راهنمای جامع

LaTeX

مرتضی ابطحی

عضو هیأت علمی دانشگاه علوم پایه‌ی دامغان

فهرست مطالب

فصل ۱

اساس کار

در این فصل به مبانی و اساسِ کار با LaTeX پرداخته می‌شود. مفاهیم اولیه و ساختارهای اساسی LaTeX بررسی می‌گردد. خواننده پس از مطالعه‌ی این فصل آگاهی کافی از چگونگی کارکرد LaTeX به‌دست می‌آورد که برای دنبال کردن ادامه‌ی کتاب سودمند است. در ابتدا تاریخچه‌ای از صنعت چاپ و حروفچینی ارائه می‌کنیم.

۱.۱ تاریخچه‌ی مختصری از صنعت چاپ و حروفچینی

در دوره‌ی پیش از صنعت چاپ و حروفچینی، یک کتاب با نسخه‌برداری به‌طور دستی پدید می‌آمد و انتشار می‌یافت. کار نسخه‌برداری از یک کتاب، کاری سخت و دشوار بود. ماه‌ها و حتی سال‌ها طول می‌کشید و به افراد زیادی برای این کار نیاز بود، به‌طوری که فراهم کردن چند جلد کتاب و پدید آوردن یک کتابخانه، کاری شگفت‌آور و تحسین برانگیز می‌نمود. این وضعیت تا اختراع گوتنبرگ، در سال ۱۴۴۰ میلادی، ادامه داشت.

تکامل در صنعت چاپ و حروفچینی را می‌توان به چهار دوره تقسیم کرد.

دوره‌ی گوتنبرگ، شروع ۱۴۵۰ میلادی

اساس اختراع گوتنبرگ این‌گونه بود که ابتدا نقش حروف بر اجسام مناسب کنده‌کاری می‌شد و سپس، با ریختن فلز مذاب در این قالب‌های کنده‌کاری شده، حروف فلزی ایجاد می‌شد. سپس حروف ایجاد شده کنار هم قرار می‌گرفت تا کلمات و پاراگراف‌ها شکل گیرد و یک صفحه‌ی کامل پدید آید. صفحه‌ی پدید آمده با حروف فلزی را به جوهر آغشته می‌کردند و با قرار دادن آن روی برگ کاغذ، یک صفحه از کتاب چاپ می‌شد. از آن زمان عمل چیدن حروف کنار هم، برای پدید آوردن واژه‌ها و پاراگراف‌ها، **عمل حروفچینی** نام گرفت.

تا مدتی گوتنبرگ خود از اختراعش بهره‌ای نبرد. او با سرمایه‌های غرضی کار کرد و حاصل کارش به سرمایه‌گذاران رسید. تا آنکه، در سال ۱۴۵۴ میلادی، اولین چاپ انبوه از یک کتاب، با عنوان **انجیل گوتنبرگ**، با موفقیت انجام شد.

اختراع گوتنبرگ موفقیت بزرگی در صنعت چاپ به‌حساب می‌آمد، به‌طوری که اساس کار گوتنبرگ تا قرن‌ها بدون تغییر ادامه یافت. در طول دهه‌ها صنعت چاپ و حروفچینی با سرعت شگفت‌انگیزی در سراسر اروپا گسترش یافت. پس از حدود پنجاه سال، هزاران نفر در بیش از دویست شهر اروپا در چاپخانه‌ها کار می‌کردند. تعدادی از این افراد مهارت دیده بودند و بسیاری دیگر مردمانی بودند که تنها برای بدست آوردن مقدار ناچیزی پول کار می‌کردند.

دوره‌ی انقلاب صنعتی، شروع ۱۸۷۰ میلادی

انقلاب صنعتی نوآوری بزرگی در صنعت چاپ و حروفچینی پدید آورد. نیروی بخار و حرکت دورانی جایگزین دست‌های کارگران شد. ماشین بخار همان کاری را که کارگران انجام می‌دادند، با کاهش ۸۵ درصدی در زمان، به پایان می‌رساند.

دوره‌ی چاپ به کمک عکسبرداری، شروع ۱۹۶۰ میلادی

در سال ۱۹۶۰ میلادی جهش قابل توجهی در پیشرفت صنعت چاپ و حروفچینی انجام گرفت و آن چاپ با کمک عکاسی بود. اساس کار این‌گونه بود که ابتدا، به‌وسیله‌ی دوربین عکاسی،

تصویری از صفحه‌ی مورد نظر، با اندازه‌ی دلخواه، فراهم می‌شد. فیلم حاصل را روی یک صفحه‌ی فلزی مناسب قرار می‌دادند و حاصل کار را درون یک مایع شیمیایی مخصوص غوطه‌ور می‌کردند. آن قسمت از صفحه‌ی فلزی که تصویر حروف و کلمات قرار داشت، برجای می‌ماند و سایر قسمت‌های صفحه، توسط مایع شیمیایی، اصطلاحاً خورده می‌شد. در نتیجه سطحی بدست می‌آمد که در آن نوشته‌ی مورد نظر، به صورت برجسته، نمایان بود. سطح حاصل را به جوهر آغشته می‌کردند و برای چاپ استفاده می‌نمودند. این فرایند **حروفچینی سرد** نام داشت زیرا در آن از فلز مذاب و داغ استفاده نمی‌شد.

یکی از مزایای این روش آن بود که، به‌آسانی، بزرگنمایی حروف و کلمات قابل تغییر بود و حتی می‌توانستند حروف و کلمات را روی هم چاپ کنند.

دوره‌ی حروفچینی رایانه‌ای، شروع ۱۹۷۰ میلادی

در دهه‌ی ۱۹۷۰ میلادی حروفچینی بر اساس سامانه‌های رایانه‌ای ظاهر گشت. در ابتدا بهره‌گیری از این شیوه بسیار گران بود، به‌طوری که میلیون‌ها دلار هزینه داشت و بنابراین به سختی از آن استفاده می‌شد. در اواخر دهه‌ی ۱۹۸۰ میلادی مجله‌ی Time با همکاری دو برنامه‌نویس، نرم‌افزاری با عنوان *QuarkXpress* طراحی کرد که برای حروفچینی در رایانه‌های رومیزی Macintash استفاده می‌شد. کاری که با چاقو و فیلم عکاسی و روش‌های قدیمی، روزها به درازا می‌کشید، اکنون با کمک رایانه در چند ثانیه قابل انجام بود.

۲.۱ TEX چیست؟

در حدود سی سال پیش، حروفچینی نوشتجاتی که دربردارنده‌ی فرمول‌ها و علایم ریاضی بود، کاری سخت و حتی غیر ممکن بود. ابتدا دست نوشته‌ها، به‌وسیله‌ی یک ماشین حروفچینی، آماده می‌شد و سپس فرمول‌ها و علایم ریاضی، با صرف وقت و زمان زیاد، به‌طور دستی، درون متن گنجانده می‌شد و حاصل کار به چاپخانه ارسال می‌گشت.

قبل از چاپ نهایی، کتابِ حروفچینی شده در اختیار نویسنده قرار می‌گرفت و نویسنده، برای رفع اشکالات احتمالی، آن را بررسی می‌کرد و در صورت لزوم نکاتی در حاشیه‌ها می‌نوشت و آن را جهت اصلاح برای ناشر می‌فرستاد. هیچ گونه ارتباطی بین نویسنده و مسؤول حروفچینی وجود نداشت.

در آن هنگام حروفچینی متون ریاضی، گاهی، به‌عنوان یک جریمه، برای کارکنان چاپخانه، به‌حساب می‌آمد؛ در مدت زمانی که شخص جریمه شده به‌سختی یک صفحه از متن ریاضی را حروفچینی می‌کرد، همکار وی می‌توانست بیش از ده صفحه از متن معمولی را حروفچینی کند. به‌علاوه، از آنجا که ریاضی‌دانان افرادی دقیق هستند، نوشتار حروفچینی شده تا چند بار، برای انجام اصلاحات، بین نویسنده و حروفچین، در رفت و برگشت بود.

تا اینکه TEX در سال ۱۹۸۲ میلادی، توسط دونالد کنوث[1]، از دانشگاه استنفورد[2]، منتشر شد. اما داستان TEX ریشه در سال‌های پیش از آن دارد. در سال ۱۹۶۹ میلادی، جلد اول از یک کتاب با عنوان

The Art of Computer Programming

نوشته شده توسط دونالد کنوث، که یک شاه‌کار به حساب می‌آید، به‌وسیله‌ی حروف فلزی و با شیوه‌ی قدیمی، توسط یک ماشین چاپ مربوط به قرن ۱۹ میلادی، چاپ و منتشر شد. اما وقتی ویرایش دوم از جلد دوم همان کتاب، در سال ۱۹۷۶ میلادی، با کمک فناوری عکاسی و با بکارگیری قلم‌های[3] جدید، حروفچینی گردید، لازم شد همه‌ی کتاب دوباره حروفچینی و چاپ گردد، زیرا دیگر قلم‌های قدیمی در دسترس نبود.

هنگامی که کتاب حروفچینی شده، برای بررسی نهایی، به‌دست کنوث رسید، حاصل کار برای وی ناپسند آمد. در همین روزها بود که کنوث با نمونه‌هایی از حروفچینی رایانه‌ای برخورد کرد و علاقه‌مند شد خود برنامه‌ای رایانه‌ای، برای حروفچینی متون علمی، با کیفیت خروجی بالا، پدید آورد. وی در ماه مه سال ۱۹۷۷ میلادی، شروع به کار روی یک سامانه‌ی پردازش متنی کرد، که اکنون با نام TEX و METAFONT شناخته می‌شود؛ نگاه کنید به [۱۵]، [۱۶]، [۱۷] و [۱۸].

[1] Donald Knuth

[2] Stanford University

[3] Fonts

هرچند کنوث در ابتدا TEX را برای حروفچینی ویرایش دوم کتاب The Art of Computer Programming نوشت اما وی، به‌طور کلی، هدف از پدید آوردن آن را حروفچینی متن‌های دربردارنده‌ی عبارت‌های ریاضی اعلام کرده است. کنوث در پیش‌گفتار کتابِ TEX، [۱۵]، می‌نویسد:

> TEX یک سامانه‌ی جدید رایانه‌ای است که برای حروفچینی کتاب‌ها و به‌ویژه کتاب‌هایی که دربردارنده‌ی عبارت‌های ریاضی زیادی هستند، به‌طور زیبا و ماهرانه، مناسب است ...

TEX، آن‌گونه که امروزه استفاده می‌شود، در سال ۱۹۸۲ میلادی منتشر شد و پس از آن در سال ۱۹۸۹ میلادی بهبود یافت، به‌گونه‌ای که از کراکترهای ۸ بیتی و متن‌های چندزبانه پشتیبانی کند.

خیلی زود TEX در مجامع علمی و بین هزاران نفر از دانشمندان و پژوهشگران رشته‌های مختلف علمی رواج پیدا کرد، زیرا به‌وسیله‌ی آن می‌توان هر نوع نوشته را در قالب‌های گوناگون همچون شعر، نامه، مقاله، کتاب و به‌طور کلی هر قالبی که توسط نویسنده تعیین شود، به‌طور زیبا و ماهرانه، و با کیفیت عالی، حروفچینی کرد. در حال حاضر TEX به‌عنوان استانداردی برای حروفچینی متن‌های علمی، به صورت گسترده، به‌کار می‌رود.

کارآمدی TEX، به‌ویژه، هنگامی خودنمایی می‌کند که متن مورد نظر دارای فرمول‌های ریاضی فراوان و پیچیده است. با فراگیری اندکی قواعد فرمول‌نویسی می‌توان هر عبارت ریاضی پیچیده را به‌آسانی و به‌زیبایی حروفچینی کرد. TEX از توانایی حمل‌پذیری[۴] بالایی برخوردار است، به این معنی که TEX بر گستره‌ی پهناوری از سامانه‌های رایانه‌ای قابل اجرا است و رفتار آن و نتیجه‌ی کار، در همه‌ی این سامانه‌ها، یکسان است. این واقعیتی است که در ارتباطات علمی و فنی بسیار اهمیت دارد. به‌علاوه TEX یک زبان برنامه‌نویسی هم هست و با فراگیری این زبان می‌توان به توسعه و پیشرفت آن کمک کرد.

در حدود بیست سال پیش، گوردن بل[۵] در پیش‌گفتار کتابِ [۲۰] نوشته است:

---

[4] Portability

[5] Gordon Bell

> TEX بالقوه یکی از مهمترین اختراعات در زمینه‌ی حروفچینی نوشتجات در قرن حاضر به‌حساب می‌آید. TEX معرف یک زبان استاندارد در زمینه‌ی حروفچینی رایانه‌ای می‌باشد که از لحاظ اهمیت در سطح اختراع گوتنبرگ است ...

حدود یک دهه پس از متولد شدن TEX، کنوث به‌طور رسمی اعلان کرد که TEX به سطح پایدار رسیده است و بیش از این تغییر نخواهد کرد [۲۱]. هرچند هنوز هم کنوث اشتباهات احتمالی موجود در TEX را اصلاح می‌کند اما احتمال یافتن یک اشتباه در TEX تقریباً صفر است.

۳.۱ LATEX چیست؟

در سال‌های آغازین دهه‌ی ۱۹۸۰ میلادی، لزلی لمپارت[۶] شروع به‌کار روی LATEX کرد. LATEX یک سامانه‌ی آماده‌سازی و حروفچینی نوشتار بر پایه‌ی TEX است و از TEX به‌عنوان موتور حروفچین استفاده می‌کند. در واقع LATEX گردایه‌ای بزرگ از صورت‌های توسعه یافته‌ی TEX است به‌گونه‌ای که امکاناتی در اختیار کاربر قرار می‌دهد که وی بتواند، به‌آسانی، از توانایی‌های TEX بهره‌مند شود بدون آنکه نیازی به فراگیری فرمان‌های سطح پایین TEX باشد.

فرمان‌های سطح بالای LATEX به کاربر امکان می‌دهد، به‌آسانی، بسیاری از انواع گوناگون نوشتجات را حروفچینی کند. به‌عنوان نمونه، برخی از این امکانات که LATEX، با کمک برنامه‌های جانبی، در اختیار کاربر می‌گذارد و TEX فاقد آن است عبارت است از:

> **ایجاد فهرستِ مطالب، سامانه‌ی ارجاع متقابل،[۷] ایجاد کتابنامه، ایجاد نمایه،[۸] جایگذاری تصویر،** ...

کاربر، هنگام کار با LATEX، به‌جای پرداختن به جزئیات مربوط به قالبِ ظاهری نوشتار، روی ساختار منطقی آن متمرکز می‌شود.

---

[6] Lesli Lampart

[7] Cross reference

[8] Index

۱.۳.۱ مؤلف، طراح کتاب و حروفچین

معمولاً، در فرایند انتشار یک کتاب ابتدا نویسنده دست‌نوشته‌های خود را تحویل مؤسسه منتشرکننده[۹] می‌دهد و سپس طراحِ کتاب در این مؤسسه تصمیم می‌گیرد چگونه نوشته‌ی مورد نظر حروفچینی و صفحه‌آرایی شود؛ چگونه عرض ستون‌ها، اندازه‌ی حاشیه‌ها، فضای خالی قبل و بعد از سربرگ‌ها، نوعِ قلم و خصوصیاتی مانند آن، انتخاب گردد. طراح کتاب جزئیات و اطلاعات کافی را در اختیار مسؤول حروفچینی قرار می‌دهد و حروفچین، بر اساس اطلاعات داده شده، کتاب را حروفچینی می‌کند.

در واقع LaTeX نقش طراح کتاب را ایفا می‌کند و از TeX به‌عنوان مسؤول حروفچینی استفاده می‌کند. اما، در هرصورت، LaTeX یک برنامه‌ی رایانه‌ای است و نیاز به راهنمایی و اطلاعات بیشتر دارد. لازم است نویسنده این اطلاعات را با استفاده از فرمان‌های ویژه‌ای که در طول این کتاب با آن آشنا می‌شویم، به LaTeX اعلان کند.

۲.۳.۱ یک مثال ساده

در اینجا چگونگی کارکرد LaTeX را، با حروفچینی یک متن بسیار کوتاه، تجربه می‌کنیم. ویرایشگر مورد نظر خود را باز کنید و خطوط زیر را، همان‌گونه که نشان داده شده است، وارد کنید:

```
\documentclass{article}
\begin{document}
 This is my \emph{first} document prepared
 in \LaTeX.
\end{document}
```

در بکارگیری کراکتر \، که به آن *backslash* گویند، دقت کنید. این کراکتر با کراکتر /، که به آن *forward slash* گویند، متفاوت است.

فایل حاصل را، به‌طور مثال، با نام `myfile.tex` ذخیره کنید. در طول این کتاب به این نوع فایل، **فایل ورودی** LaTeX می‌گوییم. توجه کنید، لازم است فایل ورودی همواره دارای پسوند `tex` باشد. اجرا کردن برنامه و دیدن حاصل کار بستگی به سامانه‌ی راه‌انداز رایانه‌ی کاربر

[۹] ناشر

دارد. در *Microsoft Windows* با استفاده از *Command Prompt* فرمانِ زیر را، در شاخه‌ای که `myfile.tex` وجود دارد، وارد کنید:

working-folder`> latex myfile`

پس از وارد کردن فرمان بالا و اجرای کامل برنامه، حاصل کار با فرمان زیر قابل مشاهده است:

working-folder`> yap myfile`

با اجرای فرمان بالا پنجره‌ای باز می‌شود که حاصل کار به صورت زیر در آن نمایان است.[۱۰]

This is my *first* document prepared in LaTeX.

ویرایشگرهای فایل ورودی LaTeX، در واقع، نقش یک واسط گرافیکی بین کاربر و LaTeX ایفا می‌کنند . بسته به ویرایشگر انتخابی، اجرای LaTeX روی فایل ورودی متفاوت است. به‌طور مثال، در ویرایشگر معروف *WinEdt* کافی است برای اجرای LaTeX بر دکمه‌ی LaTeX و برای دیدن خروجی آن بر دکمه‌ی DVI اشاره کرد.

۳.۳.۱ فایل ورودیِ LaTeX

همان‌طور که ملاحظه شد، فایل ورودی LaTeX از نوع فایل‌های متنیِ ساده است که می‌توان آن را به‌وسیله‌ی یک ویرایشگر ساده پدید آورد. فایل ورودی LaTeX شامل متن اصلی نوشتار مورد نظر، همراه با فرمان‌های ویژه‌ای است که نحوه‌ی حروفچینی متن را مشخص می‌کند.

اکنون به فایل ورودی LaTeX در مثالِ قبل نگاهی دقیق‌تر می‌کنیم. اولین خط، یعنی `\documentclass{atricle}`، به LaTeX اعلان می‌کند نوشته‌ی مورد نظر از نوع مقاله است. اگر بخواهیم این نوشته در قالب کتاب حروفچینی شود، کافی است اولین خط به `\documentclass{book}` تغییر کند.

متنی که قرار است حروفچینی گردد باید بین `\begin{document}` و `\end{document}` قرار گیرد. در مثالِ ساده‌ی بالا، این متن فقط از یک خط تشکیل شده است. این متن را در فایل ورودی با حاصل کار در خروجی، مقایسه می‌کنیم. سه کلمه‌ی اول، همان‌گونه که وارد شده

[۱۰] قابی که متن حروفچینی شده درون آن محدود شده است، جزو خروجی نیست.

است، در خروجی و با ظاهر مطلوب دیده می‌شود. سپس حاصلِ {first}\emph در خروجی به صورت *first* آمده است. \emph یک **فرمان**[11] در LaTeX است و به‌طور کلی \emph{⟨*text*⟩} باعث می‌شود متن ⟨*text*⟩، که درون { } محدود شده است و به آن **آرگومان** فرمان \emph گوییم، به صورت تأکید شده در خروجی ظاهر گردد. در ادامه سه کلمه‌ی دیگر، همان گونه که در فایل ورودی آمده است، با ظاهر مطلوب در خروجی دیده می‌شود. و بالاخره حاصل \LaTeX در ورودی، چاپِ واژه‌ی LaTeX در خروجی است.

همان‌گونه که ملاحظه می‌شود، فرمان‌های LaTeX با کراکتر \ شروع می‌شود. نکته مهم این است که LaTeX در فهم این فرمان‌ها نسبت به حروف کوچک و بزرگ حساس است. به‌طور مثال، چنانچه \LaTeX به \latex تغییر کند، از جانب LaTeX پیام خطای

```
undefined control sequence
```

صادر می‌شود. درباره‌ی فرمان‌های LaTeX در بخشِ ۱.۶.۱، به‌طور مفصل، بحث می‌کنیم.

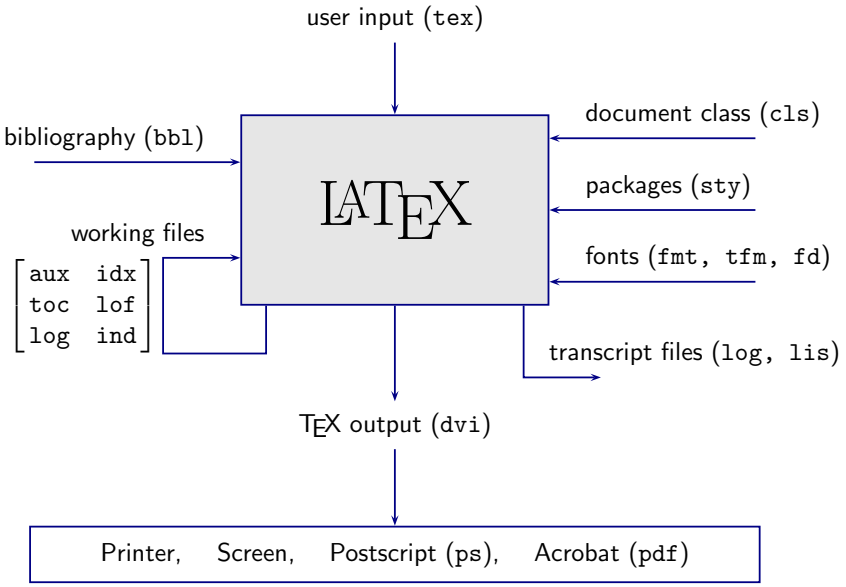
اکنون عبارتِ Today is \today را به انتهای خط اضافه نموده، پس از پردازش مجددِ فایل ورودی، حاصل کار را ببینید.

۴.۳.۱ فایل‌هایی که هنگام کار دخیل هستند

LaTeX هنگام کار از فایل‌های فراوانی استفاده می‌کند؛ فایل‌های گوناگونی را می‌خواند و در فایل‌های زیادی می‌نویسد. از بعضی فایل‌ها فقط برای خواندن اطلاعات مورد نیاز خود بهره می‌گیرد و از بسیاری از فایل‌ها برای نگهداری اطلاعاتی کمک می‌گیرد تا در موقع لزوم از آن استفاده کند. شکل ۱.۱ تا اندازه‌ای این فرایند را نمایش می‌دهد؛ این شکل نشان می‌دهد LaTeX هنگام کار معمولاً چه فایل‌هایی ایجاد می‌کند و از چه فایل‌هایی استفاده می‌کند.

همان‌گونه که در شکل ۱.۱ مشاهده می‌شود، برخی از فایل‌های مورد استفاده‌ی LaTeX فقط نقش ورودی دارند، مانند فایل‌های با پسوند tex، cls و sty، و برخی از این فایل‌ها فقط نقش خروجی دارند، مانند فایل‌های با پسوند dvi و log. اما دسته‌ای از فایل‌ها وجود دارد که هم خروجی LaTeX محسوب می‌شوند، یعنی LaTeX پس از هر اجرا آنها را به‌وجود می‌آورد، و هم

[11] Command



شکلِ ۱.۱: فایل‌هایی که هنگام کار دخیل هستند

به‌عنوان ورودیِ آن نقش دارند، یعنی LaTeX در اجراهای بعدی از اطلاعات موجود در آنها، برای کامل کردن کار خود، بهره می‌گیرد، مانند فایل‌های با پسوند aux و toc.

پس از اولین اجرای LaTeX روی فایل ورودی myfile.tex، به شاخه‌ای که این فایل در آن نگهداری می‌شود، و ما به آن **شاخه‌ی کاری**[۱۲] گوییم، نگاه کنید. به‌طور مثال، چنانچه فایل myfile.toc مشاهده شود، احتمالاً هنوز کار LaTeX در حروفچینی متن به‌طور کامل انجام نشده است. اطلاعاتی در این فایل هست که LaTeX برای اتمام کار خود به آن نیاز دارد. بنابراین لازم است بار دیگر LaTeX روی فایل myfile.tex اجرا شود تا اطلاعات مورد نظر به خروجی نهایی منتقل گردد. حتی گاهی لازم است LaTeX تا سه مرتبه روی فایل ورودی اجرا شود تا نتیجه نهایی تمام و کمال بدست آید. در این‌باره، به‌طور مثال، بخش‌های ۵.۴، ۶.۴ و ۷.۴ را ببینید.

در ادامه، به‌طور خلاصه، به توضیح برخی از فایل‌های مورد استفاده‌ی LaTeX می‌پردازیم:

tex مهمترین فایل، هنگام کار با LaTeX، فایل ورودی آن است، که توسط کاربر ایجاد می‌شود.

[12] Working folder

نام این فایل، به‌طور دلخواه، توسط کاربر تعیین می‌شود اما باید همواره دارای پسوند tex باشد. این فایل دربردارنده‌ی متن نوشته همراه با فرمان‌های ویژه‌ای است که چگونگی حروفچینی متن را به LaTeX اعلان می‌کند. در طول این کتاب به این فایل همواره **فایل ورودی** گوییم. این فایل ورودی از نوع فایل‌های متنی ساده است و آن را می‌توان با یک ویرایشگر متنی ساده پدید آورد.

cls فایلی است که دربردارنده‌ی اطلاعاتی درباره‌ی ساختار منطقی نوشتار و قالب ظاهری آن است و تعیین‌کننده‌ی رده‌ی نوشتاری است. این‌گونه فایل‌ها در شاخه‌ها و مسیرهای معینی نگهداری می‌شود به‌طوری که LaTeX، به‌آسانی، به آنها دسترسی دارد. این فایل‌ها با فرمان \documentclass فراخوانی می‌شود، مانند \documentclass{report}.

در LaTeX استاندارد پنج رده‌ی نوشتاری وجود دارد: ۱) رده‌ی نوشتاری مقاله با فایل article.cls، ۲) رده‌ی نوشتاری گزارش با فایل report.cls، ۳) رده‌ی نوشتاری کتاب با فایل book.cls، ۴) رده‌ی نوشتاری نامه با فایل letter.cls و ۵) رده‌ی نوشتاری اسلاید با فایل slides.cls. برای اطلاعات بیشتر درباره‌ی رده‌های نوشتاری بخش ۱.۴ در صفحه‌ی ۲۵۳ را ببینید.

علاوه بر رده‌های نوشتاری موجود در LaTeX استاندارد، رده‌های نوشتاری دیگری، توسط سازمان‌ها و اشخاص گوناگون، پدید آمده و به LaTeX اضافه شده است که، به‌طور مثال، می‌توان از

amsart, amsbook, memoir, prosper, beamer

نام برد.

sty بسته‌های جانبی LaTeX، که توسط اشخاص مختلف و سازمان‌های گوناگون فراهم شده است، در فایل‌های با پسوند sty ذخیره می‌شود. در این فایل‌ها فرمان‌ها و محیط‌های جدیدی تعریف شده است و به کاربر کمک می‌کند به شکل آسانتری توانایی‌های LaTeX را در اختیار گیرد. این فایل‌ها در شاخه‌ها و مسیرهای مشخصی نگهداری می‌شوند، به‌طوری که LaTeX، به‌آسانی، به آنها دسترسی دارد. این فایل‌ها با فرمان \usepackage فراخوانی می‌شود، مانند \usepackage{amsmath}.

به‌عنوان مثال، بسته‌ی amsmath توانایی LaTeX در حروفچینی عبارت‌های ریاضی را افزایش می‌دهد. بسته‌ی makeidx برای ایجاد نمایه و بسته‌ی graphicx برای گنجاندن تصاویر گرافیکی در نوشتجات LaTeX به‌کار می‌رود.

`tfm, fmt, fd` این دسته از فایل‌ها دربردارنده‌ی اطلاعاتی درباره‌ی نوعِ قلم و شکلِ قلم هستند.

در ادامه فرض می‌شود فایل ورودی LaTeX با نام `myfile.tex` ذخیره شده است. پس از هربار اجرای LaTeX بر این فایل ورودی، ممکن است فایل‌های گوناگونی پدید آید.

`dvi` مهمترین خروجی حاصل از اجرای LaTeX روی فایل ورودی `myfile.tex` فایل خروجی `myfile.dvi` می‌باشد که نمایش‌دهنده‌ی نوشتار حروفچینی شده توسط LaTeX است. این فایل را می‌توان توسط یک نمایش‌دهنده‌ی DVI، مانند *Yap* در *Microsoft Windows* و *xdvi* در *Linux*، مشاهده کرد. به‌علاوه، با فرمان

working-folder> `dvips myfile.dvi`

فایل `myfile.ps` در قالب *PostScript* بدست می‌آید که در *Microsoft Windows* توسط برنامه‌ی *GSView* قابل مشاهده است.

`log` در هربار اجرای LaTeX بر فایل ورودی `myfile.tex` گزارشی از آنچه رخ می‌دهد در صفحه‌ی نمایش از جلوی دیدگان کاربر عبور می‌کند و برای آنکه کاربر به آن دسترسی داشته باشد، این گزارش در فایل `myfile.log` ذخیره می‌شود. اطلاعاتی که در این فایل ذخیره می‌شود شامل فهرستی از فایل‌های خوانده شده، فهرستی از فایل‌های ایجاد شده، پیغام‌های هشدار دهنده، پیغام‌های خطا و اطلاعات مفید دیگر است.

فایل‌هایی که در ادامه به توضیح آن می‌پردازیم، بسته به آنکه کاربر چه فرمان‌هایی در فایل ورودی `myfile.tex` صادر کرده است و وی از LaTeX چه می‌خواهد، در هربار اجرای LaTeX بر فایل ورودی `myfile.tex`، پدید می‌آید. این فایل‌ها دربردارنده‌ی اطلاعاتی است که توسط LaTeX گردآوری و ذخیره شده است تا در اجراهای بعدی از آن استفاده کند. این فایل‌ها معمولاً برای شخص کاربر استفاده‌ای ندارد.

toc چنانچه فرمان \tableofcontents، که باعث گنجاندن فهرست مطالب می‌گردد، توسط کاربر در فایل ورودی صادر شده باشد، LaTeX در اولین اجرا فایلی با پسوند toc ایجاد می‌کند که دربردارنده‌ی عنوان فصل‌ها، بخش‌ها و زیربخش‌ها همراه با شماره‌ی صفحه‌ی متناظر است. LaTeX در اجراهای بعدی از این فایل کمک می‌گیرد تا فهرست مطالب را در خروجی نهایی بگنجاند. بنابراین، پس از هر تغییری در فایل ورودی که منجر به تغییر در فهرستِ مطالب گردد، برای آنکه این تغییرات در خروجی منعکس گردد، لازم است فایل ورودی حداقل دو بار توسط LaTeX پردازش گردد.

lof همانند toc است اما برای فهرست شکل‌ها. این فایل در صورتی ایجاد می‌شود که فرمان \listoffigures در فایل ورودی صادر شده باشد.

lot همانند toc و lof است اما برای فهرست جدول‌ها. این فایل در صورتی ایجاد می‌شود که فرمان \listoftables در فایل ورودی صادر شده باشد.

idx اگر نوشتار شامل نمایه باشد و در فایل ورودی فرمان‌های ویژه برای پدید آوردن نمایه، مانند \makeindex، صادر شده باشد، در اولین اجرای LaTeX روی فایل ورودی واژه‌های مشخص شده برای درج در نمایه در فایلی با پسوند idx ذخیره می‌شود. برای آنکه LaTeX از این اطلاعات در گنجاندن نمایه بهره گیرد، لازم است ابتدا این فایل توسط برنامه‌ی *MakeIndex*، [۱۲]، پردازش گردد. این برنامه فایل idx را خوانده واژه‌ها را مرتب می‌کند و حاصل کار را در فایلی با پسوند ind ذخیره می‌کند. اکنون فایل ind قابل استفاده توسط LaTeX است.

ind حاصل پردازش فایل idx، گزینه‌ی قبل، توسط برنامه‌ی *MakeIndex* فایلی با پسوند ind است. LaTeX از این فایل برای گنجاندن نمایه استفاده می‌کند.

aux در بخش ۹.۱ در صفحه‌ی ۶۹ خواهیم دید چگونه با به‌کارگیری سامانه‌ی برچسب‌گذاری و ارجاع متقابل در LaTeX، می‌توان عملِ ارجاع دادن به فصل‌ها، بخش‌ها، معادله‌های شماره‌گذاری شده و مانند آن را، به‌آسانی، انجام داد. انتقال اطلاعاتِ مربوط به ارجاع‌دهی، از پردازش قبلی به پردازش بعدی، بر عهده‌ی فایل کمکی با پسوند aux می‌باشد.

۵.۳.۱ چرا LaTeX؟

شیوه‌ی کار در LaTeX با پردازشگرهای متنی پیشرفته مانند *Microsoft Word* و *Corel WordPerfect* کاملاً متفاوت است. به این گونه پردازش‌گرها لفظ WYSIWYG اطلاق می‌شود که از جمله‌ی زیر گرفته شده است:

What You See Is What You Get.

هنگام کار با این گونه برنامه‌ها، کاربر ضمن حروفچینی متن شخصاً کار صفحه‌آرایی و تنظیمات مربوط به جایگذاری متن را انجام می‌دهد و این عمل به صورت تأثیر متقابل[۱۳] با برنامه انجام می‌شود، که این خود باعث بروز خطای انسانی می‌گردد. در این حالت، شخص قادر است ضمن کار نتیجه نهایی را در صفحه‌ی نمایشگرِ رایانه‌ی خود ببیند.

هنگام کار با LaTeX، از آنجا که کار حروفچینی و صفحه‌آرایی به LaTeX سپرده شده است و کاربر فقط متن و فرمان‌ها را وارد فایل ورودی می‌کند، به‌طور معمول، دیدنِ نتیجه نهایی و عملِ حروفچینی، به‌طور هم‌زمان، امکان‌پذیر نیست. پس از پردازش فایلِ ورودی LaTeX، خروجی نهایی را می‌توان توسطِ برنامه‌ی نمایش‌دهنده‌ی مخصوص مشاهده کرد و می‌توان نوشته را، پیش از فرستادن به چاپگر، اصلاح کرد. اکنون این پرسش مطرح است که

> چرا این همه دردسر؟ چرا به‌آسانی از یک پردازشگر متنی ساده یا پیشرفته از گونه‌ی WYSIWYG استفاده نکنیم؟

پاسخ این پرسش، و در واقع یکی از پاسخ‌های آن، در انگیزه‌ی ایجاد TeX نهفته است. دونالد کنوث می‌گوید هدف از پدید آوردنِ TeX دستیابی به یک برنامه‌ی رایانه‌ای است که به‌آسانی به‌توان متن‌هایی را که دربردارنده‌ی فرمول‌ها و عبارت‌های ریاضی هستند، به‌طور زیبا و ماهرانه، حروفچینی کرد. اغلب مشکل و گاهی غیر ممکن است به‌توان فرمول‌ها و عبارت‌های ریاضی پیچیده را با استفاده از پردازشگرهای متنی رایج پدید آورد. حتی برای یک متن ساده و بدون عبارت ریاضی، اگر قصد دارید حاصل کار زیبا و رضایت‌بخش باشد، LaTeX یکی از بهترین انتخاب‌ها است. به‌علاوه، LaTeX دارای ویژگی‌هایی است که در اینجا برخی از آنها را برمی‌شماریم:

[13] Interactively

۱ـ LaTeX رایگان است. این ویژگی برای کسانی که امکان دسترسی به نرم‌افزارهای تجاری ندارند و به رعایت حقوق مؤلف و ناشر اهمیت می‌دهند، بسیار برجسته است.

۲ـ کار حروفچینی و صفحه‌آرایی به LaTeX سپرده شده است و از این‌رو خطای انسانی به حداقل می‌رسد. این ویژگی موجب می‌شود نویسنده بیشتر روی محتوای نوشته‌ی خود و ساختار منطقی آن تمرکز کند و وقت خود را صرف پرداختن به قالب ظاهری آن نکند.

۳ـ رده‌های نوشتاری از پیش طراحی شده با ساختار منطقی و شکل ظاهری استاندارد و حرفه‌ای در اختیار کاربر قرار می‌گیرد.

۴ـ فرمول‌های پیچیده‌ی ریاضی به‌آسانی قابل حروفچینی است. کافی است ابتدا کاربر اندکی قواعد فرمول‌نویسی را فرا گیرد تا با استفاده از آن هر فرمول ریاضی پیچیده را به‌سادگی حروفچینی کند.

۵ـ ساختارهای پیچیده‌ای چون فهرست مطالب، پاورقی، کتابنامه و نمایه، به‌آسانی، با صادر کردن چند فرمان ساده، توسط LaTeX پدید می‌آید.

۶ـ بسته‌های جانبی فراوانی، که باعث بالا بردن توانایی کاربر در بکارگیری LaTeX می‌شود، پدید آمده است و به‌طور رایگان در اختیار کاربر قرار دارد. به‌علاوه، این بسته‌ها همواره به‌روز[۱۴] می‌شود.

۷ـ LaTeX از قابلیت حمل‌پذیری فراوانی برخوردار است، به‌طوری که تقریباً روی همه‌ی سامانه‌های راه‌انداز رایانه‌ای رایج قابل اجرا است و نتیجه‌ی کار در همه‌ی آنها یکسان است.

پس از مطالعه‌ی این کتاب و آگاهی از توانایی‌های LaTeX، به پاسخِ کاملِ پرسشِ مطرح شده در این قسمت خواهیم رسید.

[14] Up to date

۴.۱ فضاهای خالی

ابتدا به مثال زیر توجه کنید.

```
\LaTeX{} has several pre-defined styles for the layout of
typeset documents. Authors using \LaTeX{} sometimes
wish to understand how these layouts are parameterized.
The layouts package enables the display of certain of these parameterized
layouts, showing what the parameters control.
```

LaTeX has several pre-defined styles for the layout of typeset documents. Authors using LaTeX sometimes wish to understand how these layouts are parameterized. The layouts package enables the display of certain of these parameterized layouts, showing what the parameters control.

ملاحظه می‌شود، هرچند ممکن است در فایلِ ورودی خطوط در یک پاراگراف با طول متفاوت باشد، اما در فایلِ خروجی خطوط در یک پاراگراف همگی دارای طول یکسان هستند و از حاشیه سمت چپ و راست به یک فاصله تنظیم شده است. این عمل توسط LaTeX با تنظیم فاصله‌های بین کلمات انجام می‌شود.

LaTeX کراکترهای جاخالی مانند TAB و BLANK را به‌عنوان یک فضای خالی واحد در نظر می‌گیرد و فضاهای خالی متعدد پشت سر هم مانند یک فضای خالی است. فضاهای خالی در ابتدای هر خط در نظر گرفته نمی‌شود و رفتن از یک خط به خط بعدی، که این عمل معمولاً با وارد کردن کلید ENTER انجام می‌شود، به‌عنوان یک فضای خالی تعبیر می‌شود. یک خط خالی یا بیشتر، بین محتویات فایل ورودی LaTeX، مشخص‌کننده‌ی پایان پاراگراف جاری و شروع پاراگراف بعدی است. چند خط خالی هم‌ارز با یک خط خالی است.

```
It does not matter wether you
enter one or several      spaces
after a word.


An empty line, or more, starts
a new paragraph.
```

It does not matter wether you enter one or several spaces after a word.
An empty line, or more, starts a new paragraph.

به‌طور متعارف، در نوشتجات حروفچینی شده، فضایِ خالی بین دو جمله‌ی متوالی از فضای خالی بین دو کلمه‌ی واقع در یک جمله، که ما به آن فضای خالی **بین‌کلمه‌ای** گوییم، اندکی

بیشتر می‌باشد. این امر به خوانایی نوشته کمک می‌کند. LaTeX نیز از این قاعده پیروی می‌کند. اما LaTeX چگونه پایان جمله را تشخیص می‌دهد؟

به‌طور پیش‌فرض، اگر LaTeX به یکی از نشانه‌های نقطه‌گذاری

.	?	!

که بلافاصله پس از یک حرف کوچک[۱۵] قرار داشته باشد، برخورد کند، آن را پایان جمله تلقی می‌کند. این به‌خاطر آن است که اغلب جملات به یک حرف کوچک و یک نشانه‌ی نقطه‌گذاری ختم می‌شوند و اگر حرف بزرگی به یک نقطه پایان یابد معمولاً نشانه‌ی مختصرنویسی است و پایان جمله به‌حساب نمی‌آید. اما، به‌طور آشکار، این قاعده همواره درست نیست. به مثال زیر نگاه کنید:

```
Carrots are good for your eyes, since they contain Vitamin A. Have
you ever seen a rabbit wearing glasses?
```

Carrots are good for your eyes, since they contain Vitamin A. Have you ever seen a rabbit wearing glasses?

در مثال بالا نقطه‌ی پس از حرف "A" نقطه‌ی پایانِ جمله است، اما LaTeX قادر به تشخیص آن نیست و بنابراین فضای خالی پس از آن به‌خوبی رعایت نشده است. رفع این مشکل باید توسط کاربر و به‌طور دستی انجام شود. برای این کار فرمان \@، درست پیش از نقطه‌ی مورد نظر، بکار گرفته می‌شود. فرمان \@، که درست پیش از یکی از علایم نقطه‌گذاری صادر گردد، باعث می‌شود LaTeX آن را نشانه‌ی پایانِ جمله در نظر بگیرد. مثال زیر را با مثال قبل مقایسه کنید. آیا فضای خالی بیشتر بین دو جمله، نسبت به مثال قبل، قابل تشخیص است.

```
Carrots are good for your eyes, since they contain Vitamin A\@. Have
you ever seen a rabbit wearing glasses?
```

Carrots are good for your eyes, since they contain Vitamin A. Have you ever seen a rabbit wearing glasses?

از طرفِ دیگر، مواردی هست که هرچند یک حرف کوچک به یک نقطه ختم می‌شود اما آنجا پایان جمله نیست. به مثال زیر نگاه کنید:

[۱۵] منظور حروف کوچک در الفبای انگلیسی است، یعنی a,b,...,z

```
The numbers 1, 2, 3 etc. are called natural numbers. According
to Kroncker, they were made by God; all else being the work of man.
```

The numbers 1, 2, 3 etc. are called natural numbers. According to Kroncker, they were made by God; all else being the work of man.

LaTeX به اشتباه "etc." را پایان جمله در نظر گرفته است و فضای خالی پس از آن اندکی بیشتر از فضای خالی بین‌ـکلمه‌ای است. برای جلوگیری از این اشتباه از کراکتر \ همراه با یک یا چند فضای خالی پس از آن، بلافاصله بعد از "etc." استفاده می‌کنیم.

```
The numbers 1, 2, 3 etc.\ are called natural numbers. According
to Kroncker, they were made by God; all else being the work of man.
```

The numbers 1, 2, 3 etc. are called natural numbers. According to Kroncker, they were made by God; all else being the work of man.

در حالت کلی از کراکتر \ همراه با یک یا چند فضای خالی پس از آن، برای پدید آوردن یک فضای خالی بین‌ـکلمه‌ای استفاده می‌شود.

یکی دیگر از راه‌های ایجاد فضای خالی بین‌ـکلمه‌ای، وارد کردن کراکتر ~ است. تفاوت عملکرد کراکتر ~ با کراکتر \ همراه با یک فضای خالی پس از آن، این است که کراکتر ~ فضای خالی غیر قابل شکست ایجاد می‌کند. به‌طور مثال، چنانچه عبارت Zorn Lemma در انتهای خط جاری واقع شود و فضای کافی برای گنجاندن هر دو واژه در این خط موجود نباشد، ممکن است واژه‌ی Zorn انتهای خط جاری و واژه‌ی Lemma ابتدای خط بعدی جای گیرد، که چندان پسندیده نیست. اما چنانچه در فایل ورودی به‌جای عبارت `Zorn Lemma` عبارت `Zorn~Lemma` وارد شود، ضمن پدید آمدن یک فضای خالی بین دو واژه‌ی Zorn و Lemma، این دو کلمه از هم جداشدنی نیستند و بنابراین همواره عبارت Zorn Lemma در یک خط جای می‌گیرد. برای اطلاعات بیشتر درباره فرمان‌هایی که پدیدآور فضای خالی هستند، بخش ۲.۲ در صفحه‌ی ۸۳ را ببینید.

حال به نمونه‌ای دیگر از رفتار LaTeX با فضاهای خالی می‌پردازیم. ابتدا به مثال زیر نگاه کنید:

```
I think \LaTeX is fun.
```

I think LaTeXis fun.

چه بر سر فضای خالی پس از \LaTeX آمده است؟ مثل اینکه LaTeX تمام فضای خالی پس از آن را بلعیده است! در حالت کلی LaTeX فضای خالی پس از یک فرمان، که با یک کراکتر حرفی پایان می‌یابد، را حذف می‌کند و در نظر نمی‌گیرد. در اینجا بار دیگر \ به کمک می‌آید.

```
I think \LaTeX\ is fun.
```

I think LaTeX is fun.

راه حل دیگری برای این مشکل وجود دارد. می‌توان بلافاصله پس از فرمانی که با کراکتر حرفی پایان می‌یابد رشته‌ی {} قرار داد و سپس یک یا چند فضای خالی برجای گذاشت.

```
I think \LaTeX{} is fun.
```

I think LaTeX is fun.

درباره‌ی فضاهای خالی در بخش ۲.۲ توضیحات بیشتر ارائه شده است.

۵.۱ کراکترهای مخصوص

همان‌گونه که در مثال‌های پیشین ملاحظه شد، حاصل \LaTeX در فایلِ ورودی ظاهر شدنِ واژه‌ی LaTeX در خروجی است. به‌علاوه وارد کردن \ همراه با یک فضای خالی پس از آن، باعث پدید آمدن یک فضایِ خالی بین‌ـکلمه‌ای در خروجی می‌شود.

در LaTeX از علامت \ برای منظور خاصی استفاده می‌شود؛ برای آنکه به LaTeX اعلان شود آنچه پس از \ می‌آید متنی برای حروفچینی نیست، بلکه فرمان و دستوری برای انجامِ کاری است. بنابراین با وارد کردن مستقیمِ کراکتر \ در فایلِ ورودی، نمی‌توان در خروجی همان را بدست آورد. پس اگر بخواهیم در خروجی "\" داشته باشیم، چه باید کرد؟ برای این کار از فرمان \textbackslash استفاده می‌کنیم. در فصل ۳ خواهیم دید، در عبارت‌های ریاضی، برای پدید آوردن "\" از فرمان \backslash استفاده می‌شود. اکنون به مثال زیر دقت کنید:

```
The command \verb|\textbackslash| produces ``\textbackslash''. Maybe,
I have now learnt about 1% of \LaTeX.
```

The command \textbackslash produces "\". Maybe, I have now learnt about 1

در مثال بالا چه بر سر باقیمانده‌ی خط پس از 1 آمده است؟ LaTeX علامت % و آنچه پس از آن و در همان خط وارد شده است، را نادیده گرفته است. در واقع علامت % به LaTeX اعلان می‌کند متن پس از آن جملاتی است که نباید در خروجی ظاهر شود.

می‌توان از علامت % برای وارد کردن جملات توضیحی[۱۶] در فایل ورودی بهره گرفت. برنامه‌نویسانِ LaTeX از این ویژگی برای ثبت نکاتی در برنامه‌ی خود، برای استفاده‌ی خود و دیگران هنگام مطالعه‌ی فایلِ ورودی، سود می‌برند.

پس برای بدست آوردن "%" در خروجی چه باید کرد؟ کافی است در فایل ورودی \% وارد کنیم:

```
Maybe, I have now learnt about 1\% of \LaTeX.
```

Maybe, I have now learnt about 1% of LaTeX.

علامت‌های \ و % تنها دو علامت از ده علامتِ خاص در LaTeX هستند. گردایه‌ی کامل علامت‌های مخصوص در LaTeX عبارت است از

^	#	$	%	~	&	_	\	{	}

این‌ها علامت‌هایی هستند که LaTeX از آن برای منظور خاصی استفاده می‌کند. وارد کردن آن در فایل ورودی، به‌طور مستقیم، باعث پدید آمدن آن در خروجی نمی‌شود بلکه باعث انجام کاری، احتمالاً، ناخواسته یا ایجاد خطا می‌گردد. در جدولِ ۱.۱ فرمان‌های لازم، برای پدید آوردن این علامت‌ها در خروجی، گردآوری شده است.

غیر از سه علامت ~ و ^ و \، سایر علامت‌های مخصوص با جایگذاری کراکتر \، درست قبل از علامت مورد نظر، در خروجی ایجاد می‌شود. برای عملکرد ~\ و ^\ به جدول الف.۱۸ در صفحه‌ی ۳۹۱ نگاه کنید. برای عملکرد \\ به مثال زیر دقت کنید.

```
This is the first
line.\\ This is the second line.
```

This is the first line.
This is the second line.

[16] Comments

Character	Input	Output
^	\textasciicircum	^
#	\#	#
$	\$	$
%	\%	%
~	\textasciitilde	~
&	\&	&
_	_	_
\	\textbackslash	\
{	\{	{
}	\}	}

جدولِ ۱.۱: کراکترهای مخصوص

در LaTeX می‌توان از رشته‌ی \\ برای پایان دادن به خطِ جاری و آغاز خطِ جدید کمک گرفت. صورتِ کلی این فرمان به شکل \\[⟨*height*⟩] است که در آن ⟨*height*⟩ گزینه‌ی اختیاری آن محسوب می‌شود و برای تنظیم فاصله‌ی بین خط جاری و خط بعدی بکار می‌رود. ⟨*height*⟩ مقداری عددی برحسب یکی از واحدهای طولی قابل قبول در LaTeX، مانند cm یا pt می‌باشد، در این‌باره بخش ۶.۶.۱ و جدول ۴.۱ در صفحه‌ی ۴۴، را ببینید. چنانچه گزینه‌ی اختیاری ⟨*height*⟩ وارد نشود فاصله‌ی خط بعدی تا خط جاری به همان اندازه‌ی فاصله‌ی بین خطوط در حالت عادی است. چنانچه مقدار ⟨*height*⟩ منفی باشد خط بعدی به خط جاری نزدیک می‌شود و چنانچه مقدار ⟨*height*⟩ مثبت باشد خط بعدی از خط جاری دور می‌گردد. بهتر است مقدار ⟨*height*⟩ برحسب سایر متغیرهای طولی تعریف شده در LaTeX مانند \baselineskip بیان شود. این کار به حمل‌پذیری[17] نوشتار کمک می‌کند. به مثال زیر نگاه کنید.

[17] Portability

```
This is the first line.\\
This is the second line.\\[10pt]
This is the third
line.\\[-.3\baselineskip]
This is the last line.
```

This is the first line.
This is the second line.

This is the third line.
This is the last line.

دربارهٔ فرمان \\ در بخش ۱.۱.۲ توضیحات بیشتر ارائه شده است. برای اطلاعات بیشتر، دربارهٔ طول‌ها و متغیرهای طولی، بخش ۶.۶.۱ را ببینید.

۶.۱ برخی ساختارهای اساسی

در این بخش به بررسی برخی از ساختارهای اساسی LaTeX شامل فرمان‌ها، محیط‌ها، شمارنده‌ها، متغیرهای طولی و جعبه‌ها می‌پردازیم. به‌علاوه، می‌آموزیم چگونه می‌توان فرمان جدید و محیط جدید تعریف کرد.

۱.۶.۱ فرمان‌ها

تمام فرمان‌ها در LaTeX با کراکتر \ شروع می‌شود مانند \tableofcontents، \emph{first} و \&. در واقع کراکتر \ به LaTeX اعلان می‌کند آنچه بلافاصله پس از آن می‌آید متنی برای حروفچینی نیست بلکه فرمان و دستوری برای انجام کاری است. به‌طور مثال، فرمان \tableofcontents باعث می‌شود فهرستی از مطالب حروفچینی شده، در محل صدور این فرمان، درج گردد. کاربر لازم نیست هیچ کار دیگری انجام دهد؛ اگر به‌خوبی از قواعد مربوط به فصل‌بندی و بخش‌بندی استفاده کرده باشد، فرمان \tableofcontents، به‌طور خودکار، همه‌ی عنوان‌ها و شماره‌ها را درج می‌کند. در این زمینه بخش‌های ۴.۴ و ۵.۴ را ببینید.

نکته‌ی مهم آن است که LaTeX هنگام خواندن فرمان‌ها، نسبت به حروف کوچک و بزرگ حساس است. به‌طور نمونه، فرمان \LaTeX تعریف شده است و باعث پدید آمدن واژه‌ی LaTeX در خروجی می‌شود، درحالی که فرمان \latex ناشناخته است و وارد کردن آن در فایل ورودی باعث صدور پیغام خطا از جانب LaTeX می‌گردد.

فرمان‌های LaTeX، به‌طور کلی، به یکی از دو صورت زیر ظاهر می‌شوند:

- با کراکتر \ شروع می‌شود سپس با یک نام متشکل از کراکترهای حرفی ادامه می‌یابد.[۱۸] این فرمان‌ها با یک فضای خالی، یک عدد یا یک کراکتر غیرحرفی به پایان می‌رسد، مانند \bfseries و \emph{first}.
- با کراکتر \ شروع می‌شود سپس با یک کراکتر غیر حرفی به پایان می‌رسد، مانند \& و \\.

فرمان‌ها در LaTeX به دو دسته تقسیم می‌شوند:

۱ـ **فرمان‌های بدون آرگومان.** این فرمان‌ها ساده و یک‌کلمه‌ای و بدون آرگومان هستند، مانند \bfseries و \itshape. این فرمان‌ها معمولاً با یک کراکتر حرفی به پایان می‌رسند و لازم است با یک یا چند فضای خالی یا با یک کراکتر غیرحرفی از متن پس از آن جدا شوند. به‌طور مثال تمام حالت‌های زیر درست است:

```
\bfseries Thanks to Aunt Mabel for all her help.
```

```
\bfseries{}Thanks to Aunt Mabel for all her help.
```

```
\bfseries
   Thanks to Aunt Mabel for all her help.
```

نکته‌ی مهم آن است که LaTeX تمام فضای خالی موجود پس از فرمانی که با یک کراکتر حرفی به پایان می‌رسد را نادیده می‌گیرد. این مطلب، درباره‌ی فرمان‌هایی که با یک کراکتر غیرحرفی به پایان می‌رسد، درست نیست. اگر بخواهیم فضای خالی پس از یک فرمانِ پایان یافته به یک کراکتر حرفی، در خروجی حفظ شود می‌توان پس از آن رشته‌ی {} قرار داد و سپس یک یا چند فضای خالی برجای گذاشت و یا از فرمان‌های ویژه‌ای که تولید فضای خالی می‌کنند، استفاده کرد.

```
\LaTeX is fun.
\LaTeX{} is fun. \LaTeX\ is fun.
```

LaTeXis fun. LaTeX is fun. LaTeX is fun.

۲ـ **فرمان‌های دارای آرگومان.** بسیاری از فرمان‌ها در LaTeX نیاز به یک یا چند آرگومان دارند. ابتدا شکل زیر را ببینید.

[۱۸] منظور از کراکترهای حرفی، کراکترهای {a,A,b,B,...,z,Z} می‌باشد.

```
\documentclass[12pt,a4paper]{article}
\usepackage[leqno]{amsmath}
\usepackage{pstricks,graphicx}
\begin{document}
  \chapter{Banach Algebras} \label{Banach-Algebras}
    A \textbf{Banach Algebra} is ...
\end{document}
```

تمام فرمان‌ها در شکل بالا دارای یک یا چند آرگومان هستند که یا درون [ ] و یا درون { } جای گرفته است. ممکن است یک فرمان در LaTeX دارای دو نوع آرگومان باشد:

الف) **آرگومان اجباری.**[۱۹] این نوع آرگومان درون { } قرار می‌گیرد و پس از فرمان وارد می‌شود؛ مانند \vspace{1cm} یا \usepackage{amsmath}. وارد نکردن این نوع آرگومان، برای فرمانی که نیاز به آن دارد، باعث صدور پیغام خطا از جانب LaTeX می‌شود.

ب) **آرگومان اختیاری.**[۲۰] این نوع آرگومان درون [ ] جای می‌گیرد و معمولاً بلافاصله پس از فرمان مورد نظر و قبل از { } قرار می‌گیرد. وارد کردن یا وارد نکردن این نوع آرگومان به خواست کاربر بستگی دارد؛ مانند \documentclass[12pt,twocolumn]{book} یا \usepackage[leqno]{amsmath} و یا \\[2cm].

۲.۶.۱ تعریف فرمان جدید

در LaTeX هزاران فرمان مختلف، برای انجام کارهای گوناگون، فراهم شده است و در اختیار کاربر قرار گرفته است، هرچند اغلب کاربران از وجود بسیاری از این فرمان‌ها بی‌خبر هستند. با این وجود، همواره امکان دارد کاربر از LaTeX کاری بخواهد که هیچ یک از فرمان‌های از پیش تعریف شده در LaTeX آن را برآورده نکند. خوشبختانه در LaTeX امکان تعریف کردن فرمان جدید توسط کاربر فراهم شده است.

[19] Mandatory argument

[20] Optional argument

مطلب را با یک مثال ادامه می‌دهیم. فرض کنیم نویسنده‌ای قصد دارد نوشته‌ی خود را با LaTeX حروفچینی کند و بنابراین مقدمات را برای ایجاد فایل ورودی و وارد نمودن متن نوشته در آن، فراهم می‌کند. این شخص می‌داند در نوشته‌اش عبارت "if and only if" فراوان به‌کار رفته است. وی به‌آسانی می‌تواند، در مکانی مناسب از فایل ورودی، فرمانی جدید تعریف کند تا با استفاده از آن، هربار از نوشتن عبارت مذکور رهایی یابد.

در LaTeX تعریف فرمان جدید با کمک دستور \newcommand انجام می‌شود.

```
\newcommand{\Iff}{if and only if}
```

در این مثال نام \Iff برای فرمان جدید انتخاب شده است. در این رابطه، باید به موارد زیر دقت کرد.

۱ـ نامِ فرمانِ جدید همواره با کراکتر \ آغاز می‌شود و منحصراً از کراکترهای حرفی تشکیل می‌گردد.

۲ـ لازم است نام انتخاب شده برای فرمان جدید یکتا باشد، یعنی این نام از پیش برای LaTeX تعریف شده نباشد. در این مثال، نخستین گزینه‌ای که برای نام فرمان جدید مناسب به‌نظر می‌رسد، نامِ \iff است. مسأله این است که فرمان \iff از پیش برای LaTeX تعریف شده است و برای منظور دیگری به‌کار می‌رود و بنابراین نام \Iff را برای فرمان جدید برگزیده‌ایم.

۳ـ فرمان جدید از همان مکانی که تعریف شده است قابل استفاده است و پیش از آن برای LaTeX معنی‌دار نیست. به‌علاوه، فرمان جدید در همان گروه‌بندی که درون آن تعریف شده است معتبر است و قابل بکارگیری می‌باشد و خارج از آن برای LaTeX ناشناخته است. در اینجا، منظور از یک گروه‌بندی محدود کردن متن، اشیاء و فرمان‌ها درون { } یا درون یک محیط \begin{⟨*env-name*⟩} ... \end{⟨*env-name*⟩} است. در این‌باره بخش ۸.۱ در صفحه‌ی ۶۷ را ببینید.

۴ـ چنانچه بخواهیم فرمان جدید به‌طور سراسری در فایل ورودی قابل استفاده باشد، بهترین مکان، برای تعریف فرمان جدید، مقدمه‌ی فایل ورودی، یعنی پس از فرمان \documentclass و پیش از فرمان \begin{document}، می‌باشد.

به مثال بالا بر می‌گردیم. با تعریف فرمان جدید، این شخص می‌تواند زین پس، به‌جای وارد کردن عبارت `if and only if`، از فرمانِ `\Iff` بهره گیرد.

```
\newcommand{\Iff}{if and only if}
A function $f:X\to Y$ is continuous \Iff{} $f^{-1}(U)$ is open in $X$,
for every open set $U\subset Y$.
```

A function $f : X \to Y$ is continuous if and only if $f^{-1}(U)$ is open in X, for every open set $U \subset Y$.

ملاحظه می‌شود، از آنجا که فرمان جدید `\Iff` به یک کراکتر حرفی ختم می‌شود، همان‌گونه که در بخش ۴.۱ بیان شد، برای آنکه LaTeX فضای خالی پس از آن را نادیده نگیرد، پایان آن رشته‌ی `{}` قرار داده‌ایم.

بسته‌ی xspace. بدون شک، این موضوع که LaTeX فضای خالی پس از فرمانی که به یک کراکتر حرفی ختم می‌شود، را نادیده می‌انگارد یک موضوع آزاردهنده هنگام کار با LaTeX است. خوشبختانه این مشکل، برای فرمان‌هایی که توسط کاربر تعریف می‌شود، به‌وسیله‌ی بسته‌ی xspace، [۹]، قابل حل است؛ به‌این ترتیب که ابتدا در مقدمه‌ی فایل ورودی بسته‌ی xspace را با فرمان `\usepackage{xspace}` فراخوانی کرده سپس، هنگام تعریفِ فرمانِ جدید، دستور `\xspace` را به پایان آن اضافه می‌کنیم. مثال زیر این موضوع را به‌خوبی روشن می‌کند.

```
% in preamble: \usepackage{xspace}
\newcommand{\USA}{United States of America\xspace}
\newcommand{\GB}{Great Britain\xspace}
The \USA has 50 states. The \USA, \GB and Canada have close cultural links.
```

The United States of America has 50 states. The United States of America, Great Britain and Canada have close cultural links.

تعریف فرمان جدیدی که دارای آرگومان است. به مثال اصلی این بخش برمی‌گردیم. فرض کنیم عبارت $(x_1, x_2, \ldots, x_n)$ نیز، به‌مراتب، در متن استفاده می‌شود. این یک عبارت ریاضی است که با فرمان `$(x_1,x_2,\ldots,x_n)$` پدید می‌آید. برای آنکه هربار نیاز به

وارد کردن این فرمان نسبتاً طولانی نباشد، کافی است ابتدا فرمان زیر، در مکان مناسب، به فایل ورودی اضافه گردد:

```
\newcommand{\vect}{$(x_1,x_2,\ldots,x_n)$}
```

بدین ترتیب با وارد کردنِ \vect در فایل ورودی، عبارتِ $(x_۱, x_۲, \ldots, x_n)$ در خروجی حاصل می‌شود.

```
\newcommand{\vect}{$(x_1,x_2,\ldots,x_n)$}
We often write $x$ to denote \vect.
```

We often write x to denote $(x_1, x_2, \ldots, x_n)$.

حال چنانچه عبارت‌هایی مانند $(y_۱, y_۲, \ldots, y_n)$ و $(z_۱, z_۲, \ldots, z_n)$ نیز در نوشته زیاد بکار می‌رود، آیا باید برای هر کدام فرمانی جداگانه تعریف کرد؟ این‌طور نیست، بلکه کافی است فرمان \vect را به‌طور مناسب تغییر داد:

```
\newcommand{\vect}[1]{$(#1_1,#1_2,\ldots,#1_n)$}
```

عدد 1 در [1] بیان‌گر آن است که فرمان \vect دارای یک آرگومان اجباری است که در مکانِ #1 قرار می‌گیرد.

```
\newcommand{\R}{\mathbb{R}}
\newcommand{\vect}[1]{$(#1_1,#1_2,\ldots,#1_n)$}
Let \vect{x} and \vect{y} be elements of $\R^n$. Then \ldots
```

Let $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ be elements of $\mathbb{R}^n$. Then …

و بالاخره فرض کنید نوشته‌ی مورد نظر، علاوه بر عبارت‌های $(x_۱, x_۲, \ldots, x_n)$، $(y_۱, y_۲, \ldots, y_n)$ و مانند آن، شامل عبارت‌هایی چون $(x_۱, x_۲, \ldots, x_m)$ و $(y_۱, y_۲, \ldots, y_p)$ هم هست. می‌توان فرمان \vect را به‌گونه‌ای تغییر داد که تمام این حالت‌ها را دربر گیرد:

```
\newcommand{\vect}[2]{$(#1_1,#1_2,\ldots,#1_#2)$}
```

اکنون فرمان \vect دارای دو آرگومان اجباری به صورت \vect{⟨*arg1*⟩}{⟨*arg2*⟩} است؛ آرگومان ⟨*arg1*⟩ در مکان #1 و آرگومان ⟨*arg2*⟩ در مکان #2 قرار می‌گیرد.

```
\newcommand{\vect}[2]{$(#1_1,#1_2,\ldots,#1_#2)$}
Two vectors \vect{x}{m} and \vect{y}{n} are equal if and only if $n=m$ and
$x_i=y_i$ for $1\leq i\leq n$.
```

Two vectors $(x_1, x_2, \ldots, x_m)$ and $(y_1, y_2, \ldots, y_n)$ are equal if and only if $n = m$ and $x_i = y_i$ for $1 \leq i \leq n$.

با استفاده از دستور \newcommand می‌توان فرمان‌های جدید با حداکثر ۹ آرگومان تعریف کرد.

تعریف فرمان جدیدی که دارای آرگومان اختیاری است. صورت کلی فرمان \newcommand به شکل زیر است.

```
\newcommand{⟨new-com-name⟩}[⟨number⟩][⟨default⟩]{⟨definition⟩}
```

در اینجا ⟨*new-com-name*⟩ نام فرمان جدید است و ⟨*number*⟩ عددی است در مجموعه‌ی $\{۱, ۲, \ldots, ۹\}$ که وارد کردن آن اختیاری است. عدد ⟨*number*⟩ مشخص‌کننده‌ی تعداد آرگومان‌های فرمان جدید است.

وارد کردن گزینه‌ی ⟨*default*⟩ هم اختیاری است. چنانچه گزینه‌ی ⟨*default*⟩ وارد گردد، LaTeX نخستین آرگومان فرمان جدید را به‌عنوان آرگومان اختیاری محسوب می‌کند به‌طوری که اگر، هنگام صادر کردن فرمان جدید، آرگومان اختیاری توسط کاربر وارد نشود، LaTeX به‌طور پیش‌فرض گزینه‌ی ⟨*default*⟩ را برای آن، منظور می‌کند. بنابراین، وارد کردن گزینه‌ی ⟨*default*⟩ هنگامی معنی‌دار است که پیش از آن گزینه‌ی ⟨*number*⟩ وارد شده باشد. به مثال زیر توجه کنید.

```
\newcommand{\vect}[1][x]{$(#1_1,#1_2,\ldots,#1_n)$}
    \vect \qquad \vect[y] \qquad \vect[z] \par
\renewcommand{\vect}[2][x]{$(#1_1,#1_2,\ldots,#1_#2)$}
    \vect{n} \qquad \vect[y]{m} \qquad \vect[z]{p}
```

$(x_1, x_2, \ldots, x_n)$ $(y_1, y_2, \ldots, y_n)$ $(z_1, z_2, \ldots, z_n)$
$(x_1, x_2, \ldots, x_n)$ $(y_1, y_2, \ldots, y_m)$ $(z_1, z_2, \ldots, z_p)$

فرمان \renewcommand. همان‌طور که در مثال بالا ملاحظه می‌شود، از دستور \renewcommand برای دوباره تعریف کردن فرمانی موجود و از پیش تعریف شده، استفاده می‌شود. صورت کلی این فرمان به شکل زیر است.

```
\renewcommand{⟨old-com-name⟩}[⟨number⟩][⟨default⟩]{⟨definition⟩}
```

همه چیز مانند فرمان \newcommand است جز آنکه در اینجا ⟨*old-com-name*⟩ نام یک فرمان از پیش تعریف شده می‌باشد.

در مثال‌های این بخش، فرمان جدید \vect به‌گونه‌ای تعریف شده است که فقط در حالت متنی قابل استفاده است. تعریف فرمان جدید که در حالت ریاضی کارایی داشته باشد نیز امکان پذیر است. حتی می‌توان فرمان جدید را طوری تعریف کرد که در هر دو حالت متنی و ریاضی قابل بکارگیری باشد. برای جزئیات بیشتر بخش ۶.۳ در صفحه‌ی ۱۸۴ را ببینید.

۳.۶.۱ محیط‌ها

در LaTeX بسیاری از فرمان‌ها قسمتی از متن را به‌عنوان آرگومان دریافت می‌کنند و عملی روی آن انجام می‌دهند، مانند \textit{⟨*text*⟩} که قلمِ *italic* را به متن ⟨*text*⟩ اعمال می‌کند. حال اگر بخواهیم LaTeX روی یک متن طولانی، که در یک یا چند پاراگراف جای گرفته است، عملی انجام دهد، چندان پسندیده نیست اگر آن متن طولانی، به‌عنوان آرگومان یک فرمان، درون { } محدود گردد؛ این کار علاوه بر مشکلات فنی، از خوانایی فایلِ ورودی می‌کاهد. از این‌رو، در LaTeX ساختار دیگری با نام **محیط**[۲۱] فراهم شده است. یک محیط با فرمان \begin{⟨*env-name*⟩} آغاز می‌شود و با فرمان \end{⟨*env-name*⟩} پایان می‌یابد، که در اینجا ⟨*env-name*⟩ نام محیط است و تنها از کاراکترهای حرفی {a,A,b,B,...,z,Z} تشکیل می‌شود.

محیط‌ها حالت خاصی از فرمان‌ها هستند. به‌طور مثال اگر فرمان \textit{⟨*text*⟩} باعث اعمال قلم *italic* به متن ⟨*text*⟩ می‌گردد، محیطِ itshape به‌شکل

[21] Environment

```
\begin{itshape}
  ⟨text⟩
\end{itshape}
```

نیز باعث اعمال قلم *italic* به متن ⟨*text*⟩ می‌شود. هرچند عملکرد هردو یکسان است، اما طبیعی است در صورتی که متن ⟨*text*⟩ یک جمله‌ی کوتاه است از فرمان \textit و آنجا که ⟨*text*⟩ یک متن طولانی است از محیط itshape استفاده کنیم.

اولین محیطی که، هنگام کار با LaTeX، با آن روبرو می‌شویم، مهمترین و اصلی‌ترین محیط در LaTeX، یعنی محیط document است. آنچه قرار است توسط LaTeX حروفچینی شود باید در محیط document، یعنی پس از فرمان \begin{document} و پیش از فرمان \end{document}، قرار گیرد. اگر متنی پس از فرمان \end{document} وارد شود توسط LaTeX نادیده گرفته می‌شود و اگر متنی پیش از فرمان \begin{document} وارد گردد از جانب LaTeX پیغام خطای

```
Missing \begin{document}
```

صادر می‌شود.

در اینجا گردایه‌ای از مهمترین محیط‌های از پیش تعریف شده در LaTeX استاندارد جمع‌آوری شده است.

abstract	array	center	description	displaymath	document
enumerate	equation	figure	flushleft	flushright	itemize
letter	math	minipage	quotation	quote	table
tabular	thebibliography	titlepage	verbatim	verse	

در ادامه به توضیح مختصری درباره‌ی هریک از این محیط‌ها می‌پردازیم.

abstract این محیط در رده‌های نوشتاری article و report تعریف شده است. همان‌طور که از نام آن پیدا است، برای حروفچینی چکیده‌ی نوشتار به‌کار می‌رود؛ بدین معنی که چکیده‌ی نوشتار در محیط abstract جای می‌گیرد. در این‌باره بخش ۳.۴ در صفحه‌ی ۲۶۱ را ببینید.

array این محیط منحصراً در حالت ریاضی قابل استفاده است و برای چیدن اشیاء و علایم

در آرایشی با چند سطر و ستون به‌کار می‌رود. به‌طور مثال، برای حروفچینی ماتریس‌ها می‌توان از محیط array بهره گرفت. در این‌باره بخش ۹.۴.۳ در صفحه‌ی ۱۶۵ را ببینید.

center این محیط برای وسط‌گذاری اشیاء به‌کار می‌رود. آنچه درون محیط center جای می‌گیرد، در خروجی با فاصله‌ی مساوی از حاشیه‌های چپ و راست قرار می‌گیرد. در این‌باره بخش ۱.۴.۲ در صفحه‌ی ۹۳ را ببینید.

description این محیط برای پدید آوردن فهرست‌های توضیحی به‌کار می‌رود. در این‌باره بخش ۳.۵.۲ در صفحه‌ی ۱۱۷ را ببینید.

displaymath این محیط برای حروفچینی عبارت‌های ریاضی به صورت جِلوه یافته یا نمایشی به‌کار می‌رود. در این‌باره بخش ۱.۳ در صفحه‌ی ۱۵۰ را ببینید.

document اصلی‌ترین محیط در LaTeX است. بدنه‌ی اصلی متن درون محیط document قرار می‌گیرد.

enumerate این محیط برای پدید آوردن فهرست‌های شمارشی به‌کار می‌رود. در این‌باره بخش ۲.۵.۲ در صفحه‌ی ۱۱۱ را ببینید.

equation این محیط برای حروفچینی فرمول‌ها و معادله‌های ریاضی، که قرار است از سوی LaTeX به‌طور خودکار شماره‌گذاری شود، به‌کار می‌رود. در این‌باره بخش ۱۰.۳ در صفحه‌ی ۲۰۵ را ببینید.

figure این محیط برای پدید آوردن شکل‌های شناور و شکل‌هایی که قرار است از سوی LaTeX به‌طور خودکار شماره‌گذاری شود، به‌کار می‌رود. در این‌باره فصل ۷ را ببینید.

flushleft و flushright محیط flushleft برای راندن متن به حاشیه‌ی سمت چپ و محیط flushright برای راندن متن به حاشیه‌ی سمت راست به‌کار می‌رود. در این‌باره بخش ۲.۴.۲ در صفحه‌ی ۹۴ را ببینید.

itemize این محیط برای پدید آوردن فهرست‌های به‌هم‌ریخته یا تصادفی به‌کار می‌رود. در این‌باره بخش ۱.۵.۲ در صفحه‌ی ۱۰۶ را ببینید.

letter این محیط منحصراً در رده‌ی نوشتاری letter تعریف شده است و برای حروفچینی نامه به‌کار می‌رود. در این‌باره بخش ۲.۱.۴ در صفحه‌ی ۲۵۵ را ببینید.

math این محیط برای حروفچینی عبارت‌های ریاضی به صورت درون-خطی به‌کار می‌رود. در این‌باره بخش ۱.۳ در صفحه‌ی ۱۵۰ را ببینید.

minipage این محیط برای پدید آوردن جعبه‌های پاراگرافی به‌کار می‌رود. در این‌باره بخش ۸.۶.۱ در صفحه‌ی ۴۹ را ببینید.

quotation و quote از این دو محیط برای حروفچینی یک متن به صورت جِلوه یافته استفاده می‌شود؛ به‌عنوان نمونه برای نقل قول مستقیم. در این‌باره بخش ۳.۴.۲ در صفحه‌ی ۹۶ را ببینید.

table این محیط برای پدید آوردن جدول‌های شناور و جدول‌هایی که قرار است از سوی LaTeX به‌طور خودکار شماره‌گذاری شود، به‌کار می‌رود. در این‌باره فصل ۷ را ببینید.

tabular این محیط برای رسم جدول به‌کار می‌رود. در این‌باره بخش ۶.۲ در صفحه‌ی ۱۱۸ را ببینید.

thebibliography این محیط برای حروفچینی کتابنامه یا فهرست منابع به‌کار می‌رود. در این‌باره بخش ۱.۶.۴ در صفحه‌ی ۲۷۱ را ببینید.

verbatim این محیط برای چاپ لفظ به لفظ [۲۲] به‌کار می‌رود؛ بدین معنی که متنِ جای گرفته درون محیط verbatim، عیناً با همان آرایشی که در فایل ورودی ظاهر شده است، در خروجی ظاهر می‌شود. در این‌باره بخش ۱۱.۱ را ببینید.

verse این محیط برای حروفچینی شعر به‌کار می‌رود. در این‌باره بخش ۴.۴.۲ در صفحه‌ی ۱۰۱ را ببینید.

[22] Verbatim

۴.۶.۱ تعریف محیط جدید

ممکن است محیط‌های تعریف شده در LaTeX و بسته‌های جانبی، همه‌ی نیاز کاربر را برآورده نکند. این موضوع در LaTeX پیش‌بینی شده است و از این‌رو امکان تعریف کردن محیط‌های جدید توسط کاربر فراهم شده است. این کار با کمک فرمان \newenvironment قابل انجام است. صورت کلی این فرمان به شکل زیر است.

```
\newenvironment{⟨new-env-name⟩}[⟨number⟩][⟨default⟩]%
               {⟨begin-def⟩}
               {⟨end-def⟩}
```

در اینجا ⟨*new-env-name*⟩ نامی است که کاربر برای محیط جدید برمی‌گزیند. این نام باید منحصراً از کاراکترهای حرفی {a,A,b,B,...,z,Z} تشکیل گردد و به‌علاوه این نام باید یکتا باشد، یعنی از پیش برای منظور دیگری استفاده نشده باشد. به‌طور مثال، اگر نام parbox برای محیط جدید انتخاب شود، از آنجا که در LaTeX فرمان \parbox تعریف شده است، از جانب LaTeX پیغام خطای

```
Command \parbox already defined
```

صادر می‌گردد.

⟨*number*⟩ عددی است در مجموعه‌ی {۱, ۲, ..., ۹} که مشخص‌کننده‌ی تعداد آرگومان‌های محیط جدید است. وارد کردن گزینه‌ی ⟨*number*⟩ اختیاری است و چنانچه، هنگام تعریف محیط جدید، این گزینه وارد نشود به این معنی است که محیط جدید هیچ آرگومانی نمی‌پذیرد.

وارد کردن گزینه‌ی ⟨*default*⟩ هم اختیاری است. اگر این گزینه وارد شود آنگاه LaTeX نخستین آرگومان محیط جدید را به‌عنوان یک آرگومان اختیاری محسوب می‌کند و به این ترتیب چنانچه، هنگام فراخوانی محیط جدید، گزینه‌ی اختیاری آن توسط کاربر وارد نشود، LaTeX به‌طور پیش‌فرض گزینه‌ی ⟨*default*⟩ را برای آن منظور می‌کند. روشن است وارد کردن گزینه‌ی اختیاری ⟨*default*⟩ هنگامی معنی‌دار است که پیش از آن گزینه‌ی ⟨*number*⟩ وارد شده باشد.

و بالاخره آنکه ⟨*begin-def*⟩ اعمالی است که LaTeX هنگام ورود به محیط انجام می‌دهد و ⟨*end-def*⟩ کارهایی است که LaTeX هنگام خروج از آن انجام می‌دهد. معمولاً در ⟨*begin-def*⟩ و ⟨*end-def*⟩ از فرمان‌ها و محیط‌های از پیش تعریف شده در LaTeX استفاده می‌شود. این موضوع در مثال‌های پیشِ رو روشن می‌گردد.

مثال ۱. فرض کنیم محیطِ ⟨*old-name*⟩ در LaTeX از پیش تعریف شده است و کاربر به‌مراتب از آن استفاده می‌کند اما، بنا به دلایلی، کاربر دوست دارد آن را با نامِ جدید ⟨*new-name*⟩ به‌کار برد. در این صورت می‌توان به شکل زیر محیطی جدید با نام ⟨*new-name*⟩ و با همان کارکرد محیط ⟨*old-name*⟩ تعریف کرد.

```
\newenvironment{⟨new-name⟩}{\begin{⟨old-name⟩}}{\end{⟨old-name⟩}}
```

در واقع فراخوانی محیطِ ⟨*new-name*⟩ هم‌ارز با فراخوانی محیط ⟨*old-name*⟩ است.

مثال ۲. به شکل ۲.۱ نگاه کنید. در این شکل محیط جدیدی با نام `itabstract` تعریف شده است، که این محیط برای حروفچینی چکیده‌ی نوشتار با قلمِ *italic* مناسب است.

مثال ۳. به شکل ۳.۱ نگاه کنید. در این شکل محیط جدیدی با نام `namedthm` تعریف شده است. این محیط برای حروفچینی قضیه‌هایی که مشهور به نامِ ویژه‌ای هستند، مناسب است. در این شکل از فرمانِ `\newtheorem*` سود جسته‌ایم که برای جزئیات بیشتر، درباره‌ی این فرمان، بخش ۲۰.۳ در صفحه‌ی ۲۲۹ را ملاحظه نمایید. دقت کنید محیط جدید `namedthm` دارای یک آرگومان است که آن هم اختیاری می‌باشد. مشاهده می‌شود، آنجا که آرگومان اختیاری وارد نشده است، به‌طور پیش‌فرض، واژه‌ی Theorem برای آن منظور گردیده است.

فرمان `\renewenvironment`. همان‌طور که در ابتدا بیان شد، نامِ انتخابی برای محیط جدید باید یکتا باشد و از پیش برای LaTeX تعریف شده نباشد. گاهی، بنا به دلایلی، نیاز

```
\newenvironment{itabstract}%
   {\begin{center} \textsc{Abstract} \end{center} \begin{quote} \itshape}
   {\end{quote}}
\begin{itabstract}
  We consider when certain Banach sequence algebras $A$ on the set $N$ are
  approximately amenable. Some general results are obtained, and we resolve
  the special cases where $A = \ell^p$ for $1\leq p < \infty$.
\end{itabstract}
\section{Introduction}   The concept of amenability for a Banach algebra $A$,
introduced by Johnson in 1972.
```

Abstract

We consider when certain Banach sequence algebras A on the set N are approximately amenable. Some general results are obtained, and we resolve the special cases where $A = \ell^p$ for $1 \leq p < \infty$.

1 Introduction

The concept of amenability for a Banach algebra A, introduced by Johnson in 1972.

شکلِ ۲.۱: نمونه‌ای از تعریف محیط جدید برای حروفچینی چکیده

```
\newcommand{\thisname}{}
\newtheorem*{thisthm}{\thisname}

\newenvironment{namedthm}[1][Theorem]%
{\renewcommand{\thisname}{#1}\begin{thisthm}}{\end{thisthm}}

\begin{namedthm}[The Hausdorff Maximal Principle]
  Every partially ordered set has a maximal linearly ordered subset.
\end{namedthm}

\begin{namedthm}[Zorn Lemma]
  If $X$ is a partially ordered set and every linearly ordered subset of
  $X$ has an upper bound, then $X$ has a maximal element.
\end{namedthm}

\begin{namedthm}
  Let $X$ be a compact space. If $f:X\to\mathbb{R}$ is continuous then there
  exist $p,q\in X$ such that, for every $x\in X$, $f(p)\leq f(x) \leq f(q)$.
\end{namedthm}
```

The Hausdorff Maximal Principle. *Every partially ordered set has a maximal linearly ordered subset.*

Zorn Lemma. *If X is a partially ordered set and every linearly ordered subset of X has an upper bound, then X has a maximal element.*

Theorem. *Let X be a compact space. If $f : X \to \mathbb{R}$ is continuous then there exist $p, q \in X$ such that, for every $x \in X$, $f(p) \leq f(x) \leq f(q)$.*

شکلِ ۳.۱: نمونه‌ای از تعریف محیط جدید برای حروفچینی قضیه‌ها

است از یک نام موجود و تعریف شده برای محیط جدید استفاده کنیم؛ مثلاً آنجا که بخواهیم عملکرد یک محیط موجود و از پیش تعریف شده را تغییر دهیم. در این صورت باید از فرمان \renewenvironment بهره گرفت. صورت کلی این فرمان به‌شکل زیر است.

```
\renewenvironment{⟨old-env-name⟩}[⟨number⟩][⟨default⟩]%
                 {⟨begin-def⟩}
                 {⟨end-def⟩}
```

همه چیز با فرمان \newenvironment یکی است جز آنکه باید ⟨*old-env-name*⟩ از پیش برای LaTeX تعریف شده باشد.

۵.۶.۱ شمارنده‌ها

یکی از کتابهای مورد علاقه‌ی خود را، که به زبان انگلیسی حروفچینی شده است، از قفسه برداشته تورق نمایید. مشاهده می‌شود اجزای گوناگونی در آن کتاب هست که شماره‌گذاری شده است. به‌طور اعم صفحات، فصل‌ها و بخش‌ها و بخصوص اگر یک کتاب ریاضی باشد تعریف‌ها، لم‌ها، قضیه‌ها و بسیاری از معادله‌ها شماره‌گذاری شده است. حال به چگونگی شماره‌گذاری این اشیاء دقت کنید.

- احتمالاً صفحات اولیه کتاب شامل پیشگفتار، فهرست مطالب و مانند آن، با ارقام یونانی کوچک {i, ii, iii, ...} شماره‌گذاری شده است و با آغاز متن اصلی از ابتدای فصل اول، صفحات با ارقام {1, 2, 3, ...} شماره‌گذاری شده است.
- شماره‌گذاری فصل‌ها با ارقام {1, 2, 3, ...} و یا احتمالاً با ارقام یونانی بزرگ {I, II, III, ...} انجام شده است.
- در شماره‌گذاری بخش‌ها، شماره‌ی فصل جاری نیز لحاظ شده است. به‌علاوه، هرگاه شماره فصل تغییر می‌کند شمارگذاری بخش‌ها از ابتدا آغاز می‌شود، مانند

1.1, 1.2, 1.3, ... 2.1, 2.2, 2.3, ...

- احتمالاً در پایان کتاب پیوست‌ها و ضمیمه‌ها با حروف بزرگ الفبای انگلیسی {A, B, C, ...} شماره‌گذاری شده است.

در امر شماره‌گذاری خودکار، LaTeX از توانایی چشمگیری برخوردار است. اگر کاربر قادر باشد به‌درستی از امکانات و توانایی‌های LaTeX بهره گیرد، هیچگاه نیازی به شماره‌گذاری اشیاء به‌طور دستی نیست. LaTeX قادر است هر شیئی را، که به‌طور طبیعی نیاز به شماره‌گذاری دارد، به‌طور خودکار و به شیوه‌های گوناگون شماره‌گذاری کند؛ از صفحات و فصل‌ها و بخش‌ها گرفته تا شکل‌ها و جدول‌ها و حتی گزینه‌های یک فهرست شمارشی.

LaTeX برای شماره‌گذاری هر شیء از یک متغیر شمارشی، که به آن **شمارنده** گوییم، استفاده می‌کند. در زیر گردایه‌ای از تمام شمارنده‌های موجود و از پیش تعریف شده در LaTeX استاندارد جمع‌آوری شده است، که از نام هریک پیدا است برای شمارش چه جزئی به‌کار می‌رود. به‌عنوان نمونه، شماره‌ی بخش جاری در متغیر شمارشی `section` ذخیره می‌شود.

```
part            paragraph       figure          enumi
chapter         subparagraph    table           enumii
section         page            footnote        enumiii
subsection      equation        mpfootnote      enumiv
subsubsection
```

به‌طور طبیعی، انتظار داریم شیئی که قرار است، به‌طور خودکار، توسط LaTeX شماره‌گذاری گردد با یک فرمان مخصوص یا با استفاده از یک محیط مخصوص پدید آید. به عنوان مثال، برای ایجاد فصلی با عنوان ⟨*chapter-title*⟩ در LaTeX فرمان `\chapter{`⟨*chapter-title*⟩`}` بکار می‌رود و برای آنکه معادله‌ای از سوی LaTeX به‌طور خودکار شماره‌گذاری شود کافی است آن را در محیط `\begin{equation} ... \end{equation}` قرار داد.

در LaTeX نام شمارنده‌ی یک شیء شماره‌گذاری شده، با نام فرمان یا محیطی که پدیدآور آن شیء می‌باشد، یکسان است. به‌طور مثال، LaTeX برای شمارش فصل‌ها از شمارنده‌ی `chapter` و برای شماره‌گذاری معادله‌ها از شمارنده‌ی `equation` استفاده می‌کند.

ارزش یک شمارنده و قالب نمایشی آن. اکنون به‌طور کلی فرض کنیم ⟨*counter*⟩ یکی از شمارنده‌های از پیش تعریف شده در LaTeX باشد. ارزش این شمارنده عددی صحیح است که

با فرمان \value{⟨*counter*⟩} قابل دسترسی است (اما نه قابل نمایش!) و **قالب نمایشی** این شمارنده قالبی است که نمایش‌دهنده‌ی ارزش آن است و با فرمان \the⟨*counter*⟩ قابل دسترسی و قابل نمایش است.

به عنوان مثال، در بخش سوم از فصل چهارم یک کتاب، ارزش شمارنده‌ی chapter مقدار ۴ و ارزش شمارنده‌ی section مقدار ۳ است. درحالی که قالب نمایشی شمارنده‌ی chapter ممکن است IV و قالب نمایشی شمارنده‌ی section ممکن است IV.3 باشد.

در LaTeX استاندارد پنج قالب برای نمایش شماره‌ها وجود دارد که در جدول ۲.۱ در صفحه‌ی ۳۹ گردآوری شده است. چنانچه بخواهیم به محتوای یک شمارنده در یکی از قالب‌های نمایشی جدول ۲.۱ دست یابیم، کافی است نام قالب مورد نظر را به‌عنوان فرمانی که شمارنده‌ی مورد نظر آرگومان آن است، به‌کار بریم.

arabic	در این قالب ارزش شمارنده با ارقام عربی {1, 2, 3, ...} نمایش داده می‌شود.
roman	در این قالب ارزش شمارنده با ارقام یونانی کوچک {i, ii, iii, ...} نمایش داده می‌شود.
Roman	در این قالب ارزش شمارنده با ارقام یونانی بزرگ {I, II, III, ...} نمایش داده می‌شود.
alph	در این قالب ارزش شمارنده با حروف کوچک الفبای انگلیسی {a, b, c, ...} نمایش داده می‌شود. در این حالت ارزش شمارنده باید بین ۱ تا ۲۶ باشد.
Alph	در این قالب ارزش شمارنده با حروف بزرگ الفبای انگلیسی {A, B, C, ...} نمایش داده می‌شود. در این حالت ارزش شمارنده باید بین ۱ تا ۲۶ باشد.
fnsymbol	در این قالب ارزش شمارنده با نشانه‌های پاورقی *، †، ‡، §، ¶، ‖، **، ††، ‡‡ نمایش داده می‌شود. در این حالت ارزش شمارنده باید بین ۱ تا ۹ باشد.

جدولِ ۲.۱: قالب نمایشی برای شمارنده‌ها

به‌طور مثال، اگر در بخش سوم از فصل چهارم یک کتاب فرمان \Alph{chapter} صادر شود، حاصل آن نمایش شماره‌ی فصل به صورت D است و چنانچه فرمان \roman{section}

صادر گردد، حاصل آن نمایش شماره‌ی بخش به صورت iii می‌باشد. در این‌باره شکل ۴.۱ را ببینید.

تغییر دادن ارزش یک شمارنده. به‌آسانی می‌توان ارزش یک شمارنده را تغییر داد و این کار با یکی از فرمان‌های زیر انجام می‌شود.

```
\setcounter{⟨counter⟩}{⟨value⟩}
\setcounter{⟨counter1⟩}{\value{⟨counter2⟩}}
\addtocounter{⟨counter⟩}{⟨value⟩}
\addtocounter{⟨counter1⟩}{\value{⟨counter2⟩}}
```

در فرمان‌های اول و سوم ⟨*value*⟩ یک عدد صحیح است. دومین فرمان باعث می‌شود ارزش شمارنده‌ی ⟨*counter1*⟩ برابر با ارزش شمارنده‌ی ⟨*counter2*⟩ منظور شود. سومین فرمان موجب می‌شود به ارزش شمارنده‌ی ⟨*counter*⟩ مقدار ⟨*value*⟩ اضافه گردد.

تغییر دادن چگونگی نمایش یک شمارنده. چگونگی نمایش شمارنده‌ی ⟨*counter*⟩ در چگونگی تعریفِ فرمان \the⟨*counter*⟩ نهفته است. اگر بخواهیم چگونگی نمایش شمارنده‌ی ⟨*counter*⟩ را تغییر دهیم باید تعریفِ فرمان \the⟨*counter*⟩ را به‌طور مناسب، با کمک دستور \renewcommand، تغییر دهیم. شکل ۴.۱ این موضوع را روشن می‌کند.

تعریف شمارنده جدید. در صورت نیاز، کاربر می‌تواند شمارنده‌ی جدید تعریف کند و این کار با کمک فرمان \newcounter به صورت زیر انجام می‌شود.

```
\newcounter{⟨new-counter⟩}[⟨old-counter⟩]
```

در اینجا ⟨*new-counter*⟩ نام شمارنده‌ی جدیدی است که توسط کاربر انتخاب می‌شود و کاربر از آن در طول متن استفاده می‌نماید و ⟨*old-counter*⟩ نام یک شمارنده‌ی از پیش تعریف شده است. باید دقت کرد نامی که برای شمارنده‌ی جدید برگزیده می‌شود یکتا باشد، یعنی از پیش برای منظور دیگری تعریف نشده باشد.

```
\setcounter{section}{5}
\setcounter{subsection}{3}
\setcounter{equation}{8}

\renewcommand{\thesection}{\Alph{section}}
\renewcommand{\thesubsection}{\thesection.\Roman{subsection}}
\renewcommand{\theequation}{\thesubsection-\arabic{equation}}

\subsection{Counters}

\verb|\thesection|=\thesection,\hspace{\fill}
\verb|\alph{section}|=\alph{section}                \\
\verb|\thesubsection|=\thesubsection,\hspace{\fill}
\verb|\arabic{subsection}|=\arabic{subsection}     \\
\verb|\theequation|=\theequation,\hspace{\fill}
\verb|\roman{equation}|=\roman{equation}

\subsection{Conclusion}
Therefore,
   \begin{equation}
      a^2+b^2=c^2
   \end{equation}
```

E.IV Counters

\thesection=E, \alph{section}=e
\thesubsection=E.IV, \arabic{subsection}=4
\theequation=E.IV-8, \roman{equation}=viii

E.V Conclusion

Therefore,

$$a^2 + b^2 = c^2 \tag{E.V-9}$$

شکلِ ۴.۱: تغییر دادن چگونگی نمایش یک شمارنده

وارد کردن ⟨*old-counter*⟩ اختیاری است. وارد کردن گزینه اختیاری ⟨*old-counter*⟩ باعث می‌شود هرگاه شمارنده‌ی ⟨*old-counter*⟩ تغییر کند شمارش در ⟨*new-counter*⟩ از ابتدا آغاز شود، مانند رابطه‌ی بین شمارنده‌های `section` و `chapter` در رده‌های نوشتاری `book` و `report`.

به این ترتیب آنچه برای تعریف شمارنده‌های مربوط به فصل‌بندی و بخش‌بندی در فایل `book.cls` آمده است، احتمالاً، چیزی شبیه به شکل زیر است.

```
\newcounter{chapter}
\newcounter{section}[chapter]
\newcounter{subsection}[section]
\newcommand{\thechapter}{\arabic{chapter}}
\newcommand{\thesection}{\thechapter.\arabic{section}}
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

۶.۶.۱ طول‌ها و متغیرهای طولی

برای آنکه LaTeX کار حروفچینی را به‌خوبی انجام دهد نیاز به اطلاعات فراوانی دارد. دسته‌ی مهمی از این اطلاعات مربوط به اندازه‌های طولی می‌شود، مانند ابعاد کاغذ، فاصله‌ی بین خطوط، اندازه‌ی حاشیه‌ها، طول هر خط و صدها اندازه‌ی طولی دیگر که در نحوه‌ی حروفچینی دخالت دارند.

در LaTeX استاندارد برای هریک از اندازه‌های طولی مورد نیاز یک متغیر طولی تعریف شده است و در هر متغیر طولی، به‌طور پیش‌فرض و بسته به رده‌ی نوشتاری، یک مقدار اولیه ذخیره شده است. اما به‌طور آشکار، بسته به شرایط، لازم است بسیاری از این اندازه‌ها به‌طور مناسبی تغییر کند. از این‌رو در LaTeX فرمان‌های ویژه‌ای برای مقداردهی به متغیرهای طولی فراهم شده است. در جدول ۳.۱ گردایه‌ای از مهمترین متغیرهای طولی از پیش تعریف شده در LaTeX استاندارد، جمع‌آوری شده است. شکل ۱.۵ در صفحه‌ی ۳۰۱، هم نمایش دهنده‌ی متغیرهای طولی در یک صفحه می‌باشد. دقت کنید، نام هر متغیر طولی با کراکتر \ آغاز می‌شود.

در LaTeX اندازه‌ها بر حسب واحدهای طولی مشخصی بیان می‌شود. واحدهای طولی تعریف شده در LaTeX در جدول ۴.۱ گردآوری شده است.

متغیر طولی	شرح
\paperwith	عرض کاغذ
\paperheigh	ارتفاع کاغذ
\oddsidemargin	اندازه‌ی حاشیه‌ی چپ در حروفچینی یک‌رو
\evensidemargin	اندازه‌ی حاشیه‌ی چپ برای صفحات زوج در حروفچینی دورو
\textwidth	عرض بدنه‌ی اصلی متن
\topmargin	اندازه‌ی حاشیه‌ی بالایی
\textheigh	ارتفاع بدنه‌ی اصلی متن
\linewidth	طول خط جاری
\parindent	میزان تورفتگی اولین خط از هر پاراگراف
\mathindent	میزان تورفتگی عبارت‌های ریاضی در حالت جِلوه یافته هنگامی که گزینه‌ی fleqn بر فرمان \documentclass اعمال شده است.

جدولِ ۳.۱: برخی متغیرهای طولی از پیش تعریف شده در LaTeX

واحد اندازه‌گیری	شرح
pt (point)	۱ pt = ۰/۰۱۳۸۳۷ in = ۰/۳۵۱ mm
sp (scaled point)	کوچکترین واحد طولی در LaTeX، ۶۵۵۳۶ sp = ۱ pt
bp (big point)	۷۲ bp = ۱ in
mm (milimeter)	۱ mm = ۲/۸۴۵ pt
pc (pica)	۱ pc = ۱۲ pt = ۴/۲۱۸ mm
cm (centimeter)	۱ cm = ۱۰ mm = ۲/۳۷۱ pc
in (inch)	۱ in = ۲۵/۴ mm = ۷۲/۲۷ pt
ex	تقریباً ارتفاع حرف x در اندازه‌ی قلم جاری
em	تقریباً پهنای حرف M در اندازه‌ی قلم جاری
mu	واحد اندازه‌گیری درحالت ریاضی، ۱۸ mu = ۱ em

جدولِ ۴.۱: واحدهای اندازه‌گیری طولی قابل قبول در LaTeX

تغییر دادن مقدار یک متغیر طولی. به‌آسانی می‌توان مقدار یک متغیر طولی را تغییر داد و این کار با یکی از فرمان‌های زیر انجام می‌شود.

```
\setlength{⟨length-var⟩}{⟨lenght⟩}
\setlength{⟨length-var1⟩}{⟨length-var2⟩}
\addtolength{⟨length-var⟩}{⟨lenght⟩}
\addtolength{⟨length-var1⟩}{⟨length-var2⟩}
```

در اینجا ⟨*length-var*⟩ نام یک متغیر طولی است که همواره با کراکتر \ آغاز می‌شود و ⟨*length*⟩ یک مقدار عددی برحسب یکی از واحدهای طولی بیان شده در جدول ۴.۱ می‌باشد. به‌عنوان مثال، برای آنکه عرض بدنه‌ی اصلی متن مقدار ۱۴ سانتی‌متر منظور شود، می‌توان از فرمان `\setlength{\textwidth}{14cm}` در مقدمه‌ی فایل ورودی بهره گرفت.

به‌عنوان مثالی دیگر به شکل ۵.۱ نگاه کنید. همان‌طور که در این شکل ملاحظه می‌شود، چنانچه بخواهیم فاصله‌ی بین خطوط را به‌طور موضعی تغییر دهیم، تغییر دادن متغیر طولی `\baselineskip`، همراه با گروه‌بندی مناسب، گزینه‌ی خوبی به‌نظر می‌رسد. اما چنانچه بخواهیم فاصله‌ی بین خطوط را در طول نوشتار به‌طور سراسری تغییر دهیم بهتر است، به‌جای دستکاری متغیر طولی `\baselineskip`، ضریب آن یعنی `\baselinestretch` را تغییر داد. این‌کار، با کمک فرمان `\renewcommand`، به صورت زیر انجام می‌شود.

```
\renewcommand{\baselinestretch}{⟨c⟩}
```

در اینجا ⟨c⟩ یک مقدار مثبت است. اگر ⟨c⟩ در بازه‌ی $(۰, ۱)$ باشد خطوط به‌هم نزدیک می‌شوند و اگر ⟨c⟩ در بازه‌ی $(۱, \infty)$ باشد خطوط از هم دور می‌گردند.

تعریف متغیر طولی جدید. ممکن است کاربر به یک متغیر طولی جدید نیاز داشته باشد. تعریف کردن متغیر طولی جدید بسیار آسان است و با کمک فرمان `\newlength` به صورت زیر انجام می‌شود.

```
\newlength{⟨new-length-var⟩}
```

```
\verb|\baselineskip| is a length command which specifies the minimum
space between the bottom of two successive lines in a paragraph.
Its value may be automatically reset by \LaTeX, for example, by font
changes in the text.

\addtolength{\baselineskip}{1.5ex}
\verb|\baselinestretch| scales the value of \verb|\baselineskip|.
Its default value is $1.0$ but it may be reset with
a \verb|\renewcommand| command. If one wants to change the spacing in
a document one should reset \verb|\baselinestretch| and not
\verb|\baselineskip| as the latter may be reset automatically by \LaTeX{}
to account for local variations in the text, but it is always scaled by
the former.
```

`\baselineskip` is a length command which specifies the minimum space between the bottom of two successive lines in a paragraph. Its value may be automatically reset by LaTeX, for example, by font changes in the text.

`\baselinestretch` scales the value of `\baselineskip`. Its default value is 1.0 but it may be reset with a `\renewcommand` command. If one wants to change the spacing in a document one should reset `\baselinestretch` and not `\baselineskip` as the latter may be reset automatically by LaTeX to account for local variations in the text, but it is always scaled by the former.

شکلِ ۵.۱: تغییر دادن فاصله‌ی بین خطوط

در اینجا ⟨*new-length-var*⟩ نام متغیر طولی است که کاربر به دلخواه انتخاب می‌کند. این نام باید یکتا باشد، یعنی از قبل برای LaTeX تعریف شده نباشد و باید همواره با کراکتر \ آغاز گردد.

نمایش مقدار ذخیره شده در یک متغیر طولی. با فرمان \the⟨*length-var*⟩ می‌توان مقدار ذخیره شده در متغیر طولی ⟨*length-var*⟩ را نمایش داد. شکل ۶.۱ را ببینید.

```
\sffamily \verb|\oddsidemargin| = \the\oddsidemargin \hfill
                                \verb|\textwidth| = \the\textwidth \\[1ex]
\newlength{\mylen}\setlength{\mylen}{1em} 1em = \the\mylen  \hfill
\addtolength{\mylen}{1ex}                    1em + 1ex = \the\mylen \\[1ex]
\settowidth{\mylen}{abc} width of abc = \the\mylen \hfill
\settowidth{\mylen}{ABC}                  width of ABC = \the\mylen \\ [1ex]
\settowidth{\mylen}{\Large ABC}    width of {\Large ABC} = \the\mylen
```

\oddsidemargin = 34.42784pt	\textwidth = 369.31221pt
1em = 10.00002pt	1em + 1ex = 14.44446pt
width of abc = 14.69447pt	width of ABC = 19.72227pt
width of ABC = 27.68298pt	

شکلِ ۶.۱: نمایش مقدار ذخیره شده در یک متغیر طولی

همان‌طور که در شکل ۶.۱ دیده می‌شود، به‌طور کلی، برای ذخیره کردن عرض شیء ⟨*object*⟩ در متغیر طولی ⟨*length-var*⟩ از فرمان

```
\settowidth{⟨length-var⟩}{⟨object⟩}
```

استفاده می‌شود. به‌طور مشابه، برای ذخیره کردن ارتفاع و عمق شیء ⟨*object*⟩ در متغیر طولی ⟨*length-var*⟩ از فرمان‌های زیر استفاده می‌شود.

```
\settoheight{⟨length-var⟩}{⟨object⟩}
```

```
\settodepth{⟨length-var⟩}{⟨object⟩}
```

۷.۶.۱ متغیرهای طولی انعطاف‌پذیر

برای آشنایی بهتر با متغیرهای طولی انعطاف‌پذیر، ابتدا به متغیرهای طولی از پیش تعریف شده‌ی \smallskipamount و \bigskipamount توجه می‌کنیم. با صادر کردن فرمان \smallskip فضای خالی عمودی، به اندازه‌ای که در متغیر طولی \smallskipamount نگهداری می‌شود، پدید می‌آید و با صادر کردن فرمان \bigskip فضای خالی عمودی، به اندازه‌ای که در متغیر طولی \bigskipamount ذخیره شده است، ایجاد می‌شود.

از قسمت قبل می‌دانیم، برای نمایش مقدار ذخیره شده در یک متغیر طولی، از فرمان \the، درست پیش از آن متغیر طولی، استفاده می‌کنیم.

```
\sffamily
\the\smallskipamount  \hspace{\fill} \the\bigskipamount
```

3.0pt plus 1.0pt minus 1.0pt 12.0pt plus 4.0pt minus 4.0pt

مثال بالا نشان می‌دهد که متغیر طولی \smallskipamount دارای مقدار 3pt است، که البته می‌تواند، در صورت نیاز، به اندازه‌ی 1pt فشرده گردد و به اندازه‌ی 1pt هم منبسط شود، و متغیر طولی \bigskipamount دارای مقدار 12pt است، که البته می‌تواند، در صورت نیاز، به اندازه‌ی 4pt فشرده گردد و به اندازه‌ی 4pt هم منبسط شود.

دو متغیر طولی \smallskipamount و \bigskipamount نمونه‌ای از متغیرهای طولی انعطاف‌پذیر در LaTeX است. LaTeX از متغیرهای طولی انعطاف‌پذیر، برای برقراری توازن در چیدن پاراگراف‌ها و اشیاء در یک صفحه، بهره می‌گیرد.

به‌طور کلی، فرض کنیم ⟨*length*⟩، ⟨δ⟩ و ⟨ε⟩ مقادیر عددی، برحسب واحدهای طولی قابل قبول در LaTeX باشند، که ⟨δ⟩ و ⟨ε⟩ مثبت هستند. فرض کنیم ⟨*len-var*⟩ یک متغیر طولی باشد. در این صورت، فرمان

```
\setlength{⟨len-var⟩}{⟨length⟩ plus ⟨ε⟩ minus ⟨δ⟩}
```

سبب می‌شود مقدار ⟨*length*⟩ برای متغیر طولی ⟨*len-var*⟩ منظور گردد، به‌طوری که، در صورت نیاز، این متغیر طولی بتواند به اندازه‌ی ⟨δ⟩ فشرده گردد و به اندازه‌ی ⟨ε⟩ منبسط شود.

```
\sffamily \newlength{\mylength}
\setlength{\mylength}{.1in plus 2mm minus 5pt}  \the\mylength \hfill
\addtolength{\mylength}{2ex}                    \the\mylength
```

7.22743pt plus 5.69054pt minus 5.0pt 16.11632pt plus 5.69054pt minus 5.0pt

یکی از متغیرهای طولی انعطاف‌پذیر پرکاربر و پراهمیت در LaTeX، متغیر طولی `\fill` می‌باشد. این متغیر طولی در بازه‌ی $(0, \infty)$ تغییر می‌کند.

```
A B C \par                          A \hspace{\fill} B C \par
A B \hspace{\fill} C \par           A \hspace{\fill} B \hspace{\fill} C \par
```

A B C
A B C
A B C
A B C

۸.۶.۱ جعبه‌ها

برای آنکه آسانتر به بررسی مفهوم جعبه در LaTeX بپردازیم، ابتدا به متن زیر، که توسط LaTeX حروفچینی شده است، توجه نمایید.

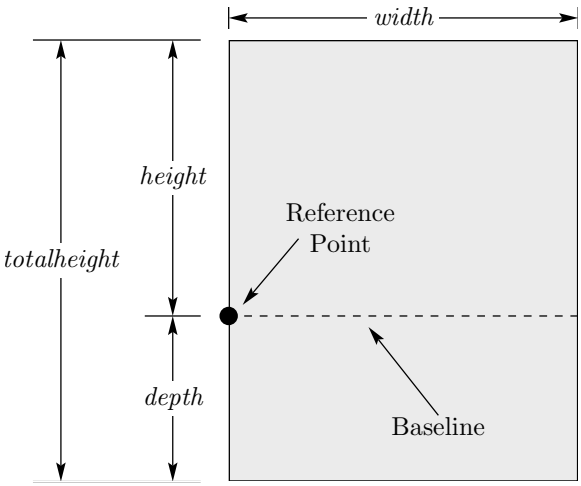
A **box** is any LaTeX object that is treated as a unit (a single character).
Characters, framed text, graphics such as StarLines, and tabular materials such as | A | B | / | C | D | are examples of boxes in LaTeX.

در متن بالا، LaTeX برای چیدن جدول، تصویر گرافیکی و عبارت قاب‌دار framed text روی خط زمینه، همانند سایر کراکترها و حروف رفتار کرده است.

یک **جعبه**[۲۳] در LaTeX عبارت است از یک شیء یا مجموعه‌ای از اشیاء که از جانب LaTeX به‌عنوان یک کراکتر در نظر گرفته می‌شود. بنابراین یک جعبه قابل تقسیم به دو بخش نیست.

[23] Box

ابتدا شکل ۷.۱ را ببینید. تصور ظاهری ما از یک جعبه در LaTeX، همان‌طور که این شکل نشان می‌دهد، مستطیلی است که اضلاع آن موازی محورهای فرضی مختصات است. پهنای این مستطیل، که با $width$ نمایش داده شده است، برابر است با بیشترین فضایی که جعبه از لحاظ افقی اشغال می‌کند و ارتفاع این مستطیل، که با $totalheight$ نمایش داده شده است، برابر است با بیشترین فضایی که این جعبه از لحاظ عمودی اشغال می‌کند.



شکلِ ۷.۱: شکل نمادین یک جعبه در LaTeX

هر جعبه در LaTeX دارای یک **نقطه‌ی پایه**[۲۴] است، به‌طوری که ابعاد جعبه و مختصات سایر نقاط جعبه، نسبت به این نقطه سنجیده می‌شود. نقطه‌ی پایه همواره بر منتهاالیه سمت چپ جعبه قرار دارد و به‌طور فرضی، مبدأ مختصات منطبق بر نقطه‌ی پایه است. برای یک جعبه، **خط پایه**[۲۵] خطی افقی است که از نقطه‌ی پایه می‌گذرد و بنابراین خط پایه منطبق بر محور طول‌ها است.

[24] Reference point

[25] Baseline

ابعاد جعبه. اندازه‌ی یک جعبه با سه مشخصه تعیین می‌شود:

- ارتفاع جعبه که با $height$ نمایش داده می‌شود و $height$ برابر است با فاصله‌ی خط پایه تا بالاترین نقطه‌ی جعبه یا همان سقف جعبه،
- عمق جعبه که با $depth$ نمایش داده می‌شود و $depth$ برابر است با فاصله‌ی خط پایه تا پایین‌ترین نقطه‌ی جعبه یا همان کف جعبه،
- پهنای جعبه که با $width$ نمایش داده می‌شود و $width$ برابر است با بیشترین فضایی که جعبه از لحاظ افقی اشغال می‌کند،
- درازای جعبه یا ارتفاع کل آن که با $totalheight$ نمایش داده می‌شود و $totalheight$ برابر است با بیشترین فضایی که جعبه از لحاظ عمودی اشغال می‌کند. بنابراین

$$totalheight = height + depth.$$

LaTeX کراکترها و حروف را به‌گونه‌ای می‌چیند که نقطه‌ی پایه‌ی آنها روی خط زمینه واقع گردد و بنابراین خط پایه‌ی آنها بر خط زمینه منطبق می‌شود. LaTeX برای قرار دادن هر شیء که یک جعبه تلقی می‌شود به همین صورت عمل می‌کند.

رده‌ی مهمی از جعبه‌ها در LaTeX تصاویر گرافیکی هستند. نقطه‌ی پایه برای یک تصویر گرافیکی ذخیره شده در قالب EPS،[۲۶] منطبق بر گوشه‌ی پایین سمت چپ آن تصویر است. چگونگی گنجاندن تصاویر گرافیکی در نوشتار LaTeX در فصل ۶ توضیح داده شده است.

رده‌ای دیگر از جعبه‌ها در LaTeX جدول‌ها هستند که توسط محیط `tabular` پدید می‌آیند. چگونگی رسم جدول در بخش ۶.۲ آمده است. خواهیم دید چگونه می‌توان موقعیت نقطه‌ی پایه برای یک جدول رسم شده توسط محیط `tabular` را با اعمال گزینه‌های ویژه تعیین نمود.

انواع گوناگون جعبه

در LaTeX جعبه‌ها به انواع گوناگون دسته‌بندی می‌شود، که سه نوع مهم آن عبارت است از ۱) جعبه‌های LR، ۲) جعبه‌های پاراگرافی و ۳) جعبه‌های از نوع Rule، که در ادامه به تشریح

[26] Encapsulated PostScript

هر یک از آنها می‌پردازیم.

جعبه‌های LR. محتویات این جعبه‌ها از چپ به راست[۲۷] چیده می‌شود. در هر جعبه‌ی LR فقط یک خط می‌توان گنجاند هر چند ممکن است این خط بسیار بلند باشد. برای آنکه متن ⟨*text*⟩ درون یک جعبه‌ی LR جای گیرد یکی از فرمان‌های زیر بکار می‌رود.

```
\mbox{⟨text⟩}
\fbox{⟨text⟩}
\makebox[⟨width⟩]{⟨text⟩}
\framebox[⟨width⟩]{⟨text⟩}
\makebox[⟨text-position⟩][⟨width⟩]{⟨text⟩}
\framebox[⟨text-position⟩][⟨width⟩]{⟨text⟩}
```

هر دو فرمان `\mbox` و `\fbox` باعث می‌شود متن ⟨*text*⟩ درون یک جعبه‌ی LR قرار گیرد. طول جعبه و به‌طور دقیق‌تر، با توجه به نمادگذاری در شکل ۷.۱، پهنای جعبه متناسب با طول متن ⟨*text*⟩ و به‌طور خودکار از سوی LaTeX تعیین می‌شود. از آنجا که یک جعبه در LaTeX شیء واحدی است که قابل تقسیم به دو بخش نیست، چنانچه متن ⟨*text*⟩ بلندتر از فضای موجود در خط جاری باشد اضافی آن به حاشیه‌ها نفوذ می‌کند.

```
For instance,
\verb|\fbox{some words}| gives
\fbox{some words} whereas
\verb|\mbox| will do the
\mbox{same thing, but without
the ruled frame around} the text.
```

For instance, `\fbox{some words}` gives some words whereas `\mbox` will do the same thing, but without the ruled frame around the text.

همان‌گونه که مشاهده می‌شود، تفاوت `\mbox` و `\fbox` در قرار دادن قابی اطراف متن ⟨*text*⟩ است. در مثال بالا، از آنجا که متن

some thing, but without the ruled frame around

به‌عنوان یک شیء واحد محسوب می‌شود که طول آن از فضای خالی موجود در خط جاری بیشتر است، اضافی آن در حاشیه جای گرفته است.

[27] Left-to-Right

کارکرد فرمان‌های \makebox و \framebox، به ترتیب، شبیه فرمان‌های \mbox و \fbox است، با این تفاوت که کاربر می‌تواند با وارد کردن گزینه‌ی اختیاری ⟨*width*⟩ که یک مقدار عددی بر حسب یکی از واحدهای طولی قابل قبول در LaTeX است، پهنای جعبه را خود تعیین کند. مقدار ⟨*width*⟩ می‌تواند از طول واقعی متن ⟨*text*⟩، یعنی فضایی که ⟨*text*⟩ از لحاظ افقی اشغال می‌کند، بیشتر باشد و در این حالت قسمتی از جعبه خالی می‌ماند. مقدار ⟨*width*⟩ می‌تواند از طول واقعی متنِ ⟨*text*⟩ کمتر باشد و در این حالت قسمتی از متن خارج جعبه جای می‌گیرد. این موضوع به کاربر امکان می‌دهد دو متن را روی هم قرار دهد!

```
\framebox[5cm]{A few words of advice} \framebox{A few words of advice}
\framebox[2cm]{A few words of advice}
```

A few words of advice A few words of advice A few words of advice

و بلاخره با اعمال گزینه‌ی اختیاری ⟨*text-position*⟩، در فرمان‌های \makebox و \framebox، می‌توان محل جای‌گیری متن ⟨*text*⟩ درون جعبه‌ی LR را تعیین کرد. گزینه‌ی ⟨*text-position*⟩ یکی از حروف {l, c, r, s} است.

[l] باعث می‌شود متن ⟨*text*⟩ به منتهاالیه سمت چپ جعبه رانده شود.

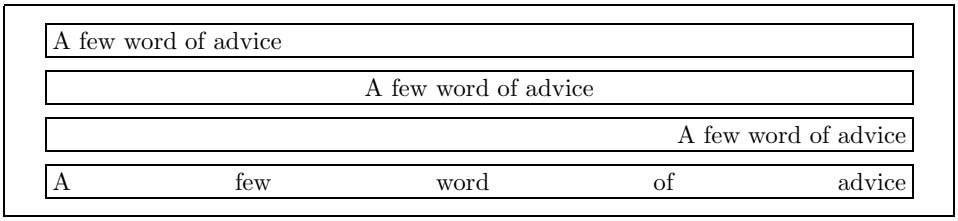
[c] باعث می‌شود متن ⟨*text*⟩ وسط جعبه جای گیرد. این گزینه پیش‌فرض است.

[r] باعث می‌شود متن ⟨*text*⟩ به منتهاالیه سمت راست جعبه رانده شود.

[s] باعث می‌شود متن ⟨*text*⟩ به‌طور متناسب کشیده شود، به‌گونه‌ای که از ابتدا تا انتهای جعبه را اشغال کند. طبیعی است، این گزینه زمانی مناسب است که پهنای جعبه، یعنی ⟨*width*⟩، از طول متن ⟨*text*⟩ بیشتر باشد.

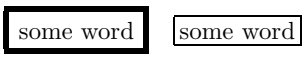
در این‌باره به مثال زیر توجه کنید.

```
\centering\fboxsep3pt
\framebox[.95\textwidth][l]{A few word of advice}  \\[1ex]
\framebox[.95\textwidth]{A few word of advice}     \\[1ex]
\framebox[.95\textwidth][r]{A few word of advice}  \\[1ex]
\framebox[.95\textwidth][s]{A few word of advice}  \\
```



متغیرهای طولی `\fboxrule` **و** `\fboxsep`**.** همان‌طور که بیان شد، دو فرمان `\fbox` و `\framebox` باعث می‌شود قابی محتویات جعبه‌ی LR را فراگیرد. ضخامت این قاب در متغیر طولی `\fboxrule` نگهداری می‌شود، که به‌طور پیش‌فرض مقدار آن `4pt` است، و فاصله‌ای که این قاب تا محتویات جعبه دارد در متغیر طولی `\fboxsep` ذخیره می‌شود، که به‌طور پیش‌فرض مقدار آن `3pt` است. همان‌طور که در بخش ۶.۶.۱ ملاحظه شد، می‌توان این مقادیر را با فرمان‌های `\setlength` و `\addtolength` تغییر داد. دقت کنید، در شکل زیر چگونه از کراکترهای { و }، برای گروه‌بندی اشیاء، سود جسته‌ایم.

```
{\setlength\fboxrule{2pt}  \setlength\fboxsep{4pt} \fbox{some word}}
{\setlength\fboxrule{.5pt} \setlength\fboxsep{2pt} \fbox{some word}}
```



جعبه‌های پاراگرافی. یک جعبه‌ی پاراگرافی با پهنای ⟨*width*⟩ در واقع یک پاراگراف با پهنای ⟨*width*⟩ است. محتویات این جعبه‌ها می‌تواند در چند خط، درون جعبه، قرار گیرد. ارتفاع جعبه بسته به حجم محتویات آن از سوی LaTeX، به‌طور خودکار، تعیین می‌شود. یک جعبه‌ی پاراگرافی به دو روش پدید می‌آید؛ روش اول بکارگیری فرمان `\parbox` است، که صورت کلی آن به شکل زیر می‌باشد.

```
\parbox[⟨base-point-position⟩]{⟨width⟩}{⟨text⟩}
```

و روش دوم استفاده از محیط minipage می‌باشد، که صورت کلی آن به شکل زیر است.

```
\begib{minipage}[⟨base-point-position⟩]{⟨width⟩}
      ⟨text⟩
\end{minipage}
```

در هردو مورد ⟨*width*⟩ یک مقدار عددی برحسب یکی از واحدهای طولی قابل قبول در LaTeX است که مشخص‌کننده‌ی پهنای جعبه است و گزینه‌ی اختیاری ⟨*base-point-position*⟩ تعیین‌کننده‌ی موقعیت نقطه‌ی پایه‌ی جعبه می‌باشد. گزینه‌ی اختیاری ⟨*base-point-position*⟩ یکی از حروف {t, c, b} است.

[t] باعث می‌شود نقطه‌ی پایه‌ی جعبه بر نقطه‌ی پایه‌ی اولین شیء قرار گرفته در جعبه، منطبق گردد.

[c] باعث می‌شود نقطه‌ی پایه‌ی جعبه از لحاظ عمودی وسط جعبه قرار گیرد. می‌دانیم، از لحاظ افقی همواره نقطه‌ی پایه‌ی هر جعبه بر منتهاالیه سمت چپ آن جعبه واقع است. این گزینه پیش‌فرض است.

[b] باعث می‌شود نقطه‌ی پایه‌ی جعبه بر نقطه‌ی پایه‌ی آخرین شیء قرار گرفته در جعبه، منطبق گردد.

توجه کنید گزینه‌ی [t] باعث نمی‌شود خط پایه‌ی جعبه بر سقفِ آن منطبق گردد، همان‌طور که گزینه‌ی [b] باعث نمی‌شود که خط پایه‌ی جعبه بر کفِ آن منطبق شود. در واقع، وقتی یک جعبه‌ی پاراگرافی فقط شامل یک خط باشد، هر سه گزینه‌ی [b]، [c] و [t] نتیجه‌ی یکسان دارد. شکل ۸.۱ را ببینید.

کاربردهای بیشتری از محیط minipage در بخش ۶.۶ در صفحه‌ی ۳۵۲ آمده است.

جعبه‌های از نوع Rule. یک جعبه از نوع Rule در واقع پاره‌خطی است که، بستگی به ابعاد جعبه، ممکن است ضخیم یا نازک، بلند یا کوتاه باشد. این نوع جعبه، به‌طور کلی، با فرمان

```
\parbox{.35\textwidth}{This is the contents of the left-most parbox.}
\hrulefill\ Current Line \hrulefill\ \parbox{.35\textwidth}{This is
the right-most parbox. Note that the typeset text looks sloppy because
\LaTeX{} cannot nicely balance the material in these narrow columns.}

\begin{minipage}[b]{.3\linewidth}
  The \verb|minipage| environment creates a vertical box like the
  \verb|parbox| command. The bottom line of this \verb|minipage|
  is aligned with the
\end{minipage}
  \hrulefill\
\begin{minipage}[c]{.3\linewidth}
  middle of this narrow \verb|parbox|, which in turn is
\end{minipage}
  \hrulefill\
\begin{minipage}[t]{.3\linewidth}
the top line of the right hand {\tt minipage}. It is recommended that the
user experiment with the positioning arguments to get used to their effects.
\end{minipage}
```

This is the contents of the left-most parbox. Current Line This is the right-most parbox. Note that the typeset text looks sloppy because LaTeX cannot nicely balance the material in these narrow columns.

The `minipage` environment creates a vertical box like the `parbox` command. The bottom line of this `minipage` is aligned with the middle of this narrow `parbox`, which in turn is the top line of the right hand `minipage`. It is recommended that the user experiment with the positioning arguments to get used to their effects.

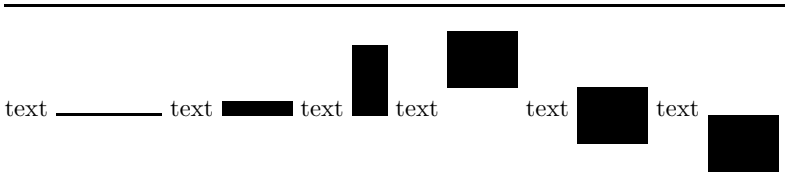
شکلِ ۸.۱: نمونه‌ای از جعبه‌های پاراگرافی در LaTeX

\rule، با صورت کلی زیر، پدید می‌آید.

```
\rule[⟨lift⟩]{⟨width⟩}{⟨totalheight⟩}
```

در اینجا ⟨*width*⟩ پهنای جعبه [=طول پاره‌خط] و ⟨*totalheight*⟩ ارتفاع جعبه [=ضخامت پاره‌خط] می‌باشد. گزینه‌ی اختیاری ⟨*lift*⟩ فاصله‌ی کف جعبه تا خط زمینه را تعیین می‌کند، که به‌نحوی مشخص‌کننده‌ی موقعیت نقطه‌ی پایه‌ی جعبه نیز هست.

```
\rule{\linewidth}{1pt} \\[2ex]
text \rule{1.5cm}{1pt} text \rule{1cm}{2mm} text \rule{5mm}{1cm} text
\rule[4mm]{1cm}{8mm} text \rule[-4mm]{1cm}{8mm} text \rule[-8mm]{1cm}{8mm}
```



۹.۶.۱ جعبه‌های تودرتو

می‌توان یک جعبه را درون جعبه‌های دیگر قرار داد. در شکل ۹.۱، یک جعبه‌ی پاراگرافی درون یک جعبه از نوع LR جای گرفته است.

۷.۱ انتخاب قلم

شکلِ ظاهری حروف و علایم که توسط LaTeX و هر سامانه‌ی حروفچینی دیگر، در خروجی ایجاد می‌شود، ناشی از دو مؤلفه‌ی **شکلِ قلم**[۲۸] و **اندازه‌ی قلم**[۲۹] می‌باشد. به‌عنوان مثال، برای تأکید یک متن آن را به شکل خوابیده، *italic*، می‌نویسند و در این کتاب، برای نشان دادن

[28] Font shape

[29] Font size

```
\centering
\fboxrule2pt\fboxsep2pt\fbox{\fboxrule.5pt\fboxsep3mm\fbox{%
\begin{minipage}{.9\linewidth}
  The box commands described above may be nested to any desired level.
  Including an LR-box within a paragraph-box or a \texttt{minipage} causes
  no obvious conceptual difficulties. The opposite, a paragraph-box within
  an LR-box, is also possible, and is easy to visualize if one keeps in mind
  that every box is a unit, treated by \LaTeX{} as a single character of
  the corresponding size
\end{minipage}}}
```

The box commands described above may be nested to any desired level. Including an LR-box within a paragraph-box or a `minipage` causes no obvious conceptual difficulties. The opposite, a paragraph-box within an LR-box, is also possible, and is easy to visualize if one keeps in mind that every box is a unit, treated by LaTeX as a single character of the corresponding size

شکلِ ۹.۱: نمونه‌ای از جعبه‌های تودرتو

محتویات فایل ورودی، از شکل قلم `typewriter` سود جسته‌ایم. همان‌گونه که شکل قلم را می‌توان عوض کرد، اندازه‌ی آن را نیز می‌توان تغییر داد:

```
We can also produce \small smaller \normalsize and \Large bigger \normalsize type.
```

We can also produce smaller and bigger type.

۱.۷.۱ شکلِ قلم

در LaTeX شکل قلم با سه مشخصه‌ی Shape، Series و Family تعیین می‌شود. جدولِ ۵.۱ دربردارنده‌ی انتخاب‌های مختلف برای این مشخصه‌ها، همراه با فرمان‌های متناظر، می‌باشد. به‌عنوان مثال، فرمان `{\textbf{Some text}}\textit` موجب می‌شود شکل قلم با مشخصه‌های

Shape=*Italic*, Series=**Boldface**, Family=Roman

برای متنِ `Some text` اعمال گردد، که حاصل آن به صورت ***Some text*** می‌باشد.

	Font styles	*Declaration*	*Command*
Shape	upright shape	{\upshape ⟨*text*⟩}	\textup{⟨*text*⟩}
	italic shape	{\itshape ⟨*text*⟩}	\textit{⟨*text*⟩}
	Slanted shape	{\slshape ⟨*text*⟩}	\textsl{⟨*text*⟩}
	SMALL CAPS SHAPE	{\scshape ⟨*text*⟩}	\textsc{⟨*text*⟩}
Series	Medium series	{\mdseries ⟨*text*⟩}	\textmd{⟨*text*⟩}
	Boldface series	{\bfseries ⟨*text*⟩}	\textbf{⟨*text*⟩}
Family	Roman family	{\rmfamily ⟨*text*⟩}	\textrm{⟨*text*⟩}
	Sans serif family	{\sffamily ⟨*text*⟩}	\textsf{⟨*text*⟩}
	Typewrite family	{\ttfamily ⟨*text*⟩}	\texttt{⟨*text*⟩}

جدولِ ۵.۱: مشخصه‌های تعیینِ شکلِ قلم

قلمِ پیش‌فرض، برای متن معمولی، دارای مشخصه‌های Shape=Upright، Series=Medium و Family=Roman است. مثال زیر ترکیبی از مشخصه‌های مختلف را نشان می‌دهد.

```
\textsf{\textbf{Sans Serif family, Boldface Series, Upright Shape.}}   \\[.5ex]
\textbf{\slshape Roman family, Boldface Series, Slanted shape.}         \\[.5ex]
{\scshape \ttfamily Typewriter family, Medium series, Small Caps shape.}
```

Sans Serif family, Boldface Series, Upright Shape.
Roman family, Boldface Series, Slanted shape.
TYPEWRITER FAMILY, MEDIUM SERIES, SMALL CAPS SHAPE.

همان‌گونه که در جدولِ ۵.۱ آمده است، به دو صورت می‌توان مشخصه‌ی شکلِ قلم را تغییر داد:

۱ـ فرمانی که متن مورد نظر آرگومان آن است، مانند \textbf{⟨*text*⟩}،

۲ـ فرمانی که اعلان‌کننده‌ی خصوصیت مورد نظر برای متنِ پس از آن است، مانند {\bfseries ⟨*text*⟩}.

باید دقت کرد در حالت دوم، یعنی استفاده از فرمانِ اعلان‌کننده، شکل قلم مورد نظر برای متنی اعمال می‌شود که اولاً پس از فرمان اعلان‌کننده قرار دارد و ثانیاً با آن در یک گروه‌بندی

جای گرفته است. در اینجا منظور از یک گروه‌بندی برای دسته‌ای از اشیاء، یعنی محدود کردن آن اشیاء درون { } و یا جای دادن آنها درون یک محیط؛ بخش ۸.۱ را ببینید.

برای روشن شدن مطلب به شکل ۱۰.۱ نگاه کنید. دقت کنید، چرا در این شکل مشخصه‌ی قلم \bfseries بر سه سطر آخر اعمال شده است.

```
By a \textbf{triangle}, we mean a polygon of three sides.    \\
By a {\bfseries triangle}, we mean a polygon of three sides. \\
By a \bfseries triangle, we mean a polygon of three sides.   \\[1ex]
By a \textit{triangle}, we mean a polygon of three sides.    \\
By a {\itshape triangle}, we mean a polygon of three sides.  \\
By a \itshape triangle, we mean a polygon of three sides.
```

By a **triangle**, we mean a polygon of three sides.
By a **triangle**, we mean a polygon of three sides.
By a **triangle, we mean a polygon of three sides.**

By a *triangle*, we mean a polygon of three sides.
By a *triangle*, we mean a polygon of three sides.
By a *triangle, we mean a polygon of three sides.*

شکلِ ۱۰.۱: اهمیت گروه‌بندی هنگام انتخاب شکل قلم

و بالاخره آنکه فرمان‌های اعلان‌کننده‌ی موجود در جدولِ ۵.۱ می‌تواند به‌عنوان نام محیط نیز به‌کار رود. از این مطلب می‌توان برای اعمالِ شکلِ قلم به یک متن طولانی، که در یک یا چند پاراگراف جای گرفته است، سود جست. به‌طور مثال، برای اعمال خصوصیت Sans Serif کافی است متن مورد نظر درون محیط sffamily جای گیرد؛ شکل ۱۱.۱ را ببینید.

اگر قلمِ بکار رفته توسطِ کاربر، در رایانه‌ی وی موجود نباشد، LaTeX یک پیغام هشدار دهنده[۳۰] صادر کرده نزدیک‌ترین قلمِ موجود را جایگزین می‌کند.

اکنون می‌توان عملکرد فرمان \emph را، که در مثالِ بخشِ ۲.۳.۱ آمده است، بهتر بررسی کرد. مشاهده کردیم، اگر در متنی با خصوصیتِ قلم Shape=Upright فرمان \emph{⟨*text*⟩} ظاهر شود، خصوصیت قلم برای ⟨*text*⟩ به Shape=*Italic* تغییر می‌کند. حال اگر درون متنی با خصوصیتِ قلم Shape=*Italic* یا Shape=*Slanted* فرمان \emph{⟨*text*⟩} وارد شود، حاصل

[30] Warning

```
These declaration names can also be used as environment names:%
\begin{sffamily}
  Thus to typeset a long passage in, say, \verb|sans serif| family,
  just enclose the passage within the commands
  \verb|\begin{sffamily} ... \end{sffamily}|.
\end{sffamily}
```

These declaration names can also be used as environment names: Thus to typeset a long passage in, say, sans serif family, just enclose the passage within the commands `\begin{sffamily} ... \end{sffamily}`.

شکلِ ۱۱.۱: فرمان‌های اعلان‌کننده‌ی شکل قلم می‌تواند به‌عنوان محیط به‌کار رود.

چیست؟ شکل ۱۲.۱ پاسخ این پرسش را به‌خوبی می‌دهد.

```
A polygon of three sides is called a \emph{triangle}. \\
\textit{A polygon of three sides is called a \emph{triangle}.} \\
\textsl{A polygon of three sides is called a \emph{triangle}.} \\
\textbf{A polygon of three sides is called a \emph{triangle}.} \\
\textsc{A polygon of three sides is called a \emph{triangle}.}
```

A polygon of three sides is called a *triangle*.
A polygon of three sides is called a triangle.
A polygon of three sides is called a triangle.
A polygon of three sides is called a *triangle*.
A POLYGON OF THREE SIDES IS CALLED A *triangle*.

شکلِ ۱۲.۱: چگونگی عملکرد فرمان \emph

۲.۷.۱ اندازه‌ی قلم

به‌طور متعارف، اندازه‌ی قلم در سامانه‌های حروفچینی برحسب واحد point، با علامت اختصاری pt، بیان می‌شود. به‌طور پیش‌فرض، LaTeX اندازه‌ی 10pt را برای متن نوشتار در نظر می‌گیرد. ده فرمانِ اعلان‌کننده برای تغییر اندازه‌ی قلم در LaTeX وجود دارد که در جدول ۶.۱

گردآوری شده است. فرمان \normalsize، همان‌طور که از نام آن پیدا است، اعلان‌کننده‌ی اندازه‌ی قلمِ پیش‌فرض برای متن است. \tiny کوچکترین اندازه و \Huge بزرگترین اندازه‌ی قابل دسترسی است.

declaration	*example*
\tiny	The Quick Brown Fox Jumps Over The Lazy Dog
\scriptsize	The Quick Brown Fox Jumps Over The Lazy Dog
\footnotesize	The Quick Brown Fox Jumps Over The Lazy Dog
\small	The Quick Brown Fox Jumps Over The Lazy Dog
\normalsize	The Quick Brown Fox Jumps Over The Lazy Dog
\large	The Quick Brown Fox Jumps Over The Lazy Dog
\Large	The Quick Brown Fox Jumps Over The Lazy
\LARGE	The Quick Brown Fox Jumps Over The
\huge	The Quick Brown Fox Jumps Over
\Huge	The Quick Brown Fox Jumps

جدولِ ۶.۱: اندازه‌های مختلف قلم در LaTeX

برخلافِ فرمان‌های مربوط به شکلِ قلم در جدولِ ۵.۱، همگی فرمان‌های مربوط به اندازه‌ی قلم از نوعِ فرمان‌های اعلان‌کننده است و بنابراین باید از گروه‌بندی مناسب استفاده کرد. فرمان‌های اعلان‌کننده‌ی اندازه‌ی قلم می‌تواند به‌عنوان نام محیط نیز بکار رود؛ شکل ۱۳.۱ را نگاه کنید.

با ترکیبِ شکل قلم و اندازه‌ی قلم، می‌توان به مجموعه‌ی وسیع و گوناگونی از نوعِ قلم دست یافت. به‌عنوان مثال، به شکلِ ۱۴.۱ نگاه کنید، که در آن نمونه‌ای از جایگذاری متن همراه با استفاده از انواع مختلف قلم در LaTeX به نمایش گذاشته شده است.

بیان چند نکته درباره‌ی شکل ۱۴.۱ مفید است. همان‌گونه که مشاهده می‌شود، آنچه بین \begin{center} و \end{cener} قرار دارد، در خروجی دقیقاً وسط خط ظاهر شده است و آنچه بین \begin{flushright} و \end{flushright} جای گرفته است، در خروجی

```
These declaration names can also be used as environment names:%
\begin{footnotesize}
  Thus to typeset a long passage in, say, \verb|footnotesize| size,
  just enclose the passage within the   commands
  \verb|\begin{footnotesize} ... \end{footnotesize}|.
\end{footnotesize}
```

These declaration names can also be used as environment names: Thus to typeset a long passage in, say, `footnotesize` size, just enclose the passage within the commands `\begin{footnotesize} ... \end{footnotesize}`.

شکلِ ۱۳.۱: فرمان‌های اعلان‌کننده‌ی اندازه‌ی قلم می‌تواند به‌عنوان محیط به‌کار رود.

به منتهاالیهِ حاشیه‌ی سمتِ راست رانده شده است. به‌طور مشابه از `\begin{flushleft}` و `\end{flushleft}` برای راندن متن به حاشیه‌ی سمتِ چپ استفاده می‌شود. در این‌باره بخش‌های ۱.۴.۲ و ۲.۴.۲ را ببینید.

۳.۷.۱ قلمِ Times و قلم‌های دیگر

LaTeX به‌طور پیش‌فرض از قلم Computer Modern، با نماد اختصاری CM، برای حروفچینی متن استفاده می‌کند. این قلم توسط دونالد کنوث برای استفاده در TeX طراحی شد، زیرا TeX با هدف حروفچینی کتاب‌ها پدید آمد و Computer Modern قلمی مناسب برای حروفچینی کتاب‌ها به‌حساب می‌آید. اما Computer Modern تنها قلمی نیست که در LaTeX قابل دسترسی است. جدول ۷.۱ دربردارنده‌ی گردایه‌ای از قلم‌های گوناگون است که، به‌طور رایگان، در اختیار کاربر قرار دارد. برای اعمال برخی از این قلم‌ها بسته‌هایی فراهم شده است. به‌طور مثال برای استفاده از قلمِ Times New Roman کافی است بسته‌ی times در مقدمه‌ی فایل ورودی فراخوانی شود و برای بهره‌گیری از قلمِ Pandora کافی است بسته‌ی pandora در مقدمه‌ی فایل ورودی فراخوانی گردد. شکل‌های ۱۵.۱ و ۱۶.۱ را ببینید.

به‌طور کلی می‌توان به همه‌ی قلم‌های گردآوری شده در جدولِ۷.۱ دست پیدا کرد. این کار با کمک فرمان‌های `\fontfamily` و `\selectfont` به صورت زیر انجام می‌شود.

```
\begin{center}
    {\bfseries \huge The \TeX nical Institute}\\[.75cm]
    {\scshape\LARGE Certificate}
\end{center}
  \noindent This is to certify that Mr.~N.~O.~Vice has undergone a course
  at this institute and is qualified to be a \TeX nician.
\begin{flushright}
    \sffamily  The Director\\ The \TeX nical Institute
\end{flushright}
```

The TEXnical Institute

Certificate

This is to certify that Mr. N. O. Vice has undergone a course at this institute and is qualified to be a TEXnician.

The Director
The TEXnical Institute

شکلِ ۱۴.۱: نمونه‌ای از جایگذاری متن همراه با استفاده از انواع مختلف قلم

font name	*font family*	package	*example*
Computer Modern Roman	`cmr`		The Quick Brown Fox Jumps Over The Lazy Dog
Computer Modern Sans	`cmss`		The Quick Brown Fox Jumps Over The Lazy Dog
Computer Modern Typewriter	`cmtt`		`The Quick Brown Fox Jumps Over The Lazy Dog`
Pandora	`panr`	pandora	The Quick Brown Fox Jumps Over The Lazy Dog
Pandora Sans	`pss`		The Quick Brown Fox Jumps Over The Lazy Dog
Pandora Typewriter	`pntt`		`The Quick Brown Fox Jumps Over The Lazy Dog`
Universal	`uni`		The Quick Brown Fox Jumps Over The ...
Concrete	`ccr`	ccr	The Quick Brown Fox Jumps Over The Lazy Dog
Avant Grade	`pag`	avant	The Quick Brown Fox Jumps Over The Lazy Dog
Bookman	`pbk`	bookman	The Quick Brown Fox Jumps Over The Lazy Dog
Courier	`pcr`	courier	`The Quick Brown Fox Jumps Over The ...`
Helvetica	`phv`	helvet	The Quick Brown Fox Jumps Over The Lazy Dog
New Century Schoolbook	`pnc`	newcent	The Quick Brown Fox Jumps Over The Lazy Dog
Palatino	`ppl`	palatino	The Quick Brown Fox Jumps Over The Lazy Dog
Times New Roman	`ptm`	times	The Quick Brown Fox Jumps Over The Lazy Dog
Zapf Chancery	`pzc`	zapfchan	*The Quick Brown Fox Jumps Over The Lazy Dog*
Charter	`bch`	charter	The Quick Brown Fox Jumps Over The Lazy Dog
URW Antiqua	`uaq`		The Quick Brown Fox Jumps Over The Lazy Dog
URW Grotesk	`ugq`		**The Quick Brown Fox Jumps Over The Lazy Dog**
Utopia	`put`	utopia	The Quick Brown Fox Jumps Over The Lazy Dog

جدولِ ۷.۱: برخی قلم‌های قابل دسترسی در LaTeX

```
% in preamble: \usepackage{times}
        The Quick Brown Fox Jumps Over The Lazy Dog.  \\[.5ex]
\textit{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsl{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsc{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[1ex]
\textbf{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[1ex]
\texttt{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsf{The Quick Brown Fox Jumps Over The Lazy Dog.}
```

The Quick Brown Fox Jumps Over The Lazy Dog.
The Quick Brown Fox Jumps Over The Lazy Dog.
The Quick Brown Fox Jumps Over The Lazy Dog.
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
The Quick Brown Fox Jumps Over The Lazy Dog.
`The Quick Brown Fox Jumps Over The Lazy Dog.`
The Quick Brown Fox Jumps Over The Lazy Dog.

شکلِ ۱۵.۱: نمونه‌ای از به‌کارگیری قلمِ Times

```
% in preamble: \usepackage{bookman}
        The Quick Brown Fox Jumps Over The Lazy Dog.  \\[.5ex]
\textit{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsl{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsc{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[1ex]
\textbf{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[1ex]
\texttt{The Quick Brown Fox Jumps Over The Lazy Dog.} \\[.5ex]
\textsf{The Quick Brown Fox Jumps Over The Lazy Dog.}
```

The Quick Brown Fox Jumps Over The Lazy Dog.
The Quick Brown Fox Jumps Over The Lazy Dog.
The Quick Brown Fox Jumps Over The Lazy Dog.
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
The Quick Brown Fox Jumps Over The Lazy Dog.
`The Quick Brown Fox Jumps Over The Lazy Dog.`
The Quick Brown Fox Jumps Over The Lazy Dog.

شکلِ ۱۶.۱: نمونه‌ای از به‌کارگیری قلمِ Bookman

```
{\fontfamily{⟨font-family⟩}\selectfont ⟨text⟩}
```

در اینجا ⟨*font-family*⟩ نامی است برای قلمِ مورد نظر، که در جدول ۷.۱ زیر ستون *font* *family* آمده است. اگر بخواهیم انتخاب قلم به‌طور موضعی رخ دهد باید متن مورد نظر و فرمان‌های مذکور، به‌طور مناسب، در یک گروه‌بندی قرار گیرد. درباره‌ی گروه‌بندی بخش ۸.۱ را ببینید.

```
{\fontfamily{phv} \selectfont Helvetica looks like this.} \hfill
{\fontfamily{bch} \selectfont Charter looks like this.} \hfill
{\fontfamily{pag} \selectfont Avant Grade looks like this.}
```

Helvetica looks like this. Charter looks like this. Avant Grade looks like this.

۸.۱ گروه‌بندی و اهمیت آن

همان‌گونه که در جدول ۱.۱ در صفحه‌ی ۲۱ ملاحظه می‌شود، کراکترهای { و } از گونه‌ی کراکترهای مخصوص به‌حساب می‌آید. با کمی دقت در فایل ورودی مشاهده می‌شود هرجا محدوده‌ای با کراکتر } آغاز شده است اندکی جلوتر آن محدوده با کراکتر { پایان یافته است و همواره کراکتر { مکمل کراکتر } است، جز در مواردی که \} یا \{ وارد شده باشد که در این دو حالت کراکترهای { و } نقش محدوده‌سازی خود را از دست می‌دهند.

به‌عنوان نمونه، آنجا که قرار است، مثلاً، فرمان \LaTeXCommand عملی را روی اشیاء ⟨*objects*⟩ انجام دهد، این اشیاء را درون { } محدود می‌کنیم؛ {⟨*objects*⟩}\LaTeXCommand. اهمیت محدود کردن اشیاء ⟨*object*⟩ درون { } به‌ویژه هنگامی خودنمایی می‌کند که تعداد این اشیاء دو یا بیشتر از دو باشد. به مثال زیر دقت کنید.

```
This is my \emph{first} document.\\
This is my \emph first document. \\
This is my \textbf first document.\\
This is my \textbf{first} document.
```

This is my *first* document.
This is my *f*irst document.
This is my **f**irst document.
This is my **first** document.

حال به نمونه‌ای دیگر از کاربرد کراکترهای { و } می‌پردازیم. در LaTeX بسیاری از فرمان‌ها از گونه‌ی فرمان‌های اعلان‌کننده است و غالباً تأثیر فرمان‌های اعلان‌کننده به‌طور موضعی است. نمونه‌ای از فرمان‌های اعلان‌کننده، فرمان‌های مربوط به شکل قلم و اندازه‌ی قلم می‌باشد، که در جدول ۵.۱ در صفحه‌ی ۵۹ و در جدول ۶.۱ در صفحه‌ی ۶۲، گردآوری شده است.

قاعده‌ی کلی آن است که چنانچه بخواهیم کاری که یک فرمان اعلان‌کننده انجام می‌دهد تنها در یک ناحیه‌ی معین معتبر باشد، آن ناحیه را با کراکترهای { و } محدود می‌کنیم. مثال زیر این موضوع را روشن می‌کند.

```
This is my {\itshape first} document

This is my \itshape first document \\
This is my {\large first} document
```

This is my *first* document
This is my *first document*
This is my first document

و بالاخره به کاربرد کراکترهای { و } هنگام حروفچینی عبارت‌های ریاضی می‌پردازیم. در بخش ۴.۳ در صفحه‌ی ۱۵۵ قواعد فرمول‌نویسی تشریح شده است. حال به مثال زیر دقت کنید که چگونه محدود کردن اشیاء توسط کراکترهای { و } بر نتیجه‌ی کار اثر می‌گذارد.

```
\[
  a^bc,\quad  a^{bc},\quad  \frac123,\quad \frac1{23},\quad \frac{12}3,\quad
  \int_2\pi^3\pi f,\quad \int_{2\pi}^{3\pi} f,\quad
  \lim_x\to\infty f(x),\quad \lim_{x\to\infty} f(x)
\]
```

$$a^bc,\quad a^{bc},\quad \frac123,\quad \frac1{23},\quad \frac{12}3,\quad \int_2\pi^3\pi f,\quad \int_{2\pi}^{3\pi} f,\quad \lim_x\to\infty f(x),\quad \lim_{x\to\infty} f(x)$$

یکی دیگر از ساختارهای LaTeX که ایجاد محدوده می‌کند، محیط‌ها هستند. چنانچه یک فرمانِ اعلان‌کننده درون محیطی صادر شود، خارج از آن محیط بی‌اثر است. به مثال زیر دقت کنید.

```
Permission is granted to copy and distribute modified versions of this manual
\begin{itshape}%
  under the conditions for \bfseries verbatim copying,
\end{itshape}%
provided that the entire resulting derived work is distributed
under the terms of a permission notice identical to this one.
```

Permission is granted to copy and distribute modified versions of this manual *under the conditions for* ***verbatim copying,*** provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

قرارداد. اکنون قرارداد می‌کنیم منظور از یک **گروه‌بندی**[۳۱] یعنی گردایه‌ای از اشیاء که در محدوده‌ی ایجاد شده توسط کراکترهای { و } و یا در محدوده‌ی پدید آمده به‌وسیله‌ی یک محیط، جای گرفته است. گاهی ممکن است چند لایه‌ی تودرتو گروه‌بندی داشته باشیم، مانند

```
{A {B {C D}}}
```

در این مثال، سه لایه تودرتو گروه‌بندی وجود دارد که، از خارج به داخل، اشیاء `A, B, C, D` همه در گروه‌بندی لایه‌ی اول، اشیاء `B, C, D` در گروه‌بندی لایه‌ی دوم و اشیاء `C, D` در گروه‌بندی لایه‌ی سوم قرار دارند.

۹.۱ سامانه‌ی برچسب‌گذاری و ارجاع‌دهی

معمولاً در نوشتجاتی که در یکی از رده‌های نوشتاری مقاله، پایان‌نامه یا کتاب جای می‌گیرند، عمل ارجاع‌دهی گریز ناپذیر است. برای ارجاع دادن خواننده به یک فصل یا یک بخش، شماره‌ی آن ذکر می‌گردد و برای ارجاع دادن خواننده به یک معادله، ابتدا آن معادله شماره‌گذاری شده سپس شماره‌ی آن ذکر می‌شود. گاهی لازم است خواننده را به مطلبی که در یک صفحه‌ی معین جای گرفته است، ارجاع داد. در این حالت شماره‌ی آن صفحه ذکر می‌گردد.

اما فرایند ارجاع‌دهی در LaTeX چگونه است؟ آیا لازم است کاربر شماره‌ی بخش یا معادله

[31] Grouping

یا صفحه‌ی مورد نظر را به‌خاطر سپارد و آن را در محل قرار دهد؟ اگر این‌گونه باشد آنگاه کار ارجاع‌دهی بسیار سخت خواهد بود؛ زیرا در هربار ویرایش نوشتار، ممکن است مطالبی به آن اضافه گردد و بخش‌هایی از آن کاسته شود. این عوامل باعث می‌شود شماره‌ها تغییر کند. تصحیح شماره‌ها، به‌طور دستی، برای یک نوشتار طولانی، مانند یک کتاب یا یک پایان‌نامه، کاری طاقت‌فرسا است.

خوشبختانه، با بکارگیری سامانه‌ی برچسب‌گذاری LaTeX، کار ارجاع‌دهی بسیار آسان گشته است. قاعده‌ی کلی این‌گونه است که:

> ابتدا شیء شماره‌گذاری شده، اعم از یک فصل، یک بخش، یک معادله یا یک قضیه، به‌وسیله‌ی یک برچسب[۳۲] نشانه‌گذاری می‌شود. سپس، به‌جای ذکر شماره‌ی آن شیء به‌طور مستقیم، از برچسب آن استفاده می‌شود. نکته‌ی مهم آن است که، سامانه‌ی برچسب‌گذاری LaTeX، منحصراً، برای اشیائی که به‌طور خودکار توسط LaTeX شماره‌گذاری می‌شوند، قابل بکارگیری است.

در LaTeX اشیائی که به کمک سامانه‌ی برچسب‌گذاری، قابل ارجاع‌دهی هستند، به سه دسته تقسیم می‌شوند:

دسته‌ی اول اشیائی نظیر فصل‌ها و بخش‌ها، جدول‌ها و شکل‌ها، قضیه‌ها و شبه‌ـقضیه‌ها،[۳۳] معادله‌ها و حتی گزینه‌های یک فهرست شمارشی،

دسته‌ی دوم همه‌ی صفحات شماره‌دار،

دسته‌ی سوم گزینه‌ها و مدخل‌های فهرست منابع یا کتابنامه.

ارجاع‌دهی به اشیاء دسته‌ی اول. برای ارجاع‌دهی به یک شیء از دسته‌ی اول، ابتدا آن شیء را به‌وسیله‌ی یک برچسب نشانه‌گذاری می‌کنیم. عمل نشانه‌گذاری با فرمان `\label{`⟨*label*⟩`}` انجام می‌شود. در اینجا، ⟨*label*⟩ برچسبی است که، به‌دلخواه، توسط کاربر اختیار می‌گردد. درباره‌ی برچسب ⟨*label*⟩، دانستن مواردی که درپی می‌آید، سودمند است.

---

[32] Label

[۳۳] منظور از شبه‌ـقضیه‌ها ساختارهایی چون تعریف‌ها، لم‌ها و گزاره‌ها هستند.

- برچسب ⟨*label*⟩ می‌تواند ترکیبی از حروف، اعداد، نشانه‌های نقطه‌گذاری و حتی عبارت‌های ریاضی باشد، با این شرط که از قواعد حروفچینی LaTeX پیروی کند.
- بهتر است برچسب ⟨*label*⟩ به‌گونه‌ای اختیار شود که به‌یاد آوردن آن آسان‌تر باشد. به‌طور مثال، برای فصلی با عنوان Introduction عبارت `chap:intro` و برای بخشی با عنوان Introduction عبارت `sec:intro` برچسب‌های مناسبی به‌نظر می‌رسد.
- باید دقت کرد، LaTeX نسبت به حروف کوچک و بزرگ و همچنین فضاهای خالی، در ⟨*label*⟩، حساس است. به‌طور مثال، عبارت‌های `sec:intro`، `Sec:intro` و `sec: intro` سه برچسب متفاوت به‌حساب می‌آید.
- بهتر است هنگام نشانه‌گذاری هریک از اشیاء، پیشوندی مناسب برای برچسب آن اختیار شود. به‌عنوان نمونه، پیشوند `chap:` برای فصل‌ها، پیشوند `sec:` برای بخش‌ها، پیشوند `eqn:` برای معادله‌ها، پیشوند `thm:` برای قضیه‌ها و پیشوند `fig:` برای شکل‌ها.

پس از آنکه یک شیء شماره‌گذاری شده از دسته‌ی اول، به‌وسیله‌ی فرمان `\label{`⟨*label*⟩`}` برچسب‌گذاری شد، می‌توان در هرجای متن، با کمک فرمان `\ref{`⟨*label*⟩`}`، به شماره‌ی آن شیء دسترسی پیدا کرد. در طول این کتاب خواهیم دید، بهترین مکان برای فرمان `\label`، برای نشانه‌گذاری یک شیء شماره‌دار، کجا است. برای روشن شدن موضوع، به یک مثال کلی، که در شکل ۱۷.۱ نمایش داده شده است، نگاه کنید.

ارجاع‌دهی به صفحات. گاهی، جهت کمک به خواننده برای پیدا کردن یک مطلب، شماره‌ی صفحه را ذکر می‌کنیم. با فرمان `\pageref{`⟨*label*⟩`}` می‌توان به شماره‌ی صفحه‌ای که برچسب ⟨*label*⟩ در آن جای گرفته است، دسترسی پیدا کرد.

ارجاع‌دهی به گزینه‌های فهرست منابع. در این‌باره، به‌طور مفصل، در بخش‌های ۱.۶.۴ و ۲.۶.۴ شرح داده شده است.

مثال. شکل ۱۷.۱ نمایش دهنده‌ی یک مثال کلی می‌باشد، که در آن انواع مختلف ارجاع‌دهی جای گرفته است؛ به محل جای‌گیری فرمان `\label` دقت کنید. در این شکل، از فرمان `\eqref` برای ارجاع‌دهی به معادله‌ها استفاده شده است. این فرمان توسط بسته‌ی amsmath فراهم شده است. در واقع فرمان `\eqref{`⟨*label*⟩`}` هم‌ارز با فرمان `(\ref{`⟨*label*⟩`})` است.

چگونگی حروفچینی معادله‌های شماره‌دار در بخش ۱۰.۳ و چگونگی حروفچینی قضیه‌ها و شبه‌قضیه‌ها در بخش ۱۴.۳ تشریح شده است.

۱۰.۱ جملات توضیحی در فایل ورودی

وقتی LaTeX به کراکتر % برخورد کند، علامت % و آنچه پس از آن تا پایان همان خط جای دارد، از سوی LaTeX نادیده گرفته می‌شود و به‌علاوه فضای خالی ابتدای خط بعد حذف می‌گردد. از این موضوع می‌توان در موارد زیر بهره گرفت:

۱ـ برای وارد کردن توضیحاتی در فایل ورودی LaTeX به‌طوری که در خروجی ظاهر نشود،

۲ـ برای حذف فاصله‌های ناخواسته که از یک خط به خط بعدی ایجاد می‌شود،

۳ـ برای شکستن خطوط طولانی به خطوط کوتاه‌تر در فایل ورودی، آنجا که این کار در حالت عادی امکان‌پذیر نیست.

```
This is an % stupid
% Better: instructive <---
example:\\
Supercal%
         ifragilist%
        iceexpialidocious\\
Hello%
world\\
Hello
World
```

This is an example:
Supercalifragilisticeexpialidocious
Helloworld
Hello World

```
\section{Analytic Functions}\label{sec:AF}
  If the development in this section has familiar ring, it should.
\begin{dfn}\label{dfn:differentiable}
  A complex-valued function $f(z)$ is \emph{differentiable} at $z_0$
  if the difference quotients
  \begin{equation}\label{eqn:derivative}
    \frac{f(z)-f(z_0)}{z-z_0}
  \end{equation}
  have a limit as $z\to z_0$.
\end{dfn}

\subsection{Chain Rule}\label{subsec:chain rule}

\begin{thm}\label{thm:hain rule}
  Suppose that $g(z)$ is differentiable at $z_0$, and suppose that
  $f(w)$ is differentiable at $w_0=g(z_0)$. Then \ldots
\end{thm}

\subsection{Cross Reference}

See section~\ref{sec:AF} and subsection~\ref{subsec:chain rule}\\
Definition~\ref{dfn:differentiable} and Theorem~\ref{thm:hain rule}\\
Equation~\eqref{eqn:derivative}
```

1 Analytic Functions

If the development in this section has familiar ring, it should.

Definition 1.1. A complex-valued function $f(z)$ is *differentiable* at z_0 if the difference quotients

$$\frac{f(z)-f(z_0)}{z-z_0} \tag{1.1}$$

have a limit as $z \to z_0$.

1.1 Chain Rule

Theorem 1.2. *Suppose that $g(z)$ is differentiable at z_0, and suppose that $f(w)$ is differentiable at $w_0 = g(z_0)$. Then ...*

1.2 Cross Reference

See section 1 and subsection 1.1
Definition 1.1 and Theorem 1.2
Equation (1.1)

شکلِ ۱۷.۱: استفاده از سامانه‌ی برچسب‌گذاری LaTeX برای ارجاع‌دهی

برای وارد کردن توضیحات طولانی بهتر است از محیط comment استفاده کرد. محیط comment توسط بسته‌ی verbatim، [۳۳]، فراهم شده است، به این معنی که برای استفاده از آن ضروری است بسته‌ی verbatim فراخوانی شود و این کار با وارد کردن فرمان \usepackage{verbatim} در مقدمه‌ی فایلِ ورودی، یعنی پس از فرمان \documentclass و پیش از فرمان \begin{document}، انجام می‌شود.

```
This is another \begin{comment}
                  rather stupid, but helpful
                \end{comment}
example for embedding comments in your document.
```

This is another example for embedding comments in your document.

توجه شود، این شیوه درون محیط‌های پیچیده‌ای چون محیط‌های فرمول‌نویسی ریاضی کارآمد نیست.

نکته‌ی دیگر درباره‌ی محیط comment آن است که، نمی‌توان از آن به صورت تودرتو استفاده نمود. به‌طور مثال، فایل ورودی زیر را در نظر بگیرید.

```
\documentclass{articel}
\begin{document}
  Some text ...
  \begin{comment}
    Commented out text ...
    \begin{comment}
      Some more commented out text ...
    \end{comment}
    and some more commented out text ...
  \end{comment}
\end{document}
```

پس از اجرا کردن LaTeX روی فایل ورودی بالا، از سوی LaTeX پیغام خطای

```
LaTeX Error: \begin{document} ended by \end{comment}
```

صادر می‌گردد. کجا خطا رخ داده است؟

۱۱.۱ حروفچینی لفظ به لفظ

هرچند ممکن است، در فایل ورودی، کلمات با فاصله‌های گوناگون از هم قرار داشته باشند و خطوط با طول متفاوت ظاهر شده باشند اما، در خروجی، متن به‌طور منظم و مرتب در پاراگراف‌ها حروفچینی می‌شود و فاصله‌ی بین کلمات و طول خطوط، از سوی LaTeX، به بهترین شیوه انتخاب می‌گردد.

```
There is a   \texttt{verbatim} environment. You    may need it if  you
write \emph{about} \LaTeX{} or   some other computer  program
or if you have to
include   portions of a source  file or user   input.
```

There is a `verbatim` environment. You may need it if you write *about* LaTeX or some other computer program or if you have to include portions of a source file or user input.

اما، گاهی نیاز است حاصل کار در خروجی عیناً همان باشد که در ورودی ظاهر شده است و آرایش کلمات و خطوط تغییر نکند، که ما به این وضعیت **حروفچینی لفظ به لفظ**[۳۴] گوییم. این وضع بیشتر در نوشتجاتی رخ می‌دهد که در آن ورودی برنامه‌های رایانه‌ای حروفچینی می‌گردد، مانند همین کتابی که پیش رو دارید.

در LaTeX، برای حروفچینی لفظ به لفظ، محیط `verbatim` فراهم شده است.

```
\begin{verbatim}
There is a   \texttt{verbatim} environment. You    may need it if  you
write \emph{about} \LaTeX{} or   some other computer  program
or if you have to
include   portions of a source  file or user   input.
\end{verbatim}
```

```
There is a   \texttt{verbatim} environment. You    may need it if  you
write \emph{about} \LaTeX{} or   some other computer  program
or if you have to
include   portions of a source  file or user   input.
```

ملاحظه می‌شود، آنچه درون محیط `verbatim` جای گرفته است، عیناً به همان شکل و همان آرایش در خروجی ظاهر شده است.

[34] Verbatim

نکته‌ی مهم آن است که نمی‌توان از محیط verbatim به صورت تودرتو بهره گرفت. به‌طور مثال، فایل ورودی زیر را در نظر بگیرید.

```
\documentclass{article}
\begin{document}
  Some text ...
  \begin{verbatim}
    Verbatim text
    \begin{verbatim}
      Some more verbatim text ...
    \end{verbatim}
    and some more verbatim text ...
  \end{verbatim}
\end{document}
```

پس از اجرا کردن LaTeX روی فایل ورودی بالا، از سوی LaTeX پیغام خطای

```
LaTeX Error: \begin{document} ended by \end{verbatim}
```

صادر می‌گردد. کجا خطا رخ داده است؟

حروفچینی لفظ به لفظ به صورت درون‌ـخطی. حروفچینی لفظ به لفظ برای عبارتی که از یک خط تجاوز نمی‌کند، به صورت درون‌ـخطی، امکان‌پذیر است. این کار با استفاده از فرمان \verb انجام می‌شود.

```
\verb|$ \sin^2 x + \cos^2 x = 1 $|
```

```
$ \sin^2 x + \cos^2 x = 1 $
```

دقت کنید، آرگومان فرمان \verb بین دو علامت | | محدود شده است و به‌علاوه نمی‌توان آرگومان فرمان \verb را در دو خط یا بیشتر از دو خط وارد کرد.

حال اگر عبارتی که در آرگومان فرمان \verb قرار دارد خود شامل کراکتر | باشد، دیگر نمی‌توان آن را بین علامت‌های | | محدود کرد. به‌عنوان مثال، با صادر کردن فرمان

```
\verb|$ |\sin x| \leq |x| $|
```

از سوی LaTeX پیغام خطا صادر می‌گردد؛

```
Missing $ inserted
```

در این حالت، می‌توان از کراکترهای دیگری چون + و - به‌عنوان محدودکننده بهره گرفت.

```
\verb+ |\sin(x)| \leq |x| + \\
\verb- |x| + |y| = 1 - \\
\verb@ |x| + |y| - |z| @ \\
\verb|   @|
```

```
|\sin(x)| \leq |x|
|x| + |y| = 1
|x| + |y| - |z|
  @
```