textu*
- typedef [ps\\_code\\_element](#) [ps\\_code\\_element](#)  
*struktura popisující řádek zdrojového kódu*
- typedef [ps\\_image\\_element](#) [ps\\_image\\_element](#)  
*struktura popisující vložený obrázek (ve formátu .eps)*
- typedef [ps\\_paragraph](#) [ps\\_paragraph](#)  
*struktura popisující odstavec textu*

## Výèty

- enum [ps\\_text\\_type](#) { [TT\\_GENERIC](#), [TT\\_BOLD](#), [TT\\_ITALIC](#), [TT\\_COURIER](#) }  
*výèet typù textu, které se mohou v textu vyskytnout*
- enum [ps\\_paragraph\\_type](#) {  
[PT\\_TEXT](#), [PT\\_HEADLINE](#), [PT\\_LIST](#), [PT\\_NOTE](#),  
[PT\\_CODE](#), [PT\\_IMAGE](#) }  
*výèet typù odstavcù*

## Funkce

- void [ps\\_del](#) ([ps\\_paragraph](#) \*para)  
*zruší celou strukturu [ps\\_paragraph](#)*
- void [ps\\_save](#) (const char \*file\_name, [ps\\_paragraph](#) \*para, [cfg\\_config](#) \*cfg, [enc\\_table](#) \*table, [afm\\_font](#) \*fnt\_base, [afm\\_font](#) \*fnt\_bold, [afm\\_font](#) \*fnt\_italic, [afm\\_font](#) \*\*fnt\_headline, [afm\\_font](#) \*fnt\_code)  
*vygeneruje post-scriptový soubor*

## Proměnné

- enum [ps\\_text\\_type](#) `ps_text_type`  
*výčet typů textu, které se mohou v textu vyskytnout*
- enum [ps\\_paragraph\\_type](#) `ps_paragraph_type`  
*výčet typů odstavců*

### 4.5.1 Detailní popis

generování post-scriptu

modul, který se stará o generování post-scriptového souboru

**Autor:**

netnoise

### 4.5.2 Dokumentace definic typů

#### 4.5.2.1 typedef struct [ps\\_paragraph](#) `ps_paragraph`

struktura popisující odstavec textu

odstavec je chápán jako řádka textu(!), která může přetéci na více řádek. takto je třeba dávat pozor na interpretaci odrážek, kdy se ve struktuře [ps\\_list\\_element](#) udržuje pouze text jedné(!) odrážky. obdobně se je třeba dívat na text kódu, kdy opět ve struktuře [ps\\_code\\_element](#) se udržují informace pouze o jednom řádku kódu.

- `PT_TEXT` - odkazuje na strukturu [ps\\_text\\_element](#) (obvyčejný text)
- `PT_HEADLINE` - odkazuje na strukturu [ps\\_headline\\_element](#) (nadpis)
- `PT_LIST` - odkazuje na strukturu [ps\\_list\\_element](#) (odrážka)
- `PT_NOTE` - odkazuje na strukturu [ps\\_note\\_element](#) (poznámka)
- `PT_CODE` - odkazuje na strukturu [ps\\_code\\_element](#) (řádek kódu)

### 4.5.3 Dokumentace výřetových typů

#### 4.5.3.1 enum [ps\\_paragraph\\_type](#)

výčet typů odstavců

**Hodnoty výřetu:**

*`PT_TEXT`* obvyčejný nezvýrazněný text

*`PT_HEADLINE`* nadpis

*`PT_LIST`* odrážka (jedna)

*`PT_NOTE`* poznámka

*`PT_CODE`* řádek kódu

*`PT_IMAGE`* vložený obrázek

#### 4.5.3.2 enum `ps_text_type`

výčet typù textu, které se mohou v textu vyskytnout

Hodnoty výřtu:

*TT\_GENERIC* klasický nezvýrazněný text

*TT\_BOLD* tučný text

*TT\_ITALIC* kurzíva

*TT\_COURIER* neproporcionální písmo

#### 4.5.4 Dokumentace funkcí

##### 4.5.4.1 void `ps_del` (`ps_paragraph` \* *para*)

zruší celou strukturu `ps_paragraph`

Parametry:

*para* struktura určená ke zničení

##### 4.5.4.2 void `ps_save` (`const char` \* *file\_name*, `ps_paragraph` \* *para*, `cfg_config` \* *cfg*, `enc_table` \* *table*, `afm_font` \* *fnt\_base*, `afm_font` \* *fnt\_bold*, `afm_font` \* *fnt\_italic*, `afm_font` \* *fnt\_headline*, `afm_font` \* *fnt\_code*)

vygeneruje post-scriptový soubor

Parametry:

*file\_name* jméno souboru, který se bude generovat

*para* data dokumentu, která budou do souboru uložena

*cfg* struktura s informacemi o konfiguraci

*table* kódovací tabuka pro převod národních znakù

*fnt\_base* základní font, kterým se bude sázet text

*fnt\_bold* zvýrazněný font, kterým se bude sázet zvýrazněný text

*fnt\_italic* kurzivní font, kterým se bude sázet kurzivní text

*fnt\_headline* pole 3 fontù, které budou použity pro sazbu nadpisù

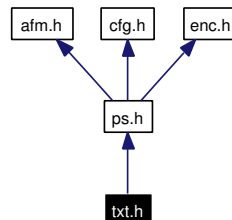
*fnt\_code* font, kterým se bude sázet neproporcionální text (tj. zdrojové kódy)

## 4.6 Dokumentace souboru txt.h

načítání vstupního textového souboru

```
#include "ps.h"
```

Graf závislostí na vkládaných souborech pro txt.h:



### Funkce

- void `txt_load` (const char \*file\_name, `ps_paragraph` \*\*para)  
*nahraje strukturu dokumentu z textového souboru*

#### 4.6.1 Detailní popis

načítání vstupního textového souboru

modul, který se stará o načítání dat ze vstupního textového souboru

**Autor:**

kolodij

#### 4.6.2 Dokumentace funkcí

##### 4.6.2.1 void `txt_load` (const char \*file\_name, `ps_paragraph` \*\* para)

nahraje strukturu dokumentu z textového souboru

**Parametry:**

*file\_name* jméno souboru

*para* vytvořený seznam odstavců