

# 目录

1	Hello ConT <sub>E</sub> Xt !	1
1	安裝 ConT <sub>E</sub> Xt	1
2	Hello ConT <sub>E</sub> Xt !	2
3	在編輯器中編輯文稿	2
4	獲取更多信息	3
2	文本編輯	5
1	添加字體	5
2	常見問題及解決方式	11
3	附錄 3: 字型、字體、字形	14
4	調整字體	17
5	段落和空白	23
A	Index	27



# Hello ConT<sub>E</sub>Xt!

# 1

歡迎進入 ConT<sub>E</sub>Xt 的世界! 在學習不同的程序語言的時候,第一次總是學習如何輸入和輸出 Hello World! 現在我們也開始試著如何在 ConT<sub>E</sub>Xt 中得到一份包含 Hello World! 的文件吧。

## 1 安裝 ConT<sub>E</sub>Xt

T<sub>E</sub>X 的世界包含了很多的分支: L<sup>A</sup>T<sub>E</sub>X、XeT<sub>E</sub>X、ConT<sub>E</sub>Xt 等等,安裝他們最方便的方式就是直接下載 Tex Live 軟件套件<sup>\*</sup>,<sup>\*\*</sup>。這樣可以最大限度的避免一些問題的發生。但是方便的同時,不可避免的就是會占據大量的存儲空間。ConT<sub>E</sub>Xt 作為從 T<sub>E</sub>X 中發展出來的項目,雖然也可以通過 Tex Live 進行安裝,但同時也有他自己的安裝方式。Wiki<sup>\*\*</sup>中比較詳細的描述了在不同系統環境下如何安裝。下面介紹如何在 MacOS 上進行安裝<sup>\*</sup>

1 安裝 ConT <sub>E</sub> Xt .....	1
2 Hello ConT <sub>E</sub> Xt! .....	2
3 在編輯器中編輯文稿 .....	2
4 獲取更多信息 .....	3

<sup>\*</sup> 該軟件套件的安裝方法可以參考 [tex:ctanmirror](#)<sup>✎</sup> 和 [tex:installation](#)<sup>✎</sup>。

<sup>\*\*</sup> 在 macOS 平臺上的軟件套件為 MacT<sub>E</sub>X。

<sup>\*\*</sup> 本文所說的 ConT<sub>E</sub>Xt Wiki 在沒有特殊說明的情況下都是指 ConT<sub>E</sub>Xt Garden 這個 Wiki 網站。

<sup>\*</sup> Wiki 同時提供了 ConT<sub>E</sub>Xt Mark IV 和 ConT<sub>E</sub>Xt LMTX 版本的安裝過程,更多內容可以參照 [ctxofficial:install\\_mkiv](#)<sup>✎</sup> 和 [ctxofficial:install\\_lmrx](#)<sup>✎</sup>。根據 Wiki 說明,ConT<sub>E</sub>Xt LMTX 中已經包含了 ConT<sub>E</sub>Xt Mark IV。按照本文的安裝步驟,即可安裝 ConT<sub>E</sub>Xt LMTX。

Download



图 1.1 為 macOS 下載 ConT<sub>E</sub>Xt



图 1.1 安裝 ConT<sub>E</sub>Xt 到用戶文件夾下

1. 透過 Wiki [ctxofficial:installation](#)<sup>✎</sup> 可以找到對應的下載鏈接。如果你的 MacBook 是 Apple Silicon 芯片的話,可以參照圖片 1.1 中的箭頭指向下載 ConT<sub>E</sub>Xt,如果是 Inter 芯片的話,可以選擇 X86 的鏈接進行下載。
2. 將下載好的壓縮包進行解壓(通過 Safari 下載的話,會自行解壓),把解壓好的文件夾重命名為 context。
3. 將 context 文件夾移動到你想要安裝的位置。比如放在用戶文件夾下。可以在終端<sup>\*\*</sup>依次執行如下的操作:
  - i. 在終端中輸入: `echo $HOME`(注意: HOME 務必請大寫),此時會得到一個路徑。
  - ii. 在終端中繼續輸入: `mv` (注意: `mv` 的後面有一個空格),然後拖動你下載好的 context 文件夾到 `mv` 的後面,然後再空一格,把剛纔通過 `echo $HOME` 得到的路徑複製到後面,回車後即可移動 context 文件夾到用戶文件夾下。具體的操作可以參照圖片 1.1。
4. 安裝 ConT<sub>E</sub>Xt。在終端中輸入 `cd $HOME/context` 就可以跳轉到 context 文件夾下。繼續在終端中輸入 `sh install.sh`,執行安裝程序。
5. 等待安裝結束。

<sup>\*\*</sup> 在 spotlight 中檢索 terminal,回車后即可啓動。

- 將 ConT<sub>E</sub>Xt 添加到系統環境。在終端中輸入下面的命令,回車即可。如果出現錯誤提示,可以試着將 `/.bashrc` 替換成 `/.zprofile`。

```
echo 'export PATH=$HOME/context/tex/texmf-linux-64/bin:$PATH'
>> /.bashrc
```

經過上述步驟之後,ConT<sub>E</sub>Xt 就已經安裝結束。可以通過 `context --version` 可以查看安裝的 ConT<sub>E</sub>Xt 版本。如果出現類似下方的信息就說明安裝成功了。

```
mtx-context | ConTEXt Process Management 1.05
mtx-context |
mtx-context | main context file: /texmf-context/tex/context/base/mkiv/context.mkiv
mtx-context | current version: 2023.09.26 18:19
mtx-context | main context file: /texmf-context/tex/context/base/mkxl/context.mkxl
mtx-context | current version: 2023.09.26 18:19
```

## 2 Hello ConT<sub>E</sub>Xt !

安裝結束後,我們就可以真正的開始進行我們的創作了。參考下面的操作步驟,來生成我們的 Hello World 吧。

\*\* 後綴就是指點號之後的內容。

- 新建一個文本文檔,並對他命名。例如: `ctx-abc.tex`。注意: 後綴務必是 `tex` \*\*。
- 在其中輸入如下內容: `\starttext Hello World! \stoptext`。
- 啓動終端。在終端中輸入 `context` (注意: `context` 后有一個空格。) 之後拖動你新建的 `ctx-abc.tex` 文件到終端中,即可得到該文件的具體位置。
- 回車即可。就可以看到在同一文件下多了一個名為 `ctx-abc.pdf` 的文件。打開它,完成我們的第一次 ConT<sub>E</sub>Xt 之旅吧。

SourceCode	Result
<code>\starttext Hello World! \stoptext</code>	Hello World!

## 3 在編輯器中編輯文稿

在上一節中我們已經排版出了我們自己的 Hello World (雖然不太算得上是排版,但至少我們踏出了第一步不是麼)。但是每次都需要打開終端,然後輸入對應的代碼什麼的,或許有些過於麻煩了。其實,在 T<sub>E</sub>X 的世界中,有很多的類似於 Word 一般的文本編輯器。不同的地方在於 Word 是屬於「所見即所得」的軟件,但是 T<sub>E</sub>X 世界中的編輯器基本上都不具有「所見即所得」(即時預覽)的功能。但我們自由的定製和比較全面的控制內容的輸出。

如果你安裝了 TeX Live 軟件套裝的話,會發現在 Launchpad (啓動臺) 中出現幾個隨附的軟件,其中就有一個名為 TeXShop 的軟件,之後我們就會使用這款軟件來編輯我們的文章。當然如果你

沒有使用 TeX Live 軟件套裝的話,也可以自行在 TeXShop 的網站 [tex:install\\_texshop](#) 中下載。在官網中選擇 Lastest Version 就可以了。不過,可能會因為網速問題,導致下載時間較長。除此之外,也可以下載其他的編輯器,在這個 [ctxofficial:text\\_editors](#) 列舉了多種可以使用的編輯器及其相應的功能。

下載並安裝之後,我們還需要做一些多餘的動作才能讓 TeXShop 自動為我們進行排版。TeXShop 默認的幾個排版引擎中並不包括 ConTeXt。需要我們自行添加。

按照 Wiki [ctx:ctxengine\\_to\\_texshop](#) 的步驟即可為 TeXShop 添加 ConTeXt 排版引擎的快捷啟動方式。下面是步驟速覽:

1. 在任意位置新建一個文本文件,命名為 **ConTeXt LMTX.engine** (注意: 文件後綴為 **engine**),並在其中填入下面的代碼。修改 **INSTALLATION\_PATH** 為你安裝 ConTeXt 的路徑。

```
#!/bin/bash
export PATH=INSTALLATION_PATH:$PATH
INSTALLATION_PATH/mtxrun --autogenerate --script
context --directives="system.showerror" --autopdf "$1" --pu
```

2. 打開終端,輸入 **mv** (注意: **mv** 后有一個空格。),然後拖動該文件 **ConTeXt LMTX.engine** 到終端中,之後在空一格,然後繼續輸入 **/Library/TeXShop/Engines/** 後,回車即可。
3. 打開 TeXShop, 打開 設置 (Preferences) → 排版引擎 (Typeset), 修改默認的排版引擎 **ConTeXt LMTX** 即可。之後就可以直接在 TeXShop 中通過點擊排版(**typeset**)直接進行排版了。可以參照圖片 1.2、1.3、1.4。

## 4 獲取更多信息

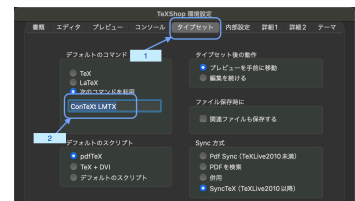
到此為止,我們已經基本上可以創作一篇屬於自己的英文文章了。但這些還遠沒有辦法達到我們的要求,比如: 如何插入章節? 如何對文字進行強調? 頁眉頁腳又該怎麼設置? 最重要的: 如何在 ConTeXt 中使用中文進行輸入等等這些在之後的章節中,我們會繼續進行介紹。每一章都以比較明顯易知的名字命名,可以通過目錄跳轉到自己感興趣的章節。另外在整個文章最後也提供了本文所使用的大部分命令的合集,並指明了 Wiki 的鏈接和命令合集中的頁碼。感興趣的讀者可以自行查找翻看。這一小節主要說明遇到問題後應該在何處提問或者查找答案。

1. *ConTeXt reference manual* : Hans Hagen, 2001/11/12, [ctxbooks:cont-enp](#) :

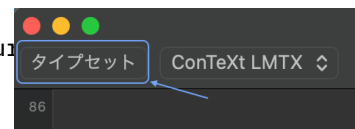
本書是 Hans Hagen 編寫的參考手冊。書中記錄了一部分 ConTeXt 的實現過程以及一些排版方面的內容。本書是作為參考手冊編寫的,其中含有大量的命令和示例。不過本書尚未完結,在 2013 年 Hans Hagen 和 Taco Hoekwater 對其進行了一



边图 1.2 打開設置文件



边图 1.3 設置排印引擎



边图 1.4 排印按鍵

\* 網址為：<http://pmrb.free.fr/conte-xtref.pdf>。似乎無法打開。

次更新。\* 不過本書部頭有點大（2001 版共計 370 餘頁，畢竟是作為參考手冊存在）。同時，本書擁有豐富的附錄，特別是命令附錄。由於命令過多，因此下條中單獨列出：命令合集。

2. 命令合集：Hans Hagen, 2023/6/4, [ctxbooks:setup-en](#)<sup>\*</sup>：作為 *ConT<sub>E</sub>Xt reference manual* 中附錄單獨釋出，最近一次更新在 2023/6/4，含有 ConT<sub>E</sub>Xt 中的系統命令、排版命令和作者命令。同時附有簡單的參數說明。作為命令查找備忘是不錯的選擇。
3. 漫步 ConT<sub>E</sub>Xt Mark IV：Ton Otten, PRAGMA ADE, 2017/10/5, [ctxbooks:ma-cb-en](#)<sup>\*</sup>：本文類似於 L<sup>A</sup>T<sub>E</sub>X 世界中的 A (Not So) *Short Introduction to L<sup>A</sup>T<sub>E</sub>X2*。文中對大部分的 ConT<sub>E</sub>Xt 命令進行了簡短的介紹，但也僅限於進行簡短的介紹。用來瞭解 ConT<sub>E</sub>Xt 是最合不過了。本書提供了多個語種的翻譯版本，不過中文版是沒有的。<sup>\*\*</sup>
4. ConT<sub>E</sub>Xt 蹊徑：李延瑞, 2023 年 3 月 31 日, [ctxbooks:context-notes](#)<sup>\*</sup>：這是一篇中文使用教程，文章中介紹了如何在 Windows 中安裝以及使用。主要章節包括：安裝、字體、文本格式化（文本樣式）、列表、參考文獻、公式、圖片、表格、抄錄環境（代碼）、MetaPost 等等。語言簡練，含有大量的示例。同時本文作者也是 zhfonts 模塊的作者。<sup>\*\*</sup>

其他一些可能用得到的網址或文件：

1. 隨附發行版的手冊。ConT<sub>E</sub>Xt 的發行版會隨附一些使用手冊和一些簡單的示例。他們安裝在 `INSTALLATION_PATH/context-osx-arm64/tex/texmf-context/doc` 目錄下。可以查看這些文件。他們全部以英文寫就。
2. Maillist。如果這些手冊和 Wiki 都無法解答你的問題，可以試試發郵件詢問世界範圍內的 ConT<sub>E</sub>Xt 使用者。大部分情況下，你都可以獲得不錯的解決方案。<sup>\*</sup>
3. stackexchange 問答社區。在這裡你可以檢索到很多有用的問答，就如同在知乎（未被大量無效問題汙染時的知乎）上一樣。不過，這個網站可能需要一些技巧才能夠在國內使用。當然，如果你有足夠耐心的話，可以慢慢等待他的加載。<sup>\*\*</sup>
4. 萬能的搜索引擎。例如：[bing](#), [Google](#)。

<sup>\*\*</sup> zhfonts 模塊的下載地址：[ctxmodules:zhfonts](#)<sup>\*</sup>。

<sup>\*</sup> 郵件地址是：[ctx:mailing\\_list](#)<sup>\*</sup>。你需要先註冊，才能夠提問。

<sup>\*\*</sup> 網址位於：[ctx:stackexchange](#)<sup>\*</sup>。

通過前面的內容,我們已經可以在 ConTeXt 中書寫文章了,只是我們只能夠在其中輸入英文。如果我們嘗試着在其中輸入中文的話,會發現在生成的 PDF 中沒有正確的顯示漢字。這是因為 ConTeXt 預定義的字體是只有英文字符,我們並沒有定義漢字應該用什麼字體顯示。ConTeXt 自然就不知道該如何顯示漢字了。因此,接下來我們就試着在 ConTeXt 中定義中文字體。

1 添加字體 .....	5
2 常見問題及解決方式 ....	11
3 附錄 3: 字型、字體、 字形 .....	14
4 調整字體 .....	17
5 段落和空白 .....	23

## 1 添加字體

### 1 在 macOS 系統為 ConTeXt 添加系統字體

雖然 ConTeXt 在為字體便捷添加上做了努力,但是對於中文字體的添加依然需要我們額外執行一些操作。下文主要以在 macOS 系統上為 ConTeXt 添加中文字體為主要敘述內容,Windows 系統中如果有不同的部分以別列或腳註的形式給出。

#### • 將系統字體文件路徑添加到環境變量中

#### Info

注意,在最新的版本中無須做過多的設置,此步可以略過。ConTeXt 已經在 mtxrund.lua 中添加了 Windows 和 macOS 的系統文件路徑。若發現無法找到字體時,在執行此步。

特別地,macOS 系統中有些字體文件並不是隨系統直接安裝的,而是可選安裝。這些字體文件並沒有安裝在系統字體路徑下。它們位於

`/System/Library/AssetsV2/com_apple_MobileAsset_Font7/`

文件夾下。而這個文件夾並沒有被 ConTeXt 默認寫入 mtxrund.lua,需要手動添加。華文或者華康的大部分字體都是位於這個文件夾下。這些字體需要執行此步才可以被 ConTeXt 索引。

在 macOS 系統中: 打開終端,並在其中輸入\*\*

```
open -a textedit .zprofile
```

這段命令將使用系統的文本編輯器打開配置文件。之後在打開的文本編輯器中繼續輸入下面的命令後,保存關閉文本編輯器即可。如此,可以將路徑\* 添加到系統環境中。ConTeXt 將從這幾個文件路徑中檢索字體。

\*\* 配置文件的名字會根據終端不同會有不同的名字,一般而言為 .zprofile。其他可能的名字: .bashrc。

\* `/Library/Fonts/, /System/Library/Fonts, $HOME/Library/Fonts, /System/Library/AssetsV2/com_apple_MobileAsset_Font7/`



\*\* 根據 [ctxofficial:morefont](#) 中描述的,在 ConT<sub>E</sub>Xt 2023 版本中無需這一步,但如果發現 ConT<sub>E</sub>Xt 無法找到 Windows 系統已經安裝的字體,可以試試這一步驟。

\*\* !!! 請注意,每當你安裝新字體或者更新 ConT<sub>E</sub>Xt 後都需要執行這個操作來重新建立索引。如果發現沒有找到需要的字體,可以多執行一次試試。

```
export OSFONTPATH=/Library/Fonts:/System/Library/Fonts:$HOME/Library/Fonts:/System/Library/AssetsV2/com_apple_MobileAssets_Font7/
```

在 Windows 系統中: 在開始菜單中找到終端,右擊終端后,在更多中點擊以管理員身份運行。在命令行界面輸入下面的命令後回車即可\*\*

```
setx OSFONTPATH c:/windows/fonts//m
```

### ● 刷新和為字體編制索引\*\*

同樣是在終端中輸入下方所示命令。之後就會看到一連串的代碼快速閃現。

```
mtxrun --generate & mtxrun --script fonts --reload
```

請注意,如果你需要使用 macOS 系統中的華文或者華康系列的字體,則需要執行下面的命令才可以正確的建立索引。同時,由於 ttc 字體似乎無法正確地索引,使用這個命令也可以解決此問題。

```
mtxrun --generate & mtxrun --script fonts --reload --force
```

系統將從上面的路徑中查找字體信息,並創建索引以便在日後快速調用。

### ● 挑選需要使用的字體

如果你沒有特別的字體要求,可以直接使用 ConT<sub>E</sub>Xt 中已經預設好的打字集。關於這些預設好的打字集可以直接跳到節 4 查看,下面的內容可以略過。

\* macOS 系統可以直接打開字體冊 (Font Book) 來查看。Windows 系統可以直接在 C:/windows/fonts/ 這個文件夾下查看,或者 Windows 11 系統也可以通過設置 → 個性化 → 字體進行查看。

\*\* 注意,只是大體相當。並非完全對應。

.....  
先在系統字體目錄\*下找出自己想要的字體。一般而言,至少需要一種字體來作為中文字體。大多數情況下,都是需要選擇二至四種字體。它們分別是:襯線體、非襯線體、等寬體、手寫體,等等。其中,襯線體和宋體相當,非襯線體和黑體相當,等寬體(打印機體)專用於顯示代碼,手寫體和楷體相當。\*當然,也可以直接選擇宋體、黑體、楷體、仿宋。例如:

表 2.1 字體選擇示例

族類	中文字型	英文字型
襯線體(rm)	思源宋體	Source Serif Pro
無襯線體(ss)	思源黑體	Source Sans Pro
打印機體(tt)	仿宋	Source Code Pro

### ● 查詢挑選字體的相關信息

選擇完自己想要的字體之後,就可以通過下方的命令\*\*來查詢字體的 fontfamily,以便於 ConT<sub>E</sub>Xt 能夠正確的查找到你想要使用的字

\*\* 更多的命令可以參考: [mtx-fonts.html](#)。  
該文件隨發行版發布,可以直接在文件瀏覽器中搜索查看。



體。

```
mtxrun --script fonts --list --all YourFontName
```

假設在我們的系統中安裝了名為 SourceHanSerifCN-Regular.otf 的字體文件，\* 我們可以使用這個字體名作為關鍵字進行搜索，也可以簡單地截取這個字體文件的某幾個字作為關鍵字檢索。例如：source。 \*\*

\* 即思源宋體，字重為標準。

\*\* 注意，作為檢索字體的關鍵字，其大小寫可能會影響字體的檢索，當使用小寫檢索不到的時候，試試用大寫，或者直接先用字體文件名檢索。

```
mtxrun --script font --list --file --all source
```

你會得到如下的字體信息（省略了一部分不重要的內容）：

familyname	weight	fontname	filename
sourcehanserif	semibold	sourcehanserifcnsemibold	Dir/SourceHanSerifCN-SemiBold.otf
sourcehanserif	normal	sourcehanserifcnregular	Dir/SourceHanSerifCN-Regular.otf
sourcehanserif	medium	sourcehanserifcnmedium	Dir/SourceHanSerifCN-Medium.otf
sourcehanserif	light	sourcehanserifcnlight	Dir/SourceHanSerifCN-Light.otf
sourcehanserif	heavy	sourcehanserifcnheavy	Dir/SourceHanSerifCN-Heavy.otf
sourcehanserif	extralight	sourcehanserifcnextralight	Dir/SourceHanSerifCN-ExtraLight.otf
sourcehanserif	bold	sourcehanserifcnbold	Dir/SourceHanSerifCN-Bold.otf

其中我們就可以看到我們需要的 SourceHanSerifCN-Regular.otf 字體的 fontfamily 是 sourcehanserif。同時，我們也可以看到在查找到的結果中除了顯示了 normal 的字重（weight），還有其他的中等（medium）、粗體（bold）、細體（light）等。收集其中的 fontname 信息，稍後會在編制字體組合的時候用到。

這樣我們就可以指定思源宋體（Source Han Serif）作為襯綫體。依照相同的方法我們可以得到思源黑體（Source Han Sans）、仿宋（fangsong）。

## • 編制合適的字體組合

找到字體信息后，我們就可以通過 `\definefontfamily` \*\* 命令 \*\* `cmd:definefontfamily` 來使用我們挑選的這些字體了。

```
\definefontfamily[source][rm][sourcehanserifcn][tf=file:SourceHanSerifCN-Regular.otf]
\definefontfamily[source][ss][sourcehansancn][bf=file:SourceHanSansCN-Regular.otf]
\definefontfamily[source][tt][sourcehanserifcn][it=file:SourceHanSerifCN-Bold.otf]

\definefontfamily[source][rm][sourcehanserifcn][tf=name:sourcehanserifcnregular]
\definefontfamily[source][ss][sourcehansancn][bf=name:sourcehansancnregular]
\definefontfamily[source][tt][sourcehanserifcn][it=name:sourcehanserifcnbold]

\definefontfamily[source][rm][sourcehanserif][tf={style:regular}]
\definefontfamily[source][ss][sourcehansans][bf={style:regular}]
\definefontfamily[source][tt][sourcehanserif][it={style:bold}]
```

SourceCode	Result
<pre> \definefontfamily[source][rm][sourcehanserifcn] [tf=style:regular,bf=style:bold] \definefontfamily[source][ss][sourcehansanscn] [tf=style:regular,bf=style:bold] \definefontfamily[source][tt][fangsong] [tf=name:fangsong] \setupbodyfont [source] \tfx% 縮小字號 \rm 這是襯線體。      字體信息為:\the\font。 \hfil\par \ss 這是非襯線體(粗體)。字體信息為:\the\font。 \hfil\par \tt 這是等寬體(斜體)。字體信息為:\the\font。 \hfil\par </pre>	<p>這是襯線體。字體信息為:&lt;106: SourceHanSerifCN-Regular @ 9.0pt&gt;。</p> <p>這是非襯線體(粗體)。字體信息為:&lt;107: SourceHanSansCN-Regular @ 9.0pt&gt;。</p> <p>這是等寬體(斜體)。字體信息為:&lt;108: simfang @ 9.0pt&gt;。</p>

例如上述的命令就定義了一個名為 `source` 的字體組合,並通過命令 `\setupbodyfont[source]` 來啟用。\*,\*\* 上述命令的具體用法如下:

\* `cmd:definefontfamily`  
 \*\* `cmd:setupbodyfont`

```

\definefontfamily [_OPT_ARGI_] [_OPT_ARGII_] [_OPT_ARGIII_]
[_OPT_ARGIV_]
_OPT_ARGI_ : 指定字體組合的名稱,將通過 \setupbodyfont 啟用。
_OPT_ARGII_ : 指定字體族類。可以指定為:
    rm ss tt mm hw cg
    roman serif regular sansserif sans
    support teletype type mono
    handwritten calligraphic
    math mathematics
_OPT_ARGIII_ : 指定字體。即通過 pattern 檢索到的相關信息。
    指定方式可以是
    FONT FAMILY: [sourcehansanscn] 或者是
    FONTNAME: [name:sourcehanserifcnregular] 或者是
    FILENAME: [file:sourcehansansCN-Regular.otf]
_OPT_ARGIV_ : 指定字體的不同樣式。指定方式為
    stylename = file:FILENAME
    可選的 stylename 參數包括: tf it sl bf bi sc
    或者是指定字體特性 \paren{features}。
    指定方式為 features = FEATURENAME
    或者是指定字體縮放 \paren{rscale}。
    指定方式為 rscale = NUMBER
    不同的指定之間用逗號 \paren{,} 分割

```

### • 為英文設定字體

如果沒有特殊的英文字體要求,可以不執行此步。中文字體內部已經包含了英文字符。只是相對而言並不如英文字體那麼美觀。ConTeXt 為我們提供了字體回退機制: 即當系統中出現了指定的字體中沒有的字符,就可以通過回退字體為字符提供顯示。否則在最終生成的文件中將顯示空白。可以試著不指定中文字體,而直接在 ConTeXt 中輸入中文,猜猜看這時會發生什麼。

如果你需要為英文指定單獨的字體,可以使用 `\definefallbackfamily` 來為不同的族類指定英文字體。該命令基本上和 `definefontfamily` 具有相同的用法,區別在於該命令指定了

回退範圍。該命令的一些用法如下\*\*：

\*\* `cmd:definefallbackfamily`

```
\definefallbackfamily [_OPT_ARGI_][_OPT_ARGII_][_OPT_ARGIII_]
[_OPT_ARGIV_]
_OPT_ARGI_ : 指定字體組合的名稱,和需要指定回退字體的字體組合
名稱相同
_OPT_ARGII_ : 指定字體族類。可以指定為:
    rm ss tt mm hw cg
    roman serif regular sansserif sans
    support teletype type mono
    handwritten calligraphic
    math mathematics
_OPT_ARGIII_ : 指定字體。即通過 pattern 檢索到的相關信息。
指定方式可以是
FONT FAMILY: [sourcehansanscn] 或者是
FONTNAME: [name:sourcehanserifcnregular] 或者是
FILENAME: [file:sourcehansansCN-Regular.otf]
_OPT_ARGIV_ : 指定字體的不同樣式。指定方式為
    stylename = file:FILENAME
可選的 stylename 參數包括: tf it sl bf bi sc
或者是指定字體特性 \paren{features}。指定方式為
features = FEATURENAME
或者是指定字體縮放 \paren{rscale}。指定方式為 rscale
= NUMBER
或者是指定字符範圍 \paren{range},也就是需要回退的字符
範圍。
    在官方 Wiki 上詳細說明了不同的字符範圍。
    指定方式為 range = {_RANGE_NAME_}
    或者是否指定強制回退 \paren{force}。指定方式為
force = yes/force = no
不同的指定之間用逗號 \paren{,}分割
```

在 Wiki [ctxofficial:unicodeblocks](#) 上描述了字符範圍,並為不同的範圍設定了範圍別名。例如,想要回退的範圍是英文字母、部分常見英文標點、部分擴展性質的字母時,就可以按照如下方式進行定義。其中 `basiclatin` 的範圍是常見字母和標點,`latinsupplement` 的範圍則是擴展性的字母(帶有重音等符號的字母)。

```
\definefallbackfamily [source][rm][libertinusserif]
[range={basiclatin,latinsupplement},force=yes]
```

### • 啓用編制好的字體組合

在定義好我們的字體組合後,ConT<sub>E</sub>Xt 並不會立即啓用我們設置好的字體。通過命令 `\setupbodyfont` \* 就可以啓用了。具體用法如下:

\* `cmd:setupbodyfont`

```
\setupbodyfont [_OPT_ARG_]
[_OPT_ARG_]: typescript name
    serif roman
    sans sansserif
    mono type teletype
```

handwritten calligraphic  
5pt ... 12pt  
不同的指定之間用逗號 \paren{,} 分割

例如: \setupbodyfont[source,rm,8pt] 定義了正文字體使用 source 字體組合,主要使用襯線體,並且正文字號為 8pt。

SourceCode	Result
<pre> \definefontfamily[source][rm][sourcehansercfn] [tf=file:SourceHansSerifCN-Regular.otf, it={style:bold}] \definefontfamily[source][ss][sourcehansercfn] [tf=file:SourceHansSerifCN-Regular.otf, it={style:bold}] \definefontfamily[source][tt][sourcehansercfn] [tf=file:SourceHansSerifCN-Bold.otf, it={style:regular}] \setupbodyfont [source, rm] \tfxx% 縮小字號為 xx 尺寸。 \rm 這裏用思源宋體定義為本行字體,同時使用{\it 思源宋 體 Bold 字重}作為斜體。 \crlf Here we use SourceHansSerif as the font of this line, and use SourceHansSerif Heavy font weight as italics. \fancybreak\tfxx \ss 這裏用思源黑體定義為本行字體,同時使用{\it 思源宋 體 Bold 字重}作為斜體。 \crlf Here we use SourceHansSans bold font to define the font of this line, and at the same time, we use SourceHansSerif Heavy font weight as italics. \fancybreak\tfxx \tt 這裏用思源宋體定義為本行字體,同時使用{\it 思源宋 體 Regular 字重}作為斜體。 \crlf Here, SourceMono font is defined as the font of this line, and SourceHansSerif Heavy is used as italics. </pre>	<p>這裏用思源宋體定義為本行字體,同時使用思源宋體 Bold 字重作為斜體。 Here we use SourceHansSerif as the font of this line, and use SourceHansSerif Heavy font weight as italics. 這裏用思源黑體定義為本行字體,同時使用思源宋體 Bold 字重作為斜體。 Here we use SourceHansSans bold font to define the font of this line, and at the same time, we use SourceHansSerif Heavy font weight as italics. 這裏用思源宋體定義為本行字體,同時使用思源宋體 Regular 字重作為斜體。 Here, SourceMono font is defined as the font of this line, and SourceHansSerif Heavy is used as italics.</p>

## 2 創建自定義的打字集

## 3 其他的字體添加方式

除了上述的定義字體的方式之外,還有以下幾種方式來定義字體以及為此字體指定特殊的回退字體。區別於 \definefontfamily 和 \definefallbackfamily 定義的是字體族類,而 \definefont\*\* 和 \definefontfallback\*\* 定義的是某個具體的字體及其對應的回退字體。

```

** cmd:definefont
** cmd:definefontfallback

```

```

\definefont [_OPT_ARGI][_OPT_ARGII][_OPT_ARGIII]
_OPT_ARGI: 指定字體名稱,之後可以使用命令 _OPT_ARGI 啟用該
字體。
_OPT_ARGII: 指定字體。指定方式可以是 FONT FAMILY,
            或者是 FONTNAME,
            或者是 FILENAME
_OPT_ARGIII: 屬性繼承自 / BTEX \ctxcmdinfo{setupinterlinespace}/ETEX

\definefontfallback [_OPT_ARGI][_OPT_ARGII][_OPT_ARGIII]
[_OPT_ARGIV]
_OPT_ARGI: 指定字體組合的名稱,和需要指定回退字體的字體組合
名稱相同

```

`_OPT_ARGII`：指定字體。指定方式可以是 FONT FAMILY，  
或者是 FONTNAME，  
或者是 FILENAME

`_OPT_ARGIII`：指定字符範圍，也就是需要回退的字符範圍。  
在官方 Wiki 上詳細說明了不同的字符範圍。

`_OPT_ARGIV`：指定字體縮放 `\paren{rscale}`。指定方式為 `rscale = NUMBER`  
或者是是否指定強制回退 `\paren{force}`。指定方式為 `force = yes/force = no`  
不同的指定之間用逗號 `\paren{,}` 分割

SourceCode	Result
<code>\definefont[song][name:sourcehanseri]</code> <code>\song 這是定義後的字體。</code>	這是定義後的字體。

除了上面這些命令之外，還有一個可以立刻啟用字體的命令——`\definedfont`。和之前命令的區別是：該命令 \* 可以設置字體，並立 \* cmd:definedfont  
即啟用。`\definefont` 需要定義一個名稱，並在使用該名稱定義的命令後才可以啟用字體。`\definefontfamily` 則是定義一系列字體，只有使用 `\setupbodyfont` 或者 `\switchtobodyfont` \*\* 才可以啟用 \*\* cmd:switchtobodyfont  
定義的字體。

#### 4 使用 ConT<sub>E</sub>Xt 預設好的打字集

在 ConT<sub>E</sub>Xt 中已經設置了一些打字集。主要分佈在文件 `type-imp-mscore.mkiv` 和 `type-imp-osx.mkiv` 中。他們可以通過下面的命令啟用：

```
%%% for type-imp-mscore.mkiv
\setupbodyfont[mschinese]% 將啟用華文字體
%%% for type-imp-osx.mkiv
\setupbodyfont[stsong]
\setupbodyfont[stheiti]
\setupbodyfont[stkaiti]
\setupbodyfont[stfangsong]
%%% 選擇以上四種當中的任一種啟用。
```

## 2 常見問題及解決方式

### 1 在系統中全局安裝打字集，而不必每次在導言區引入打字集

這似乎已經不起作用，請不必嘗試。此問題僅作存檔

ConT<sub>E</sub>Xt 可以通過命令 `\setupbodyfont` 或者命令 `\switchtobodyfont` 來直接調用系統隨附的打字集。但用戶自己設定的打字集則必須在導言區使用 `\usetypescriptfile` 來提前引入。想要在系統範圍內安裝，實現和預設字體一般的引用效果，需要將用戶自己設計的打字集放置在系統目錄下。

首先,我們找到系統目錄的路徑。打開該路徑後,在該路徑下創建字體目錄。

```
mtxrunc --resolve-path 'TEXMFLOCAL'% 尋找系統路徑
/opt/context/tex/texmf-local % 路徑結果
cd /opt/context/tex/texmf-local mkdir -p tex/context/fonts/mkiv/
% 創建字體目錄
```

現在,把用戶創建的打字集複製到該路徑下。當然,可以對這些打字集分類到不同的目錄下。由於我們已經將新檔案新增到系統目錄中,我們必須重建檔名資料庫。

```
$ mtrunc --generate
```

這將需要一點時間,並將大量訊息列印到終端。之後,您可以檢查樹中是否檢測到新檔案:

```
$ mtrunc --find-file type-imp-garamond.mkiv
% 查詢 type-imp-garamond.mkiv 文件
/tex/texmf-local/tex/context/fonts/mkiv/type-imp-garamond.mkiv
```

如果這一切都有效,您就可以對文件進行排版了。打字稿應由 `\setupbodyfont` 自動找到(不需要 `\usetyescriptfile` 或類似命令)。

## 2 macOS 系統中華文字體的若干問題

**字體合集稿中設置了華文字體卻不起作用** macOS 系統中已經預製了華文字庫,包括:宋體、黑體、楷體、仿宋、行楷等。通過命令 `mtxrunc --script font --list --all stkaiti` 來查詢楷體的話,會返回空值。也就是說,即便我們設置了對應的字體合集稿,在最終的 PDF 中也不會生成任何該字體的文字。一種解決辦法是卸載該字體並重新安裝。具體過程如下:

通過命令 `fc-list stkaiti` 來查看華文楷體安裝的位置。結果如下(此處的位置已經是下載重新安裝後的位置了。)

```
fc-list stkaiti% 命令
/Library/Fonts/Kaiti.ttc: STKaiti:style=Regular,標準體,Normal,
レギュラー,常规体
% 命令結果,/Library/Fonts/Kaiti.ttc 就是位置了。此處命令結
果有刪減部分樣式的外文名稱。
```

知道了華文楷體的安裝位置後,就可以通過訪達跳轉到該位置了:打開訪達,使用快捷鍵 `Shift + Command + G` 打開移動到文件位置的窗口,複製華文楷體的具體位置到此處,回車後就可以跳轉到該文件的所在位置了。把這個文件拖動到任意其他文件夾,就可以卸載該字體。之後重新安裝該字體即可。

可能用得到的信息:



1. Sonoma 系統的預製字體
2. `fc-list` 命令的使用方法

字體合集稿中設置了華文字體卻只有粗體這是由於 macOS 系統預裝的華文字體是 `ttc` 格式。而 `ConTeXt` 似乎對這種格式支持度並不完美<sup>\*</sup>, 只能夠讀取一部分的格式。暫時的解決辦法是將 `ttc` 字體文件<sup>\*\*</sup> from mailing list for `ConTeXt` users 解包為 `ttf` 或者其他格式的字體, 然後重新安裝解包後的字體即可。

第二種解決辦法: \*

\* 答案來源: *Using Gill Sans on macOS* <sup>✎</sup>

一般情況下, 使用 `mtxrun --generate & mtxrun --script fonts --reload` 來生成字體文件索引。但這種情況話, 無法正常檢測 `TTC` 字體的子集。使用 `mtxrun --script fonts --reload --force` 來替代上述命令就可以強制檢測後生成字體文件索引。

需要注意的是, 使用 `--force` 參數後, 之後安裝新字體並重建索引時也需要使用 `--force` 參數來重建。否則, `ConTeXt` 依然無法正確的識別 `TTC` 字體檔中的子字體。

### 3 為什麼 `it` 和 `bf` 命令無法聯合生效?

`ConTeXt` 似乎無法讓 `it` 和 `bf` 聯合生效。只有最後一個命令會生效。例如:

SourceCode	Result
<code>{\bf\it 生效的是 it 命令}\}</code> <code>{\it\bf 生效的是 bf 命令}</code>	生效的是 <i>it</i> 命令 生效的是 <b>bf</b> 命令

想要獲得聯合效果, 請使用 `\bi`, `\boldface \italicface`。

SourceCode	Result
<code>{\bi 這是 bold 和 italic 的聯合效果}\}</code> <code>{\boldface\italicface 這是 bold 和 italic 的聯合效果}\}</code> <code>{\italicface\boldface 這是 bold 和 italic 的聯合效果}\}</code>	這是 <b><i>bold</i></b> 和 <i>italic</i> 的聯合效果 這是 <b><i>bold</i></b> 和 <i>italic</i> 的聯合效果 這是 <b><i>bold</i></b> 和 <i>italic</i> 的聯合效果

### 4 我的選定的字體組合中沒有對應的斜體和粗體, `ConTeXt` 中可以設置偽斜體和偽粗體模擬效果嗎?

不同於英文字體一般在設計時會設計 `normal`、`bold`、`italic`、`small-caps` 等一系列樣式。中文字體由於數量龐大的漢字, 在設計時, 一般只會設計一種樣式。同時, 中文字體中一般沒有斜體的概念。對於斜體的處理, 或者是使用偽斜體的方式, 模擬斜體樣式; 或者是乾脆換一種字體來表示切換了樣式。

回到問題本身<sup>\*\*</sup>, 在 `ConTeXt` 中, 可以使用兩種方式來設置偽斜體和偽粗體: `font feature` 和 `effect`。<sup>\*\*</sup> 答案來源: *define a bf-compatible fake-bold switcher* <sup>✎</sup>



SourceCode	Result
<pre>\definedfont [file:lmroman12-regular.otf] @@ no feature:\\ by add feature "boldened", you can see the word has been boldened!  \definedfont [file:lmroman12-regular.otf*boldened] @@ boldened feature:\\ by add feature "boldened", you can see the word has been boldened!  \definedfont[file:lmroman12-bold.otf] @@ the real bold font</pre>	<p>@@ no feature: by add feature "boldened", you can see the word has been boldened!</p> <p>@@ boldened feature: by add feature "boldened", you can see the word <b>has been boldened!</b></p> <p>@@ the real bold font</p>

`boldened` 還被設置了不同數值的粗度:`boldened-10`,`boldened-15`, `boldened-20`,`boldened-25`, `boldened-30`。默認下的 `boldened` 就是 `boldened-30`。這些 `feature` 也可以用於其他設置字體的命令:  
`\definefontfamily` 以及 `typescript` 之中。

SourceCode	Result
<pre>@@ no effect:\\ by add effect "thicker", you can see the word has been boldened!  \defineeffect[thicker] [alternative=both, rulethickness=0.3pt] \define\fakebf{effect[thicker]} {\fakebf @@ by add effect "thicker", you can see the word has been boldened!}  {\bf @@ the real bold font}</pre>	<p>@@ no effect: by add effect "thicker", you can see the word has been boldened!</p> <p>@@ by add effect "thicker", you can see the <b>word has been boldened!</b></p> <p>@@ the real bold font</p>

如果想要同時添加偽斜體和偽粗體效果,可以使用:

SourceCode	Result
<pre>@@ no effect:\\ by add feature "fakebolditalic", you can see the word has been boldened!  {\definefontfeature [fakedbolditalic] [default] [effect={width=0.30,auto=yes,slant=.2}] }%\feature[=][fakedbolditalic] @@ by feature "fakebolditalic", you can see the word has been boldened!}}  {\bi @@ the real bold font}</pre>	<p>@@ no effect: by add feature "fakebolditalic", you can see the word has been boldened!</p> <p>@@ by feature "fakebolditalic", you can see the word has been boldened!</p> <p>@@ the real bold font</p>

## 5 無法正常開啓 *OpenType Font* 的連字(*liga*)特性

可以試着在設置字體特性時添加 `mode=node`。例如:

```
\definefontfeature
[hant-latin-default]
[hans-default][mode=node,liga=yes]
```

## 3 附錄 3: 字型、字體、字形

- **Typeface** A typeface is a particular set of glyphs or sorts (an alphabet and its corresponding accessories such as numerals

and punctuation) that share a common design.

- **Font** A font is a particular set of glyphs within a typeface.

這是一組相似性很高的概念。**Typeface** 指的是一組具有特定設計理念的設計;**Font** 則是這組設計中的一個子集。例如: **Garamond Typeface** 是具有不同字體尺寸 (**font size**), 不同字體樣式 (**font style**) 和不同字體字重 (**font weight**) 的一組風格 (**type**)。這種設計風格的特點是小寫字母『e』上的圓圈很小, 而且很閉合。大寫字母類似於羅馬方形大寫字母, 字母『M』有向外的襯線。在一些字母上, 如字母『R』字母的腿延伸到字母的其餘部分。這些特徵使 **Garamond** 具有易於識別的外觀。而 **Font** 則是指這組設計風格中的一組具體設計。例如字號為 10pt, 字體樣式為 **italic** 且字重為 **bold** 的一組設計就被成為字體。但是在實際的使用中, 兩者經常混同。出現這組相似度高的概念原因大體是因為活版排字時期遺留下來的。活版排字時期在設計字體時, 由於物理的限制, 只能夠設計出幾種字號的同一種風格的字體。這種不同的字號或者字重的一組設計風格就被成為 **Typeface**。隨著信息技術的發展, 現在可以將不同字號、字重、樣式都壓制在一個文件當中, 就模糊掉了兩者的區別。

對於不是從事專業設計 (**typography**) 的人來, 沒有必要區分這兩個概念。比較重要的是下面的一組關於字體分類的概念, 透過這組概念我們可以合理的搭配字體, 設計出或正式或古典的頁面文字風格。但同時需要注意的是, 這些分類方式並不是絕對不可改變的。他們大多具有主觀性或根據部分特點進行歸類。但瞭解這些可以幫助我們去更好的設計不同的風格。

- **Serif typeface** **Serif** 字體, 即襯線字體, 因其字形端點帶有襯線 (**serif**) 而得名。字形具有豐富的變化。同時由於這種設計起源於古羅馬, 又被成為 **Roman typeface**。如果要和中文字體進行類比的話, 宋體字和襯線體具有相同的特點。他們都具有豐富的設計和點線變化。在日文字體環境話, 被稱為明朝體 (**Mincho**)。觀察下面兩段話, 就可以看出相較於無襯線體, 襯線體的字形端點處具有豐富的粗細變化, 且端點類似於三角。

比較常見的 **Serif** 字體有以下幾種, 他們基本都具有不同的變體(不同的公司以該種風格為主進行了特別設計, 推出了自己公司的版本)。

襯線字體隨着時間的發展, 擁有了許多不同的子類, 但這些都是針對西文字體環境而言。中文字體環境下的字體分類多可以依照書法和印刷進行區分。就西文字體而言, 有 **Venetian Roman**、**Old Roman**、**Traditional Roman**、**Modern Roman**、**Slab Roman** 這幾種子類。這幾種子類之間並不是那麼容易進行區分, 因此, 我們可以通過 **serif** 的特徵只將其分為三類:

Arial Avenir DIN Pro  
Gill Sans Helvetica Neue  
Optima Futura

表 2.2 襯線體的分類和特點




	Bracketed serif	Hairline serif	Slab serif
			
	襯線和文字線的交匯處具有像三角一樣的形狀，給人溫和的印象	襯線並不具有太多變化，纖細且平直的襯線給人清晰、藝術的印象	襯線處是長且粗的形狀，給人穩定、強力感
<div>1. Bracketed serif: 這類字體具有類似三角形的襯線，給人溫暖、柔和的感覺。</div> <div>2. Hairline serif: 這類字體的襯線為直線，看起來更加清爽、明晰，同時給人纖細、藝術的感覺。</div> <div>3. Slab serif: 這類字體不具有多變的襯線，其襯線是長且粗的。給人強力、穩定的感覺。</div>			
ヴェネチア・ローマン			
15 世紀に作られたセリフ体。			
右肩上がりの「e」の横線や、斜めに傾いた「O」の軸など、平筆やハケで書かれた字のニュアンスが残っている。			
	ABCDEORW	abcde 123	Adobe Jenson
オールド・ローマン			
16 世紀以降の活字をもとに作られたセリフ体。			
「e」の横線は水平で、ヴェネチア・ローマンほどクセがない。			
	ABCDEORW	abcde123	EB Garamond
トランジショナル・ローマン			
オールド・ローマンとモダン・ローマンの移行期のセリフ体。			
細い線と太い線のコントラストが大きく、よりモダンな印象に。「O」の軸も垂直。			
	ABCDEORW	abcde123	Baskerville URW
モダン・ローマン			
18 世紀後半以降の活字をもとに作られたセリフ体。			
手書きの名残が少なく、細いヘアラインセリフが用いられている。			
	ABCDEORW	abcde123	Didot
スラブセリフ・ローマン			
19 世紀以降に作られたセリフ体で、長方形の太いセリフが特徴。			
線幅の太さにも大きながない。			
	ABCDEORW	abcde123	Rockwell

表 2.3 襯線體的分類和實例

- **Sans-serif typeface** Sans-serif 字體, 經常會簡單的稱為 Sans 字體。sans 是法語中「無」的意思。因此, sans-serif 就是不含有襯線的字體。在設計上由於其沒有特別豐富的變化, 線條比較平直, 因而具有較好的區分度。在誕生初期, 主要用於廣告和標識。現如今, 由於其設計上的特點沒有特別的變化, 易於觀看, 被大量的應用於屏幕顯示。類比到中文字體, 黑體在設計上就是源於這種風格。同時, 在日文字體環境中, 經常會被稱為哥特體(Gothic)<sup>1</sup>。

和襯線體相同, 隨着歷史的發展, 無襯線題也是經歷多次的變化並形成了幾種不同的風格子類。分別是: Grotesque Sans-Serif、Realist Sans-Serif、Humanist Sans-Serif、**Geometric Sans-Serif**。

比較常見的 Sans-serif 字體有以下幾種, 他們基本都具有不同的變體(不同的公司以該種風格為主進行了特別設計, 推出了自己公司的版本)。

Mono

## 4 調整字體

### 1 切換字體 (alternative)

#### 命令速覽

- 命令 `rm` 的主要功能是切換字體為襯線體 / 宋體, 參見 `cmd:rm`
- 命令 `ss` 的主要功能是切換字體為無襯線題 / 黑體, 參見 `cmd:ss`
- 命令 `tt` 的主要功能是切換字體為等寬字體, 參見 `cmd:tt`
- 命令 `hw` 的主要功能是切換字體為手寫體, 參見 `cmd:hw`
- 命令 `tf` 的主要功能是使用默認字重, 參見 `cmd:tf`
- 命令 `bf` 的主要功能是**切換為粗體**, 參見 `cmd:bf`
- 命令 `it` 的主要功能是**切換為意大利體**, 參見 `cmd:it`
- 命令 `bi` 的主要功能是**切換為意大利體且加粗**, 參見 `cmd:bi`
- 命令 `sl` 的主要功能是**切換為斜體**, 參見 `cmd:sl`
- 命令 `bs` 的主要功能是**切換為斜體且加粗**, 參見 `cmd:bs`
- 命令 `sc` 的主要功能是使用小型大寫字母 SMALL CAPS, 參見 `cmd:sc`
- 命令 `em` 的主要功能是對內容進行強調, 默認為斜體, 參見 `cmd:em`
- 命令 `tfa` 的主要功能是使用默認字重, 字號為 `a`, 默認字號為: `xx/x/a/b/c/d`, 同時 `tf` 也可以替換為上述部分命令, 參見 `cmd:tfa`

在上一節中, 我們已經能夠在 ConTeXt 系統中使用字體並正確的顯示中文了。但是, 還記得嗎? 我們在設置字體時, 選擇了三種不同的

<sup>1</sup> Gothic 在歐洲指的是 Blackletter。Grotesk 和 Gothic 偶爾也會使用, 但現在一般情況下統一稱為 Sans-serif。

Arial Avenir DIN Pro  
Gill Sans Helvetica Neue  
Optima Futura

字體: 襯線體(**rm**)、非襯線體(**ss**)以及打印機體(**tt**)。如果我們想要在文檔中中切換這些已經設定好的字體,可以使用下面的命令進行切換。

使用 **\rm**、**\ss**、**\tt** 就可以切換到自己想要的字體。需要注意的是, **\rm**、**\ss**、**\tt** 是全局命令,即在不對需要切換的字體命令設定範圍時<sup>\*\*</sup>,他將自動將該命令之後的所有文字切換到該字體,除非某些部分有特殊設定。例如:

<sup>\*\*</sup> 使用花括號 **{}** 就可以框定命令作用的範圍,即命令僅在範圍內生效。

SourceCode	Result
<pre>\tfx% \rm 通過命令 \type{\rm} 進行設定。現在是襯線體。 \par % 因為下一行切換到了 \ss,所以此為襯線體 \ss 通過命令 \type{\ss} 進行設定。現在是非襯線體。 \par % 因為下一行切換到了 \tt,所以此為非襯線體 \tt 通過命令 \type{\tt} 進行設定。現在是打印機體。 \par 這一行沒有進行特殊的設定, 但這一將繼承上一行的設定, 即這一將會被設定為打印機體 \paren{\tt}。</pre>	通過命令 <b>\rm</b> 進行設定。現在是襯線體。 通過命令 <b>\ss</b> 進行設定。現在是非襯線體。 通過命令 <b>\tt</b> 進行設定。現在是打印機體。 這一行沒有進行特殊的設定,但這一將繼承上一行的設定,即這一將會被設定為打印機體( <b>tt</b> )。

因此,對於只需要臨時變換字體的情況,需要注意用花括號(**{}**)把需要切換的範圍圈定出來。例如:

SourceCode	Result
<pre>{\rm 通過命令 \type{\rm} 進行設定。現在是襯線體。 }\par % 因為使用花括號圈定了需要切換的範圍,所以花括號內為 襯線體 {\ss 通過命令 \type{\ss} 進行設定。現在是非襯線體。 }\par % 因為使用花括號圈定了需要切換的範圍,所以花括號內為 非襯線體 {\tt 通過命令 \type{\tt} 進行設定。現在是打印機體。 }\par % 因為使用花括號圈定了需要切換的範圍,所以花括號內為 打印機體  這一行沒有進行特殊的設定, 且之前的切換字體都已經進行了範圍圈定, 因此這一將繼承默認字體。 即這一將會被設定為襯線體 \paren{\rm}。</pre>	通過命令 <b>\rm</b> 進行設定。現在是襯線體。 通過命令 <b>\ss</b> 進行設定。現在是非襯線體。 通過命令 <b>\tt</b> 進行設定。現在是打印機體。 這一行沒有進行特殊的設定,且之前的切換字體都已經進行了範圍圈定,因此這一將繼承默認字體。即這一將會被設定為襯線體( <b>rm</b> )。

<sup>\*</sup> 這兩種稱呼並不出現在官方的文檔之中。如此稱呼只是為了說明這兩種命令的影響範圍。

和全局命令相對的就會有局域命令<sup>\*</sup>。正如他們的名字一樣,全局命令一旦使用就將自啓用之處起,作用於其後的全部內容。但局域命令則只會影響其後所圈定範圍的內容。如果不圈定局域命令的範圍,或者只會影響其後的一個字,或者會導致系統拋出錯誤提示。一般情況下,全局命令的使用方法是直接在想要啓用處放置即可。如果想要圈定範圍,讓全局命令只在圈定的範圍內生效,就可以用**{}**來限制其生效範圍。試試下面的命令,看看他們會如何影響其後的內容?

SourceCode	Result
------------	--------



會被 `\type{\rm}` 命令影響嗎? `\par`  
 ,用來查看 `\type{\rm}` 會不會影響這一  
 段。  
 會被 `\type{\rm}` 還是 `\type{\ss}` 命令  
 影響嗎?  
 ,用來查看 `\type{\rm}` 或者 `\type{\ss}`  
 會不會影響這一  
 段。  
 猜一猜這一段會被 `\tt` 命令影響嗎?  
 這一段呢? 會被 `\tt` 命令影響嗎

瞭解了這些之後,在使用全局或者局域命令時就需要記得為他們  
 圈定合適的範圍。當然,切換了字體,但有沒有使用花括號圈定範圍,  
 想要恢復到默認的字體,可以使用 `\restoreglobalbodyfont` \*\*。

\*\* 該命令是 ConT<sub>E</sub>Xt 提供的默認回退命令。通過該命令會將樣式、大小、字族等都恢復到默認狀態。但我們可以仿照這個命令的官方定義來自定義一系列恢復命令。

SourceCode	Result
<code>\ss\itx\the\font Sans &amp; italic &amp; asize \\ \restoreglobalbodyfont\the\font what kind ?</code>	<code>&lt;170: AdobeKaitiStd-Regular @ 9.0pt&gt;Sans &amp; italic &amp; asize &lt;1: SourceHanSerifCN-Regular @ 11.0pt&gt;what kind ?</code>

當然,ConT<sub>E</sub>Xt 也提供了一些助記命令來方便進行字體切換。

<code>\rm: RomanStyle</code>	<code>\serif: SerifStyle</code>
<code>\ss: SansSerifStyle</code>	<code>\sans: SansSerifStyle</code>
<code>\tt: TypeWriterStyle</code>	<code>\mono: MonospacingStyle</code>
<code>\hw: TypeWriterStyle</code>	<code>\handwritten: HandWrittenStyle</code>

表 2.4 切換樣式命令助記詞

## 2 字體字號 (size)

在 Word 中設定字號時,提供了兩種單位的字號:一種是類似於「一號字」、「五號字」之類的字號;一類是類似於「10pt」、「22pt」之類的字號\*\*。一般情況下,使用的字號大小為:五號字或者四號字,換  
 算 \* 成 pt 值為 10.5pt 或者 11pt。不同字號標準之間的換算如表 2.5:

\*\* pt 一般翻譯為磅

\* 換算關係為:1pt = 0.35146mm, 1mm ≈ 0.83pt, 1in = 72pt, 字高 1 釐米的字符,其磅數值大約為 28.3pt。

ConT<sub>E</sub>Xt 系統在不指定字體大小的情況下,默認選擇 12pt 字體。  
 如果想要設定全局文本字體可以在設定字體時,同時設定字號大小。

```
\setupbodyfont [_OPT_ARGI_, _OPT_ARGII_, _OPT_ARGIII_]
_OPT_ARGI_: 指定打字集名稱。例如: source-hant
_OPT_ARGII_: 指定字體樣式。例如:rm,ss,tt
_OPT_ARGIII_: 指定字體字號。例如: 11pt
```

如果想要臨時更換字體和字號,可以使用 `\definedfont` \*\* 命令。 \*\* `cmd:definedfont`  
 請注意,該命令不同於 `\definedfont` 命令。`\definedfont` 用於定義字體,`\definedfont` 則用於定義字體後並立即切換到該字體。

除了上述的兩種可以在正文中切換字體的命令外,我們可以使用 `\switchtobodyfont` \*\* 來快速切換字體和字號。該命令將  
 在設置後立即啟用,並改變其後的字體。該命令的參數設置和 `\setupbodyfont` 的參數完全相同。不同的是,`\setupbodyfont`

\*\* `cmd:switchtobodyfont`

表 2.5 字號換算簡表

中文字號	英文字號(磅)/pt	毫米 /mm
初號	42	4.82
小初	36	12.7
一号	26	9.17
小一	24	8.47
二号	22	7.76
小二	18	6.35
三号	16	5.64
小三	15	5.29
四号	14	4.94
小四	12	4.23
五号	10.5	3.7
小五	9	3.18
六号	7.5	2.56
小六	6.5	2.29
七号	5.5	1.94
八号	5	1.76

用來切換全部的樣式，它將會應用到正文、腳註等處，但 `\switchtobodyfont` 只能夠在正文處進行更改。還有一個需要注意的是，`\setupbodyfont` 和 `\switchtobodyfont` 接受的字體名稱參數是來自於打字集的設定，而 `\definefont` 和 `\definedfont` 接受的字體名稱則是通過命令行查找到的 `fontfamily`、`fontname` 等的字體文件的內部名稱。

```
{\definedfont [Serif*default at 12pt] Serif with default features}
{\definedfont [name:sourcehanserifcnregular*default at 12pt]
Sourcehanserifcnregular.}
\switchtobodyfont[iwona,ss,14pt] Quickly switch to Iwona font.
```

如果在對字號精確度要求不高的情況下，可以使用 ConTeXt 系統默認提供的一系列的字號。這些字號命令需要和 `\tf\rm\it` 等命令相結合，從小到大依次是：`\tfxx`、`\tfx`、`\tf`、`\tfa`、`\tfb`、`\tfc`、`\tfd`。以 `\tf` 為基準，向左減少，向右增加。

```
{\tfa \type{\tfa} fontsize}\par
{\tt\itx \type{\tt\itx} fontsize \& alternative}
```

特別的，這些字號命令可以和之前的 `\rm`、`\it` 之類的命令結合使



用。因此,將產生大量的可用使用的命令變體:

```
\rma 使用襯線體,字號為 a(縮放至 1.2 倍大小);
\restoreglobalbodyfont 恢復默認樣式
\ssb 使用非襯線體,字號為 b(縮放至 1.44 倍大小);
\restoreglobalbodyfont 恢復默認樣式
\rmita 使用襯線體,意大利體,且字號為 a;
\restoreglobalbodyfont 恢復默認樣式
\ssbixx 使用非襯線體,粗體,且字號為 xx;
```

除了這幾個字號之外,ConT<sub>E</sub>Xt 還定義了幾個具有語義的命令:

`\scriptscript`, `\script`, `\small`, `\big`。根據 Wiki 的說明, `\scriptscript`, `\script` 適用於數學公式模式。

你可以通過 `\showbodyfont` 命令來查看族類、樣式、字號,其結果可見下表:

```
\showbodyfont [8pt]
```

[sourcehancn] [8pt]													
	\tf	\tf	\bf	\sl	\it	\bs	\bi	\tfx	\tfxx	\tfa	\tfb	\tfc	\tfd
\rm	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag
\ss	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag
\tt	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag
\mr	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag

同時,也可以通過 `\showbodyfontenvironment` 來查看在不同基準字號下不同命令字號的具體大小。其結果可見下表:

```
\showbodyfontenvironment
```

[sourcehancn] [11.0pt]						
text	script	scriptscript	x	xx	small	big
10.5bp	9.00bp	7.50bp	8.80bp	6.60bp	8.43pt	12.65pt
10pt	7pt	5pt	8pt	6pt	8pt	12pt
11.0bp	8.80bp	6.60bp	8.80bp	6.60bp	8.83pt	13.25pt
11pt	8pt	6pt	9pt	7pt	9pt	12pt
12.0bp	9.60bp	7.20bp	9.60bp	7.20bp	9.63pt	14.45pt
12pt	9pt	7pt	10pt	8pt	10pt	14.4pt
14.4pt	11pt	9pt	12pt	10pt	12pt	17.3pt
17.3pt	12pt	10pt	14.4pt	12pt	14.4pt	20.7pt
20.7pt	14.4pt	12pt	17.3pt	14.4pt	17.3pt	20.7pt
4pt	4pt	4pt	4pt	4pt	4pt	6pt
5pt	5pt	5pt	5pt	5pt	5pt	7pt
6pt	5pt	5pt	5pt	5pt	5pt	8pt
7pt	6pt	5pt	6pt	5pt	5pt	9pt
8pt	6pt	5pt	6pt	5pt	6.4pt	9.6pt
9pt	7pt	5pt	7pt	5pt	7pt	11pt

如果覺得目前系統設定的字號不容易記憶,也可以設定自己的速記命令。設定方式有兩種: 設定縮放倍率或者設定具體字號數值。下面這種方式設定的字體是在基準字號 \* 的標準上進行的縮放。當需要明確定義某個字號命令的字號數值時,也需要指定基準字號。例如,我們需要定義一個基於本文檔基準字號縮放 1.5 倍的字號(當前文檔字號為 11pt),並將這個字號命令定義為 `sectionsize`。

\* 不指定字體大小的情況下,默認選擇 12pt 字體。

SourceCode	Result
<pre>\definefontsize[sectionsize] % define a font size for section \definebodyfontenvironment[default][sectionsize=1.5] % let sectionsize scale 1.5. Basic fontsize is \fontbody.\par \tfsectionsize Normal typeface with fontsize scaled 1.5,and the fontsize is \fontsize</pre>	<p>Basic fontsize is 11pt.</p> <p><b>Normal typeface with fontsize scaled 1.5,and the fontsize is sectionsize</b></p>

當然,如果目前所提供的字號之中沒有適合的也可以自行定義所需的字號。這時就需要同前文所說,必須指定需要縮放的基準字號。

SourceCode	Result
<pre>\definefontsize [anothersectionsize] \definebodyfontenvironment [11pt] [anothersectionsize=15pt]  Basic fontsize is \fontbody. \tfanothersectionsize Normal typeface with fontsize scaled 1.5,and the fontsize is \fontsize</pre>	<p>Basic fontsize is 11pt. Normal typeface with fontsize scaled 1.5,and the fontsize is anothersectionsize</p>

## 5 段落和空白

不同於 Word 在書寫文檔時回車一次就代表一次換行,在  $\text{\TeX}$  的世界裏,回車一次只會在同一行上產生一個空格,回車兩次才會在最終的 PDF 文件中表示一次換行。可以試試下面的這幾段文字,他們分別會如何顯示?

SourceCode	Result
<pre>% 第一個例子 昔日少年郎、今朝坐高堂。風吹池水皺、鏡照舊人裳。 % 這是一行文字。  % 第二個例子 昔日少年郎、今朝坐高堂。 風吹池水皺、鏡照舊人裳。 % 這裏只進行了一次回車,結果仍然是一行文字。  % 第三個例子 昔日少年郎、今朝坐高堂。  風吹池水皺、鏡照舊人裳。 % 這裏進行了兩次回車,結果將是兩行文字。</pre>	<pre>昔日少年郎、今朝坐高堂。風吹池水皺、鏡照舊人裳。 昔日少年郎、今朝坐高堂。風吹池水皺、鏡照舊人裳。 昔日少年郎、今朝坐高堂。 風吹池水皺、鏡照舊人裳。</pre>

此外,在 Word 中想要一段較長的垂直距離時,可能會多次回車,來生成多行以達到視覺上的垂直距離的效果。但是在  $\text{\TeX}$  中你會發現:無論回車多少次,最終都只是相當於空了一行。這是因為  $\text{\TeX}$  視多行回車為一行。如果你想要生成這種效果,可以使用 `\blank` 命令,稍後也會有簡單地說明。

SourceCode	Result
<pre>這是第一行。接下來將會空三行繼續。  這是第四行。在最終的 PDF 中,只會有一個空行。</pre>	<pre>這是第一行。接下來將會空三行繼續。 這是第四行。在最終的 PDF 中,只會有一個空行。</pre>

除了使用空一行來換行之外,還可以使用命令的方式:

```
昔日少年郎、今朝坐高堂。\\% <<--
風吹池水皺、鏡照舊人裳。\\fancybreak (1)
```

```
昔日少年郎、今朝坐高堂。\\par% <<--
風吹池水皺、鏡照舊人裳。\\fancybreak (2)
```

```
昔日少年郎、今朝坐高堂。\\break% <<--
風吹池水皺、鏡照舊人裳。\\fancybreak (3)
```

需要注意的是,以上三種方式嚴格意義上來說,只有 `\par` 是真正意義上的生成一段新段落。`\\`、`\break` 的效果是在此處立刻斷行。仔細觀察下面的實際效果,你會發現 `\\` 斷行後的第二行沒有自動縮進二個字符的寬度。`\break` 斷行後,除了第二行沒有自動縮進外,第一行文字也被大幅度拉伸。 \*\*

\*\* 這是因為 `\\` 僅執行填充剩餘空白並斷行, `\break` 則僅進行斷行,除此之外,什麼都不做。關於這一部分內容需要瞭解 `glue`(膠水)。

.....

昔日少年郎、今朝坐高堂。  
風吹池水皺、鏡照舊人裳。 ..... (1)  
昔日少年郎、今朝坐高堂。  
風吹池水皺、鏡照舊人裳。 ..... (2)  
昔 日 少 年 郎 、 今 朝 坐 高 堂 。  
風吹池水皺、鏡照舊人裳。 ..... (3)  
.....

因此,如果想要生成新一段落,只有空一行和 `\par` 這兩種方式是推薦的。

前面提到如果你想要產生多行空白段落,在 Word 中可以多次空行來實現。但在  $\text{\TeX}$  中,有多種方式可以實現:

- 1. 通過命令 `\blank[_OPT_ARG_]` 來生成空白行。例如:  
`\blank[2*line]` 將會空兩行。`\blank[3*halfline]` 將會空 1.5 行。 \*\*
  - 2. 通過命令 `\godown[_DIMENSION_]` 來生成空白行。例如:  
`\godown[2em]` 將會生成一段高 2em 的空白行,大約等同於空兩行。 \*
  - 3. 通過命令 `\vskip _DIMENSION_` \*\* 或者 `\vspacing[_OPT_ARG_]` 來生成空白行。 \*\* 例如: `\vskip 2em` 或者 `\vspacing[2em]`。
- 我們可以通過下面示例來大致感知上述命令的用法:

\*\* 有一個和 `cmd:blank` 相似的命令 `cmd:emptylines:\emptylines[_NUMBER_]`,可以生成對應數字行的空行。

\* `cmd:godown`

\*\* `_DIMENSION_` 是指尺寸。含有該參數的命令可以接受長度單位: `ex/ em/ pt/ bp/ cm/ mm` 等。這些長度的具體說明以參照附錄中的說明。對於中文,一般使用 `1em` 表示一個漢字長度。其他的 `ex/ pt/ bp` 可以不考慮。

\*\* `cmd:vskip`  
`cmd:vspacing`

SourceCode	Result
我們推薦用命令 <code>\type{\blank[_OPT_ARG_]}</code> 或者 <code>\type{\vspacing[_OPT_ARG_]}</code> 來生成空行,這兩者是同義的。這是通過 <code>\type{\blank[2*line]}</code> 生成的空行。 <code>\blank[2*line]</code> % 這裏將空兩行 這裡是隨便什麼。無須關注。	我們推薦用命令 <code>\blank[_OPT_ARG_]</code> 或者 <code>\vspacing[_OPT_ARG_]</code> 來生成空行,這兩者是同義的。這是通過 <code>\blank[2*line]</code> 生成的空行。  這裡是隨便什麼。無須關注。

SourceCode	Result
我們應該儘可能的避免使用 <code>\type{\vskip _DIMENSION_}</code> 來進行空行。這是 $\text{\TeX}$ 系統的原始命令 <code>\paren{primitive}</code> 。這是通過 <code>\type{\godown[2em]}</code> 生成的空行。 <code>\godown[2em]</code> % 這裏大致空兩行,2em 是指兩個字符的寬度。 % 因此不一定等於兩行 這裡是隨便什麼。無須關注。	我們應該儘可能的避免使用 <code>\vskip _DIMENSION_</code> 來進行空行。這是 $\text{\TeX}$ 系統的原始命令( <code>primitive</code> )。這是通過 <code>\godown[2em]</code> 生成的空行。  這裡是隨便什麼。無須關注。

SourceCode	Result
<p>我們推薦用命令 <code>\type{\blank[_OPT_ARG]}</code> 或者 <code>\type{\vsparing[_OPT_ARG]}</code> 來生成空行,這兩者是同義的。這是通過 <code>\type{\vsparing[2]}</code> 生成的空行。  <code>\vsparing[2*line]</code>          % 這裏大致空兩行          我們應該儘可能的避免使用 <code>\type{\vskip_DIMENSION}</code> 來進行空行。          這是 <math>\text{\TeX}</math> 系統的原始命令 <code>\paren{primitive}</code>。</p>	<p>我們推薦用命令 <code>\blank[_OPT_ARG]</code> 或者 <code>\vsparing[_OPT_ARG]</code> 來生成空行,這兩者是同義的。這是通過 <code>\vsparing[2]</code> 生成的空行。</p> <p>我們應該儘可能的避免使用 <code>\vskip_DIMENSION</code> 來進行空行。這是 <math>\text{\TeX}</math> 系統的原始命令(<code>primitive</code>)。</p>

和空任意行視為一次換行相同,在  $\text{\TeX}$  系統中無論空多少格,都會被視為空一次格。有了這兩種特性,我們就可以在源文件(source)中通過空格或者空行的方式來對齊文本,提高源文件的可讀性。雖然這些空行和空格並不會在最終文件中呈現出來。

SourceCode	Result
<p>其實在這段文本中,有過很多次的空行和空格。          我們就可以通過這種方式來對源文件進行格式化,提高源文件的可讀性。不至於源文件一堆文字和一堆代碼堆在一起,導致無法閱讀。那麼,我們應該如何設置多個空格呢?</p>	<p>其實在這段文本中,有過很多次的空行和空格。我們就可以通過這種方式來對源文件進行格式化,提高源文件的可讀性。不至於源文件一堆文字和一堆代碼堆在一起,導致無法閱讀。那麼,我們應該如何設置多個空格呢?</p>

在 Word 中,可以使用空格來直接達到對齊的效果。但是在  $\text{\TeX}$  中,我們可以通過下面的方式(表 2.6)來設置不同長度的空格:

	生成一個空格	空格長度 文本預覽 空格長度文本預覽
~	生成一個空格	空格長度 文本預覽 空格長度文本預覽
<code>\quad</code>	生成 1em 寬度的水平長度	空格長度 文本預覽 空格長度文本預覽
<code>\qqquad</code>	生成 2em 寬度的水平長度	空格長度 文本預覽 空格長度文本預覽
<code>\quads[_NUMBER_]</code>	生成 $n * 1\text{em}$ 寬度的水平長度	空格長度 文本預覽 ( $n=3$ )空格長度文本預覽
<code>\hspace[_OPT_ARG_]</code>	常見參數 如: <code>small(0.25em)</code> , <code>medium(0.5em)</code> , <code>big(1em)</code>	空格長度 文本預覽 ( <code>opt_arg=medium</code> ) 空格長度文本預覽
<code>\hskip_DIMENSION_</code>	生成參數長度的水平長度	空格長度 文本預覽 ( <code>4em</code> )空格長度文本預覽

表 2.6 水平長度命令及示例

由於 ConT<sub>E</sub>Xt 會在排版時壓縮掉部分空格,上述命令有可能會因為排版段落時而被壓縮掉。同時,在行首的所有空格有可能無法顯示。如果你想製造一個不被壓縮的強制不會斷連空格,可以在上述命令之前添加 `\nobreak`,這樣 ConT<sub>E</sub>Xt 就會知道後續的空格是不被允許壓縮的了。本段起始部分就使用了該命令,生成了一個 1em 的水平長度。此外還需要注意的是 `\hskip` 命令是 T<sub>E</sub>X 系統中的原始命令,他允許直接在命令之後添加想要的長度單位作為他的參數,為了避免不可預知的錯誤,最好在使用時將其包裹在 `{ }` 之中。\*

上述命令除了原始命令 `\hskip` 可以生成任意的長度外,`\quads` 和 `\hspace` 都只能生成整數長度。如果需要更加細緻的水平長度,可以查看 `\hspace` 命令\*\* 和 `\definehspace` 命令\*\* 的詳細用法。

\* `\vskip` 和 `\hskip` 都是原始命令。他們都可以在其後接續長度,對應的,`\vskip` 生成垂直距離,`\hskip` 生成水平距離。為了避免錯誤,可以使用花括號包裹整個命令,也可以在長度後接續 `\relax` 來結束該命令。例如: `\hskip 1pt\relax`。

\*\* `cmd:hspace` 。

\*\* `cmd:definehspace` 。

# Index | 附录 A

## b

`\bf` 17  
`\bi` 17  
`\blank` 24  
`\bs` 17

## d

`\definedfont` 11, 19  
`\definefallbackfamily` 9  
`\definefont` 10  
`\definefontfallback` 10  
`\definefontfamily` 7, 8

## e

`\em` 17  
`\emptylines` 24

## g

`\godown` 24

## h

`\hw` 17

## i

`\it` 17

## r

`\rm` 17

## s

`\sc` 17  
`\setupbodyfont` 8, 9  
`\sl` 17  
`\ss` 17  
`\switchtobodyfont` 11, 19

## t

`\tf` 17  
`\tfa` 17  
`\tt` 17

## v

`\vskip` 24  
`\vspacing` 24



