نامه TEX

وفا خلیقی

۲۶ مهر ۱۳۹۳ نسخه ۰/۱

فهرست مطالب

فهرست جداول

فهرست تصاویر

پیش‌گفتار

در این نوشتار، سیستم حروف‌چینی TEX بصورت کامل توضیح داده شده است. این نوشتار با استفاده از مثال‌ها، و پرسش و پاسخ‌های متعدد سعی می‌کند تا مطالب را بصورت ساده بیان کند تا درک آن‌ها آسان باشد. برای نوشتن این نوشتار از [۱] استفاده شده است. این نوشتار با استفاده از نرم‌افزار XEPersian حروف‌چینی شده است.

وفا خلیقی

فروردین ۱۳۹۴

persian-tex@tug.org

فصل ۱

TEX چیست؟

TEX یک سیستم حروف‌چینی است که توسط Donald Knuth برای حروف‌چینی کتاب‌های زیبا، به ویژه کتاب‌های که مقدار زیادی ریاضیات دارند، اختراع شده است. TEX نشان تجاری انجمن ریاضی آمریکا است.

حروف TEX حروف بزرگ یونانی $\tau\epsilon\chi$ (به ترتیب) هستند. حروف $\tau\epsilon\chi$ سه حرف اول واژه‌ای یونانی به معنای هنر و تکنولوژی هستند. TEX بصورت «تِخْ» یا «تِکْ» تلفظ می‌شود. در فایل‌های کامپیوتری و جاهایی که نمی‌توان لوگوی TEX را به درستی نوشت (E را پائینتر قرار داد)، می‌توان TEX را بصورت `TeX` نوشت.

فصل ۲

تایپ در مقایسه با حروف‌چینی

۱.۲ متن ورودی TEX

در این نوشتار متن ورودی TEX با قلم تایپ ظاهر می‌شود تا تفاوت بین متن ورودی و متن خروجی TEX قابل تشخیص باشد. کاراکترهایی که متن ورودی TEX را تشکیل می‌دهند عبارتند از:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789"#$%&@*+-=,.:;?!
()<>[]{}`'\||/_^~
```

همچنین در این نوشتار از علامت ␣ برای نشان دادن یک فاصله خالی استفاده می‌کنیم.

مثال ۱.۲. یک نمونه از کابرد علامت فاصله در جدول ۱.۲ نشان داده شده است.

ورودی (بدون علامت فاصله)	ورودی (با علامت فاصله)	خروجی
`I understand.`	`I␣understand.`	I understand.

جدول ۱.۲ نمونه‌ای از کاربرد علامت فاصله

خروجی	ورودی
‘I understand.’	`I understand.'
‘I understand.’	\lq␣I understand.\rq␣

جدول ۲.۲ نمونه‌ای از کاربرد علامت نقل‌قول تکی

خروجی	ورودی
“I understand.”	``I understand.''
“I understand.”	\lq\lq␣I understand.\rq\rq␣

جدول ۳.۲ نمونه‌ای از کاربرد علامت نقل‌قول دوتایی

۲.۲ علامت نقل‌قول

۱.۲.۲ علامت نقل‌قول تکی

برای حروف‌چینی علامت نقل‌قول چپ تکی (‘)، از ` استفاده می‌کنیم. همچنین برای حروف‌چینی علامت نقل‌قول چپ تکی (‘)، از دستور \lq می‌توانیم استفاده می‌کنیم.

برای حروف‌چینی علامت نقل‌قول راست تکی (’)، از ' استفاده می‌کنیم. همچنین برای حروف‌چینی علامت نقل‌قول راست تکی (’)، از دستور \rq می‌توانیم استفاده می‌کنیم.

اگر کاراکتر بعد از دستورات \lq و \rq یک حرف باشد، بعد از این دستورات یک فاصله قرار می‌دهیم اما اگر کاراکتر بعد از دستورات \lq و \rq یک فاصله باشد، بعد از این دستورات یک دستور فاصله (\␣) قرار می‌دهیم.

مثال ۲.۲. یک نمونه از کاربرد علامت نقل‌قول تکی در جدول ۲.۲ نشان داده شده است.

۲.۲.۲ علامت نقل‌قول دوتایی

برای حروف‌چینی علامت نقل‌قول چپ دوتایی (“)، از دو علامت نقل‌قول چپ تکی (`` یا \lq\lq) استفاده می‌کنیم. برای حروف‌چینی علامت نقل‌قول راست دوتایی (”)، از دو علامت نقل‌قول راست تکی ('' یا \rq\rq) استفاده می‌کنیم.

مثال ۳.۲. یک نمونه از کاربرد علامت نقل‌قول دوتایی در جدول ۳.۲ نشان داده شده است.

روش	ورودی	خروجی	مشکل	علت
۱	'''	”’	یک علامت نقل‌قول راست دوتایی و سپس یک علامت نقل‌قول راست تکی می‌گیریم.	TEX دو علامت نقل‌قول راست تکی اولی را به عنوان یک علامت نقل‌قول راست دوتایی و علامت نقل‌قول راست تکی آخری را به عنوان یک علامت نقل‌قول راست تکی ترجمه می‌کند.
۲	{'}''	”’	یک علامت نقل‌قول راست تکی و سپس یک علامت نقل‌قول راست دوتایی (اگر به دقت نگاه کنیم) اما همانند سه علامت نقل‌قول راست تکی با فواصل یکسان به نظر می‌رسد.	فاصله بعد از یک علامت نقل‌قول راست تکی کمتر از فاصله بعد از یک علامت نقل‌قول راست دوتایی است.
۳	''␣'	” ’	فاصله خیلی بزرگ (اندازه فاصله بین کلمات) است و TEX هنگامی که این فاصله را در هنگام ساختن پاراگراف می‌بیند، ممکن است حتی یک سطر جدید را شروع کند.	

جدول ۴.۲ راه‌حل‌های ممکن برای حروف‌چینی علامت نقل‌قول تو در تو در پرسش ۱.۲

۳.۲.۲ فاصله بعد از علامت نقل‌قول در قلم‌های Computer Modern

فاصله بعد از یک علامت نقل‌قول چپ دوتایی کمتر از فاصله بعد از یک علامت نقل‌قول چپ تکی است اما فاصله بعد از یک علامت نقل‌قول راست تکی کمتر از فاصله بعد از یک علامت نقل‌قول راست دوتایی است. بنابراین چپ و راست برعکس هم هستند.

۴.۲.۲ حروف‌چینی علامت‌های نقل‌قول تو در تو

پرسش ۱.۲. چگونه می‌توانیم یک علامت نقل‌قول راست تکی و سپس بلافاصله یک علامت نقل‌قول راست دوتایی (”’) داشته باشیم؟

پاسخ. پاسخ‌هایی که ممکن است به ذهن شما خطور کنند، در جدول ۴.۲ بیان شده‌اند.

خروجی	ورودی
’”	'\thinspace''

جدول ۵.۲ راه‌حل حروف‌چینی علامت نقل‌قول تو در تو در پرسش ۱.۲

خروجی	ورودی
‘“	`{}``
“‘	``\thinspace`
’”	'\thinspace''
”’	'''

جدول ۶.۲ نحوه حروف‌چینی انواع علامت‌های نقل‌قول تو در تو

راه‌حل واقعی پرسش ۱.۲ در جدول ۵.۲ نشان داده شده است.

نحوه حروف‌چینی انواع علامت‌های نقل‌قول تو در تو در جدول ۶.۲ نشان داده شده است.

۳.۲ خطوط تیره

انواع خطوط تیره در جدول ۷.۲ نشان داده شده‌اند.

مثال ۴.۲. یک نمونه از کاربرد خطوط تیره در جدول ۸.۲ نشان داده شده است.

پرسش ۲.۲. هنگامی که چهار hyphen را پشت سر هم تایپ می‌کنیم، چه اتفاقی می‌افتد؟

پاسخ. همانطور که در جدول ۹.۲ مشاهده می‌کنید، یک em-dash و یک hyphen می‌گیریم.

۴.۲ لیگاتورها و درهم‌رفتگی (Kerning)

لیگاتور به ترکیب مشخصی از حروف (`ff`، `fi`، `fl`، `ffi`، و `ffl`) گفته می‌شود به عنوان یک واحد در نظر گرفته می‌شوند.

مثال ۵.۲. برای اینکه بتوانید تفاوت کلمات بدون لیگاتور و با لیگاتور را درک کنید، چند نمونه در جدول ۱۰.۲ نشان داده شده است.

نام	ورودی	خروجی	کاربرد
Hyphen	-	-	برای کلمات مرکب مانند daughter-in-law یا X-rated استفاده می‌شود.
En-dash	--	–	برای محدوده اعداد مانند pages 13–34 یا جاهایی مانند Exercise 1.2.6–52 به کار می‌رود.
Em-dash	---	—	برای نقطه‌گذاری در جملات به کار می‌روند—که به طور ساده به آن‌ها dash می‌گوییم.
علامت منها	$-$	−	در فرمول‌ها به کار می‌رود.

جدول ۷.۲ انواع خطوط تیره

ورودی	خروجی
`Alice said, ``I always use an en-dash instead of a hyphen when specifying page numbers like `480--491' in a bibliography.''`	Alice said, “I always use an en-dash instead of a hyphen when specifying page numbers like ‘480–491’ in a bibliography.”

جدول ۸.۲ نمونه‌ای از کاربرد خطوط تیره

ورودی	خروجی
----	—-

جدول ۹.۲ چهار hyphen پشت سر هم

بدون لیگاتور	با لیگاتور
find	find
fluffier	fluffier
firefly	firefly
fisticuffs	fisticuffs
flagstaff	flagstaff
fireproofing	fireproofing
chiffchaff	chiffchaff
riffraff	riffraff

جدول ۱۰.۲ چند کلمه بدون لیگاتور و با لیگاتور

ترکیب حروف مجاور (مانند `A` بعد از `V`) که برای ظاهر بهتر باید به هم نزدیک شوند که درهم‌رفتگی (kerning) خوانده می‌شود.

TEX بصورت خودکار لیگاتور و درهم‌رفتگی را انجام می‌دهد بنابراین نیازی نیست نگران آن‌ها باشید.

فصل ۳

کنترل کردن TEX

۱.۳ کاراکتر گریز (Escape Character)

صفحه کلید در مقایسه با نشانه‌های زیادی که ما می‌خواهیم بنویسیم، کلیدهای خیلی کمی دارد. برای اینکه بتوانیم این محدودیت صفحه کلید را جبران کنیم، یکی از کاراکترهایی را که می‌خواهیم تایپ کنیم، برای منظور خاصی حفظ می‌کنیم. به این کاراکتر، کاراکتر گریز (escape character) می‌گوییم. هر زمانی که می‌خواهیم چیزی را تایپ کنیم که قالب نوشته‌مان را کنترل می‌کند، یا چیزی که از صفحه کلید بصورت معمولی استفاده نمی‌کند، باید این کاراکتر گریز را تایپ کنیم و سپس بلافاصله به کاری که می‌خواهیم انجام دهیم، اشاره کنیم.

معمولاً کلیدی در سمت چپ، بالای صفحه کلید وجود دارد که برچسب ESC دارد. توجه داشته باشید که این کلید، کاراکتر گریز نیست. این کلید، پیغام خاصی را به سیستم عامل می‌دهد بنابراین این کلید را با کاراکتر گریز اشتباه نگیرید.

TEX به ما اجازه می‌دهد تا از هر کاراکتری به عنوان کاراکتر گریز استفاده کنیم اما معمولاً کاراکتر خط اریب وارو یا همان backslash (\) برای این منظور استفاده می‌شود. علت این امر این است که خطهای اریب وارو هم برای تایپ کردن راحت هستند و هم به ندرت در متن عادی استفاده می‌شوند.

۲.۳ دستورات

بلافاصله بعد از تایپ \ (بلافاصله بعد از یک خط اریب وارو)، یک دستور کد شده تایپ می‌کنیم تا به TEX بگوییم چه چیزی در نظر داریم. به اینگونه دستورات، به اصطلاح، control sequences گفته می‌شود.

مثال ۱.۳. به عنوان مثال، ممکن است تایپ کنیم:

```
\input MS
```

که باعث می‌شود TEX شروع به خواندن فایل MS.tex کند. رشته کاراکترهای \input، یک control sequence است.

مثال ۲.۳. زمانی که تایپ می‌کنیم:

```
George P\'olya and Gabor Szeg\"o.
```

TEX آن را به متن زیر تبدیل می‌کند:

George Pólya and Gabor Szegö.

در این مثال، دو control sequence وجود دارد: \' و \" . از این control sequenceها برای قرار دادن اعراب (accents) روی تعدادی از حروف، استفاده می‌کنیم.

۱.۲.۳ انواع control sequenceها

دو نوع control sequence وجود دارد:

- نوع اول، مانند \input، یک control word نامیده می‌شود. یک control word (به ترتیب) از یک escape character، یک یا چند حرف، یک فاصله یا چیزی بعد از یک حرف، تشکیل شده است. TEX باید بداند که یک control sequence کجا تمام می‌شود. بنابراین اگر کاراکتر بعد از یک control word یک حرف است، باید بعد از تایپ آن control word یک فاصله قرار بدهیم.

  مثال ۳.۳. اگر \inputMS را تایپ کنید، TEX بصورت طبیعی آن را به عنوان یک control word با هفت حرف، در نظر می‌گیرد.

  پرسش ۱.۳. تعریف یک «حرف» چیست؟

  پاسخ. TEX بطور عادی ۵۲ نشانه A...Z و a...z را به عنوان حرف در نظر می‌گیرد. ارقام 0...9 به عنوان حرف در نظر گرفته نمی‌شوند بنابراین در control sequenceهای نوع اول ظاهر نمی‌شوند.

- نوع دوم، مانند \'، یک control symbol خوانده می‌شود. یک control symbol (به ترتیب) از یک escape character و یک غیرـحرف تشکیل شده است. از آن‌جایی که control sequenceهای نوع دوم همیشه دقیقاً یک نشانه بعد از escape character دارند، بعد از تایپ یک control symbol، احتیاجی به یک فاصله برای جدا کردن control sequence از حرفی که بعد از آن قرار می‌گیرد، نیست.

خروجی	ورودی
mathématique	`math\'ematique`
centimètre	`centim\`etre`

جدول ۱.۳ حروف‌چینی کلمات فرانسوی mathématique و centimètre

پرسش ۲.۳. چه control sequenceهایی در متن زیر قرار دارد؟

```
\I'm \exercise3.1\\!
```

پاسخ. سه دستور وجود دارد که دو تای آن‌ها control word هستند و دیگری یک control symbol است. دستورات `\I` و `\exercise` دستورات حرفی هستند و دستور `\\` یک control symbol است.

پرسش ۳.۳. چگونه کلمات فرانسوی mathématique و centimètre را باید تایپ کنیم؟

پاسخ. پاسخ پرسش ۳.۳ در جدول ۱.۳ نشان داده شده است.

۲.۲.۳ دستور فاصله

زمانی که یک فاصله بعد از یک control word (یک control sequenceی که کاملاً از حروف تشکیل شده است) قرار می‌گیرد، TEX این فاصله را نادیده می‌گیرد (این فاصله به عنوان یک فاصله واقعی متنی که در حال حروف‌چینی است، در نظر گرفته نمی‌شود). اما هنگامی که یک فاصله بعد از یک control symbol قرار می‌گیرد، واقعاً یک فاصله است.

پرسش ۴.۳. چنانچه بخواهیم بعد از یک control word، یک فاصله وجود داشته باشد، چه کار باید کنیم؟

پاسخ. TEX دو یا تعداد بیشتری از فاصله‌های پشت سر هم را به عنوان یک فاصله در نظر می‌گیرد. بنابراین جواب سوال این نیست که «دو فاصله» تایپ کنیم. پاسخ درست سوال این است که یک «دستور فاصله» یا control space تایپ کنیم: `\␣` که از یک escape character و یک فاصله خالی تشکیل شده است. TEX با دستور فاصله به عنوان یک فاصله‌ای که نباید نادیده گرفته شود، برخورد می‌کند. توجه کنید که `\␣` یک control sequence نوع دوم (control symbol) است. به این دلیل که بعد از escape character، یک غیر_حرف (␣) وجود دارد. دو فاصله پشت سر هم معادل یک فاصله هستند، بنابراین فاصله‌های بیشتر پس از `\␣` نادیده گرفته می‌شوند. اما اگر مثلاً می‌خواهید سه فاصله پشت سر هم داخل نوشته خود وارد کنید، می‌توانید `\␣\␣\␣` را تایپ کنید.

تایپیست‌ها معمولاً باید دو فاصله در انتهای جملات قرار دهند اما بعداً خواهیم دید که TEX روش خود را برای قرار دادن فاصله بیشتر در این موارد دارد. بنابراین لازم نیست که نگران تعداد فاصله‌هایی که تایپ می‌کنید، باشید.

کاراکترهای کنترلی غیرمرئی مانند ⟨return⟩ هم ممکن است بعد از escape character قرار بگیرند که بر طبق قوانینی که در مورد انواع control sequenceها گفته شد، control sequenceهای متفاوتی را تولید خواهند کرد. TEX بطور اولیه به گونه‌ای تنظیم شده است که با ⟨return⟩\ و ⟨tab⟩\ همانند ␣\ (دستور فاصله) برخورد می‌کند. این control sequenceهای خاص نباید از نو تعریف شوند (تعریفشان تغییر کند) زیرا زمانی که به آنها در یک فایل نگاه می‌کنید، نمی‌توانید تفاوت بین آنها را ببینید.

از آن‌جایی که control sequenceها معمولاً در آخر کلمات لازم نیستند، معمولاً لازم نیست که از دستور فاصله استفاده کنید. اما یک استثنا وجود دارد: لوگوی TEX با دستور \TeX حروف‌چینی می‌شود. برای حروف‌چینی متن زیر:

TEX ignores spaces after control words.

متن زیر را تایپ می‌کنیم:

```
\TeX\ ignores spaces after control words.
```

به \ اضافه بعد از دستور \TeX توجه کنید. \ و فاصله بعد از آن در حقیقت دستور فاصله‌ای را ایجاد می‌کند که لازم است زیرا TEX فاصله‌هایی که بلافاصله بعد از یک control word قرار می‌گیرند، را نادیده می‌گیرد. بدون این \ اضافه، نتیجه بصورت زیر در می‌آمد:

TEXignores spaces after control words.

از طرفی دیگر نمی‌توانیم در تمام موارد، بعد از دستور TEX یک \ اضافه قرار دهیم.

به عنوان مثال، نمونه زیر را در نظر بگیرید:

```
The logo `\TeX'.
```

در این مورد یک backslash اضافه به هیچ وجه کار نخواهد کرد. در حقیقت اگر تایپ کنید:

```
The logo `\TeX\'.
```

یک خروجی جالب می‌گیرید.

پرسش ۵.۳. می‌توانید حدس بزنید چه اتفاقی می‌افتد؟

ورودی	خروجی
\pi	π
\Pi	Π
\aleph	$\aleph$
\infty	∞
\le	$\le$
\ge	$\ge$
\ne	$\ne$
\oplus	$\oplus$
\otimes	$\otimes$

جدول ۲.۳ تعدادی از کاراکترهای ریاضی

پاسخ. \' دستوری است که روی حرف بعد یک اکسان اگو (acute accent) قرار می‌دهد، مانند P\'olya در مثال ۲.۳. بنابراین نتیجه این است که در کد بالا، یک اعراب (accent) روی کاراکتر غیرـخالی بعدی (که یک نقطه است) قرار می‌گیرد. به عبارت دیگر، شما یک نقطه اعراب‌دار (accented period) می‌گیرید. پس خروجی بصورت زیر خواهد بود:

The logo ‘TEX.

کامپیوترها به خوبی از فرمان‌های شما پیروی می‌کنند اما متأسفانه نمی‌توانند ذهن شما را بخوانند.

۳.۲.۳ دستورات بدوی و ماکروها

TEX حدوداً ۹۰۰ دستور دارد. اما لازم نیست که همه این دستورات را یاد بگیرید چون به تعداد زیادی از این دستورات نیاز نخواهید داشت، مگر اینکه نوشته شما خیلی پیچیده باشد. گذشته از این، دستوراتی که باید یاد بگیرید به چند دسته کوچک تقسیم می‌شوند بنابراین یادگرفتن آنها زیاد سخت نخواهد بود.

مثال ۴.۳. به عنوان مثال، تعداد زیادی از دستورات، در حقیقت، نام کاراکترهای خاصی است که در فرمول‌های ریاضی به کار می‌روند. تعدادی از این دستورات در جدول ۲.۳ نشان داده شده‌اند.

هیچ ارتباطی بین حروف کوچک و بزرگ در نام دستورات وجود ندارد.

مثال ۵.۳. \pi، \Pi، \PI، و \pI چهار دستور متفاوت هستند.

۹۰۰ دستوری که ذکر شد، تمام داستان نیست زیرا می‌توان به راحتی دستورات بیشتری تعریف کرد.

مثال ۶.۳. اگر می‌خواهید دستوراتی با نام‌های دلخواه خودتان برای نشانه‌های ریاضی تعریف کنید تا به جای دستورات خود TEX از دستورات خودتان استفاده کنید تا آنها را بهتر به خاطر بسپارید، آزادانه می‌توانید این کار را انجام دهید.

حدود ۳۰۰ دستور از این ۹۰۰ دستور، دستورات بدوی (primitives) خوانده می‌شوند. دستورات بدوی، دستورات سطح پایینی هستند که قابل تجزیه شدن به دستورات ساده‌تری نیستند. تمام دستورات دیگر نهایتاً با استفاده از دستورات بدوی تعریف می‌شوند. به اینگونه دستورات به اصطلاح «ماکرو» گفته می‌شود.

مثال ۷.۳. دستور \input یک دستور بدوی است در حالی که دستورات \' و \" ماکرو هستند. دستورات \' و \" با استفاده از دستور بدوی \accent تعریف شده‌اند.

ما به ندرت از دستورات بدوی در نوشته‌هایمان استفاده می‌کنیم به این خاطر که دستورات بدوی خیلی ابتدایی هستند. اگر قرار باشد در نوشته‌هایمان از دستورات بدوی استفاده کنیم، باید از تعداد زیادی از دستورات استفاده کنیم که وقت زیادی خواهد گرفت و اشتباهات زیادی به وجود خواهد آورد. کلاً بهتر است که به جای دستورات سطح پایین (که هر بار مجبور باشیم بگوییم چه کاری انجام بده)، از دستورات سطح بالا (که تعیین می‌کنند چه کاری مدنظر ماست) استفاده کنیم. دستورات سطح بالا تنها لازم است یک بار با استفاده از دستورات بدوی تعریف شوند.

مثال ۸.۳. دستور \TeX، لوگوی TEX را حروف‌چینی می‌کند و دستور \' یک اکسان اگو (acute accent) روی کاراکتر بعدی قرار می‌دهد. چنانچه سبک حروف چاپی تغییر کند، هر دو دستور \TeX و \' به ترکیب متفاوتی از دستورات بدوی احتیاج خواهند داشت. اگر بخواهیم لوگوی TEX را تغییر بدهیم، تنها باید یک تعریف را تغییر بدهیم و این تغییرات بصورت خودکار در هر جایی که نیاز باشند، ظاهر خواهند شد. در صورتیکه اگر لوگوی TEX هر بار با استفاده از دستورات بدوی حروف‌چینی شده بود و ما می‌خواستیم لوگوی TEX را تغییر بدهیم، باید تغییرات بسیار عظیمی در نوشته‌مان می‌دادیم (در هر جایی از نوشته‌مان که لوگوی TEX وجود داشت باید دستورات بدوی که لوگوی TEX را حروف‌چینی می‌کنند، تغییر می‌دادیم).

دستوراتی در یک سطح بالاتری نیز وجود دارند که قالب کلی یک نوشتار را کنترل می‌کنند.

مثال ۹.۳. اگر در حال حروف‌چینی یک کتاب هستیم، می‌توانیم دستوری مانند \exercise برای حروف‌چینی تمرین‌های کتاب‌مان تعریف کنیم و قبل از شروع هر تمرین این دستور \exercise را در نوشته‌مان قرار دهیم. دستور \exercise را می‌توانیم به گونه‌ای برنامه‌نویسی کنیم تا TEX بصورت خودکار تمام کارهای زیر را خودش انجام دهد:

۱. شماره تمرین را محاسبه کند (مثلاً شماره تمرین دوم در فصل سوم 3.2 باشد).

۲. عبارت **Exercise 3.2** ◂ را با قلم‌های مناسب در یک سطر جداگانه بطوری که مثلث سیاه رنگ در حاشیه چپ قرار بگیرد، حروف‌چینی کند.

۳. قبل از این سطر، کمی فاصله اضافه قرار بدهد یا اینکه این سطر را اگر مناسب بود در یک صفحه جدید قرار بدهد.

۴. بعد از این سطر صفحه جدیدی را شروع نکند.

۵. سطر بعدی تورفتگی نداشته باشد.

قطعاً به نفع ما خواهد بود که هر باری که می‌خواهیم یک تمرین را حروف‌چینی کنیم، تک تک این دستورالعمل‌ها را تایپ نکنیم. از آنجایی که تمرین‌ها با استفاده از دستور سطح بالای `\exercise` حروف‌چینی می‌شوند، اگر بخواهیم شکل و شمایل تمرین‌ها را از اساس تغییر بدهیم، تنها باید تعداد کمی از تعریف‌ها در دستور `\exercise` را تغییر بدهیم.

پرسش ۶.۳. چگونه می‌توانیم بفهمیم که یک دستور، یک دستور بدوی است یا یک ماکرو؟

پاسخ. دو روش برای این کار وجود دارد:

۱. تمام دستوراتی که در این نوشتار بیان می‌شوند، در نمایه هم ظاهر می‌شوند. در صورتی که دستوری بدوی باشد، در نمایه، جلوی دستور کلمه primitive داخل یک جفت پرانتز نوشته می‌شود. بنابراین با مراجعه به نمایه نوشتار، می‌توانیم بفهمیم که یک دستور خاص بدوی است یا ماکرو.

۲. می‌توانیم معنای هر دستور را هنگام اجرای TEX به نمایش در بیاوریم. اگر `\cs` یک دستور باشد (cs مخفف control sequence است) و تایپ کنیم `\show\cs`، TEX معنای فعلی دستور را به ما نشان می‌دهد.

مثال ۱۰.۳. `\show\input` عبارت `. \input=\input >` را نمایش می‌دهد که به این معنی است که دستور `\input`، یک دستور بدوی است.

مثال ۱۱.۳. `\show\thinspace` نتیجه زیر را می‌دهد:

```
> \thinspace=macro:
->\kern .16667em .
```

این یعنی دستور \thinspace یک ماکرو و معادل \kern .16667em می‌باشد. اگر \show\kern را تایپ کنید، متوجه خواهید شد که دستور \kern، یک دستور بدوی است.

خروجی دستور \show در ترمینال و فایل log‌ای که بعد از اجرای TEX می‌گیرید، ظاهر می‌شود.

پرسش ۷.۳. کدام یک از دستورات \␣ و \⟨return⟩، بدوی هستند؟

پاسخ. اگر به نمایه نگاه کنیم، متوجه می‌شویم که \␣ یک دستور بدوی است. \show\␣ نتیجه زیر را می‌دهد:

```
>␣\^^M=macro:
->\␣.
```

بنابراین دستور \␣ بدوی است. \show\⟨return⟩ خروجی زیر را می‌دهد:

```
>␣\^^M=macro:
->\␣.
```

بنابراین دستور \⟨return⟩ یک ماکرو است که معادل دستور بدوی \␣ می‌باشد. از آنجایی که یک return بصورت ^^M نمایش داده می‌شود، دستور \⟨return⟩ بصورت زیر تعریف شده است:

```
\def\^^M{\␣}
```

بنابراین، خروجی \show\^^M بصورت زیر است:

```
>␣\^^M=macro:
->\␣.
```

۳.۳ قالب‌های TEX

موتور TEX تنها دارای حدوداً ۳۰۰ دستور بدوی است (ماکروها یا دستورات سطح بالا در موتور TEX وجود ندارند). قالب‌های TEX (TEX formats) زیادی مانند Plain، Lollipop، LATEX، و غیره وجود دارند که روی موتور TEX می‌نشینند. این قالب‌ها دستورات سطح بالایی با استفاده از دستورات بدوی که موتور TEX در اختیار قرار می‌دهد، تعریف می‌کنند. در این نوشتار ما روی قالب plain تمرکز می‌کنیم. قالب plain، از ۶۰۰ دستور سطح بالا تشکیل شده است. بنابراین، زمانی که TEX شروع به پردازش یک نوشته می‌کند، این ۶۰۰ دستور سطح بالا

به اضافه ۳۰۰ دستور بدوی موتور TEX در دسترس هستند. به همین خاطر است که TEX ادعا می‌کند زمانی که شروع به پردازش یک نوشته می‌کند، ۹۰۰ دستور را می‌داند. در فصل‌های بعدی این نوشتار خواهیم دید که چگونه می‌توانیم از قالب plain و قلم‌هایی که همراه سیستم TEX (قلم‌های Computer Modern) وجود دارند، برای تهیه نوشتارها در یک قالب قابل تغییر و بسیار انعطاف‌پذیر که نیاز بسیاری از افراد را برآورده می‌سازند، استفاده کنیم. به خاطر بسپارید که قالب plain تنها یکی از قالب‌های بی‌شماری است که می‌تواند با استفاده از دستورات بدوی موتور TEX طراحی شود. اگر قالب دیگری می‌خواهید، می‌توانید TEX را به گونه‌ای تنظیم کنید تا هر آنچه را که مدنظر شما است، انجام دهد. بهترین راه برای طراحی یک قالب جدید احتمالاً این است که از قالب plain شروع کنیم و تعریف‌هایش را کم کم تغییر دهیم تا تجربه بیشتری بدست بیاوریم.

اگر دستوری جزء قالب plain باشد و رفتار خاصی داشته باشد، به این معنا نیست که TEX اصرار به این رفتار خاص دارد. بلکه با تغییر تعریف این دستور، رفتار TEX نیز تغییر خواهد کرد.

پرسش ۸.۳. چند دستور دو کاراکتری (شامل escape character) متفاوت می‌توانیم داشته باشیم؟

پاسخ. در TEX، ۲۵۶ کاراکتر وجود دارد. دستورات دو کاراکتری (شامل escape character) متفاوت، هم می‌توانند control word باشند و هم control symbol. TEX اجازه می‌دهد تا هر کدام از این ۲۵۶ کاراکتر، escape character باشند. بنابراین دستورات می‌توانند با escape characterهای متفاوت شروع شوند اما TEX تفاوتی بین دستورات با escape characterهای متفاوت قایل نمی‌شود. بنابراین $1 \times 256 = 256$ دستور دو کاراکتری (شامل escape character) متفاوت وجود دارد. هنگامی که TEX شروع به کار می‌کند، بیشتر این دستورات تعریف نشده هستند.

پرسش ۹.۳. چند دستور سه کاراکتری (شامل escape character) متفاوت می‌توانیم داشته باشیم؟

پاسخ. دستورات سه کاراکتری (شامل escape character) متفاوت، فقط می‌توانند control word باشند. اگر فرض کنیم که ۵۲ حرف وجود دارد (`A...Z` و `a...z`)، آنگاه

$$1 \times 52 \times 52 = 52^2 = 2,704$$

دستور سه کاراکتری (شامل escape character) متفاوت می‌توانیم داشته باشیم.

در فصل‌های بعدی خواهیم دید که با استفاده از دستور بدوی `\catcode` می‌توانیم هر کدام از این ۲۵۶ کاراکتر را به یک حرف تبدیل کنیم. بنابراین، می‌توانیم

$$1 \times 256 \times 256 = 256^2 = 65,536$$

دستور سه کاراکتری (شامل escape character) متفاوت داشته باشیم.

فصل ۴

قلم‌ها

۱.۴ انواع قلم‌ها

گاهی اوقات در نوشته‌مان می‌خواهیم قلم را تغییر دهیم مانند نمونه زیر:

to be **bold** or to *emphasize* something.

TEX با ۲۵۶ کاراکتر سر و کار دارد که به آنها «قلم‌های» حروف چاپی گفته می‌شود. از دستورات برای انتخاب یک قلم خاص استفاده می‌کنیم.

مثال ۱.۴. نمونه بالا را در قالب plain می‌توانیم به شکل زیر تایپ کنیم:

```
to be \bf bold \rm or to \sl emphasize \rm something.
```

۲.۴ دستورات قلم‌ها

دستوراتی که قالب plain برای تغییر قلم‌ها در اختیار قرار می‌دهد، در جدول ۱.۴ نشان داده شده‌اند.

هنگامی که TEX را روی فایل‌مان اجرا می‌کنیم، در ابتدای نوشته‌مان قلم عادی roman (\rm) می‌گیریم مگر اینکه از دستور قلم دیگری استفاده کرده باشیم. توجه کنید که دو قلم از این قلم‌ها، شیب مایل برای تأکید کردن دارند:

- قلم خوابیده (\sl) در اصل همان قلم roman است اما حروف کمی کج شده‌اند.
- در حالی که حروف در قلم ایتالیک (\it) با سبک متفاوتی حروف‌چینی می‌شوند.

دستور	توضیح	نمونه
\rm	قلم را به قلم عادی roman تغییر می‌دهد	Roman
\sl	قلم را به قلم خوابیده roman تغییر می‌دهد	*Slanted*
\it	قلم را به سبک ایتالیک تغییر می‌دهد	*Italic*
\tt	قلم را به قلم تایپ تغییر می‌دهد	Typewriter
\bf	قلم را به قلم سیاه تغییر می‌دهد	**Bold**

جدول ۱.۴ دستورات قالب plain برای تغییر قلم‌ها

برای بهتر درک کردن تفاوت بین سبک‌های roman و خوابیده/ایتالیک، می‌توانید به شکل حروف در یک قلم غیر خوابیده/ایتالیک نگاه کنید و آن را با شکل حروف در یک قلم خوابیده/ایتالیک مقایسه کنید. حروف چاپی خوابیده roman در دهه ۱۹۳۰ معرفی شد اما برای اولین بار بصورت گسترده در اواخر دهه ۱۹۷۰ به عنوان یک جایگزین برای حروف چاپی معمولی ایتالیک، مورد استفاده قرار گرفت. حروف خوابیده در متن‌های ریاضی سودمند خواهند بود زیرا حروف خوابیده از حروف ایتالیک در فرمول‌های ریاضی به راحتی قابل تشخیص هستند. دو کاربرد حروف چاپی ایتالیک برای دو منظور متفاوت—مثلاً هنگامی که قضیه‌های ریاضی با سبک ایتالیک بیان می‌شوند و همچنین نام متغیرها در این قضیه‌ها با سبک ایتالیک ظاهر می‌شوند—باعث سردرگمی شده است که اکنون با استفاده از حروف چاپی خوابیده پیش نمی‌آید. ما معمولاً درباره برتری نسبی حروف خوابیده نسبت به حروف ایتالیک اتفاق نظر نداریم، اما حروف چاپی خوابیده به سرعت در حال محبوب شدن برای عنوان کتاب‌ها و مجلات در کتاب‌نامه‌ها، هستند.

قلم‌های خاص برای تأکید کردن موثر هستند اما نه برای خواندن طولانی مدت. اگر قسمت بزرگی از یک کتاب با قلم سیاه، یا قلم خوابیده، یا قلم ایتالیک، حروف‌چینی شده بود، چشمان شما هنگام خواندن این کتاب خیلی زود خسته می‌شدند. بنابراین اکثر مطالب حروف‌چینی شده با قلم roman ظاهر می‌شوند.

۳.۴ چگونگی تغییر قلم

اگر هر باری که در نوشته‌مان می‌خواهیم به سبک roman برگردیم، مجبور باشیم از دستور \rm استفاده کنیم، حروف‌چینی بسیار رنج‌آور خواهد بود. TEX راه ساده‌تری برای انجام این کار با استفاده از آکولادها (curly braces) در اختیار ما قرار می‌دهد. ما می‌توانیم قلم را داخل نشانه‌های خاص } و { تغییر دهیم بدون اینکه قلم خارج این نشانه‌ها را تحت تأثیر قرار دهیم.

مثال ۲.۴. برای حروف‌چینی متن زیر:

to be **bold** or to *emphasize* something.

می‌توانیم آن را به شکل زیر تایپ کنیم:

```
to be {\bf bold} or to {\sl emphasize} something.
```

این یک مورد خاص از کاربرد ایده عمومی «گروه‌بندی» است که در مورد آن در فصل ۵ صحبت خواهیم کرد. بنابراین بهتر است که راه اول تغییر قلم را فراموش کنیم و در عوض از گروه‌بندی استفاده کنیم. در این صورت فایل TEXمان ظاهر طبیعی‌تری خواهد داشت و هرگز (یا به ندرت) مجبور خواهیم بود که از دستور \rm استفاده کنیم.

پرسش ۱.۴. چگونه می‌توانیم متن زیر را تایپ کنیم؟ (از گروه‌بندی استفاده کنید)

Ulrich Dieter, *Journal für die reine und angewandte Mathematik* **201** (1959), 37–70.

پاسخ. راحت‌تر است که از یک گروه برای دو دستور \sl و \bf استفاده کنیم.

```
Ulrich Dieter, {\sl Journal f\"ur die reine und angewandte
Mathematik\/ \bf201} (1959), 37--70.
```

\/ اصلاح/ بهبودی است که در قسمت ۴.۴ توضیح داده می‌شود.

۴.۴ اصلاح ایتالیک

به دقت به متن زیر نگاه کنید:

italicized and *slanted* words

از آنجایی که سبک‌های ایتالیک و خوابیده به سمت راست شیب دارند، حروف d در فاصله‌هایی، که این کلمات رو از کلمات بعدی که با قلم roman حروف‌چینی شده‌اند جدا می‌کنند، قرار گرفته‌اند. در نتیجه، این فاصله‌ها خیلی کم هستند هر چند در پای حروف درست هستند. برای اینکه فاصله سپید بین اینگونه کلمات رو برابر کنیم، TEX به ما اجازه می‌دهد تا دستور \/ را قبل از برگشتن به حروف غیرـخوابیده قرار دهیم. زمانی که تایپ می‌کنیم:

```
{\it italicized\/} and {\sl slanted\/} words
```

خروجی زیر را می‌گیریم:

italicized and *slanted* words

که ظاهر بهتری دارد. دستور \/ به TEX می‌گوید تا یک «اصلاح ایتالیک» (italic correction)، بسته به آن حرف، را به حرف قبلی بیفزاید. این اصلاح در یک قلم ایتالیک برای حرف f در حدود چهار برابر اصلاح برای حرف c است.

در همه موارد، استفاده از اصلاح ایتالیک، نتیجه مطلوب را نمی‌دهد چون فاصله ظاهری به عوامل دیگری بستگی دارد. قانون کلی استفاده از اصلاح ایتالیک این است که قبل از اینکه قلم را از خوابیده یا ایتالیک به roman یا سیاه تغییر دهیم، از \/ استفاده کنیم مگر اینکه کاراکتر بعدی یک نقطه یا ویرگول (comma) باشد.

مثال ۳.۴.

```
{\it italics\/} for {\it emphasis}.
```

براساس شیوه‌نامه‌های قدیمی، نقطه‌گذاری‌های بعد از یک کلمه باید دقیقاً با همان قلمی نوشته شوند که آن کلمه نوشته شده است. اما یک نقطه ویرگول (semicolon) ایتالیک شکل ظاهری نامناسبی دارد بنابراین این قانون در حال تغییر است. اگر یک نقطه ویرگول بعد از یک کلمه ایتالیک وجود داشته باشد، بهتر است آن را بصورت {\it word\/}; تایپ کنیم.

پرسش ۲.۴. در متن زیر، چگونه می‌توانیم کلمه roman را وسط یک جمله ایتالیک حروف‌چینی کنیم؟

Explain how to typeset a roman *word in the midst of an italicized sentence.*

پاسخ.

```
{\it Explain how to typeset a\/ {\rm roman} word in the midst
of an italicized sentence.}
```

تک تک حروف در هر قلمی یک اصلاح ایتالیک دارند که می‌توانیم با استفاده از \/ به آن دست پیدا کنیم. مقدار این اصلاح در قلم‌های غیر–خوابیده صفر است هر چند استثناهایی وجود دارد. برای اینکه یک 'f' سیاه داخل علامت نقل‌قول تکی بنویسم، باید آن را بصورت `{\bf f\/}' تایپ کنیم چون در غیر اینصورت 'f' می‌گیریم.

پرسش ۳.۴. چگونه می‌توانیم دستوری به نام \ic تعریف کنیم بطوری که وقتی تایپ می‌کنیم \ic c، درجینه (register) \dimen0 ی TEX، مقدار اصلاح ایتالیک حرف *c* را به ما بدهد؟

پاسخ. برای محاسبه مقدار اصلاح ایتالیک یک حرف، باید خود حرف و اصلاح ایتالیک آن را داخل یک کادر قرار دهیم و عرض کادر (a) را محاسبه کنیم. سپس تنها آن حرف (بدون اصلاح ایتالیک) را داخل کادر قرار دهیم و عرض کادر (b) را محاسبه کنیم. بنابراین مقدار اصلاح ایتالیک حرف برابر خواهد بود با $a - b$. بنابراین، دستور \ic را باید به شکل زیر تعریف کنیم:

```
\def\ic#1{\setbox0=\hbox{#1\/}\dimen0=\wd0
          \setbox0=\hbox{#1}\advance\dimen0 by -\wd0}
```

۵.۴ دستور بدوی \nullfont

دستور بدوی به نام \nullfont وجود دارد که نمایانگر قلمی است که هیچ کاراکتری ندارد. در مواردی که شما قلم دیگری را مشخص نکرده‌اید، این قلم همیشه حاضر است.

۶.۴ اندازه قلم‌ها

قلم‌ها می‌توانند اندازه و شکل متفاوتی داشته باشند.

مثال ۴.۴. قلم متن زیر

This is a test (experiment)—as you can see.

یک قلم «۱۰ پونت» خوانده می‌شود زیرا ویژگی‌های خاصی از طراحی آن با اندازه ۱۰ پونت (هنگامی که در واحد پرینترها اندازه گرفته می‌شود) مشخص می‌شوند. در فصل‌های بعدی این نوشتار در مورد واحدهای مختلف (از جمله پونت) صحبت خواهیم کرد اما در حال حاضر کافی است همینقدر بدانید که در این مثال پرانتزها دقیقاً ۱۰ پونت ارتفاع دارند و عرض کاراکتر em-dash، تنها ۱۰ پونت است.

بخشی از نوشتار ما ممکن است با قلم ۱۰ پونت، بخشی با قلم ۹ پونت، بخشی با قلم ۸ پونت، بخشی با قلم ۷ پونت، بخشی با قلم ۶ پونت، و بخشی با قلم ۵ پونت حروف‌چینی شوند. هر قلمی که در نوشته TEX ما استفاده می‌شود، با یک دستور در ارتباط است.

مثال ۵.۴. قلم roman ۱۰ پونت با دستور \tenrm و قلم roman ۹ پونت با دستور \ninerm مشخص می‌شوند. قلم‌های خوابیده‌ای که با دستورات \tenrm و \ninerm مطابقت دارند، با دستورات (به ترتیب) \tensl و \ninesl مشخص می‌شوند.

این دستورات نه داخل موتور TEX وجود دارند و نه نام قلم‌ها می‌باشند. کاربران TEX زمانی که می‌خواهند از قلم‌های جدیدی در نوشته‌شان استفاده کنند، باید نام‌های مناسبی را برای دستوراتی که این قلم‌های نو را مشخص می‌کنند، انتخاب کنند. از این دستورات برای عوض کردن قلم‌ها استفاده می‌کنیم. زمانی که از قلم‌هایی با اندازه‌های متفاوت به طور همزمان استفاده می‌کنیم، TEX حروف را براساس «خط کرسی» (baseline)شان می‌چیند.

مثال ۶.۴. اگر تایپ کنیم:

```
\tenrm smaller \ninerm and smaller
\eightrm and smaller \sevenrm and smaller
\sixrm and smaller \fiverm and smaller \tenrm
```

TEX آن را به شکل زیر حروف‌چینی می‌کند:

smaller and smaller and smaller and smaller and smaller and smaller

در ابتدای قسمت ۲.۴ گفتیم که با استفاده از دستور \rm می‌توانیم متن را با قلم roman حروف‌چینی کنیم اما بعداً گفتیم که از دستور \tenrm برای این کار استفاده می‌کنیم.

پرسش ۴.۴. استفاده از کدام یک از این دو دستور درست است؟

پاسخ. هر دو دستور به درستی کار می‌کنند.

پرسش ۵.۴. فرق بین این دو دستور چیست؟

پاسخ. زمانی که از دستور \rm استفاده می‌کنیم، در حقیقت قلم را به قلم roman در اندازه فعلی تغییر می‌دهیم. به عنوان مثال اگر اندازه فعلی قلم ۸ پونت باشد و ما از دستور \rm استفاده کنیم، قلم متن به قلم roman با اندازه ۸ پونت تغییر خواهد کرد. اما دستور \tenrm همیشه قلم را به قلم roman با اندازه ۱۰ پونت تغییر می‌دهد.

در قالب plain، قلم‌ها تنها با اندازه ۱۰ پونت وجود دارند بنابراین \rm همیشه معادل \tenrm خواهد بود اما در قالب‌های پیچیده‌تر معنای \rm در قسمت‌های مختلف نوشته، تغییر خواهد کرد.

مثال ۷.۴. در یک قالب ممکن است دستوری به نام \tenpoint وجود داشته باشد که وقتی از آن استفاده می‌کنیم، دستور \rm را معادل دستور \tenrm، دستور \sl را معادل دستور \tensl، و ...، می‌کند. در حالی که ممکن است دستوری به نام \ninepoint وجود داشته باشد که تعریف‌ها را به گونه‌ای تغییر دهد تا دستور \rm معادل دستور \ninerm و ...باشد. به این ترتیب، تعریف دستورات \rm و \sl بصورت مرتب در پشت

صحنه تغییر می‌کند و یک تایپیست نیازی به اینکه به خاطر داشته باشد در حال حاضر از چه اندازه یا چه سبک قلمی استفاده می‌کند، نخواهد داشت.

پرسش ۶.۴. چرا به جای دستورات \10point و \10rm، نام‌های \tenpoint و \tenrm را انتخاب کردیم؟

پاسخ. به این دلیل که نام یک control word از حروف تشکیل شده است نه از ارقام.

پرسش ۷.۴. فرض کنید که یک کتاب بسیار عظیم نوشته‌اید و در آن از قلم خوابیده برای تأکید متن استفاده کرده‌اید. حال ناشر به شما می‌گوید که باید تمام کلماتی که با قلم خوابیده حروف‌چینی شده‌اند، با قلم ایتالیک حروف‌چینی شوند. یک راه ساده برای انجام این کار چیست؟

پاسخ. می‌توانیم دستور زیر:

```
\def\sl{\it}
```

را در ابتدای فایل TEX خود قرار دهیم و سایر تعریف‌های دستور \sl را که ممکن است داخل فایل قالب‌مان باشد حذف کنیم (مثلاً ممکن است تعریفی از دستور \sl داخل ماکروی \tenpoint باشد).

۷.۴ چگونگی فراخوانی قلم‌ها

هر قلم یک نام خارجی دارد که آن را از قلم‌های دیگر مشخص می‌کند.

مثال ۸.۴. به عنوان مثال، قلم استفاده شده در متن زیر:

This is a test.

cmr9 نام دارد که مخفف Computer Modern Roman در اندازه ۹ پونت، است. برای اینکه TEX بتواند از این قلم استفاده کند، باید دستور \ninerm را بصورت زیر تعریف کنیم:

```
\font\ninerm=cmr9
```

بصورت کلی برای استفاده از یک قلم خاص از دستور زیر برای فراخوانی آن قلم در حافظه TEX استفاده می‌کنیم:

\font\cs=⟨external font name⟩

سپس دستور \cs آن قلم را برای حروف‌چینی انتخاب می‌کند. قالب plain بصورت اولیه تنها ۱۶ قلم را در دسترس قرار می‌دهد اما می‌توانیم با استفاده از \font به هر قلم دیگری که در کتاب‌خانه قلم سیستم ما وجود دارد، دسترسی پیدا کنیم.

معمولاً می‌توانیم با بزرگ کردن یا کوچک کردن تصویر کاراکترهای یک قلم، از آن قلم در چندین اندازه متفاوت استفاده کنیم. هر قلمی برای یک اندازه خاصی طراحی شده است که نشانگر اندازه پیش‌فرض آن قلم بصورت عادی است.

مثال ۹.۴. قلم cmr9 برای اندازه ۹ پونت طراحی شده است.

در بسیاری از سیستم‌ها می‌توان یک قلم خاص را با بزرگ کردن یا کوچک کردن مقیاس ابعاد آن قلم، در یک محدوده‌ای از اندازه‌ها، استفاده کرد. برای فراخوانی یک قلم در حافظه TEX، که می‌خواهیم مقیاس ابعاد آن را بزرگ و یا کوچک کنیم، می‌توانیم بصورت کلی از دستور زیر استفاده کنیم:

\font\cs=⟨external font name⟩ at ⟨desired size⟩

مثال ۱۰.۴. دستور زیر:

```
\font\magnifiedfiverm=cmr5 at 10pt
```

باعث می‌شود تا قلم Computer Modern Roman با اندازه ۵ پونت، در دو برابر اندازی عادی‌اش فراخوانی شود.

شما قبل از اینکه از این ویژگی استفاده کنید، باید اطمینان حاصل کنید که حروف‌چین شما از قلم در اندازه‌ای که شما می‌خواهید استفاده کنید، پشتیبانی می‌کند. TEX هر مقدار ⟨desired size⟩ی را که مثبت و کمتر از ۲۰۴۸ پونت باشد قبول می‌کند اما خروجی نهایی درست نخواهد بود مگر اینکه قلمی که ابعاد آن کوچک و یا بزرگ شده است، روی پرینتر شما موجود باشد.

پرسش ۸.۴. تفاوت cmr5 at 10pt با قلمی که اندازه عادی‌اش ۱۰ پونت است (cmr10) چیست؟

پاسخ. تفاوت‌های زیادی وجود دارد. قلمی که به خوبی طراحی شده باشد، در اندازه‌های پونت متفاوت شکل متفاوتی خواهد داشت و حروف اغلب ارتفاع و عرض نسبی متفاوتی خواهند داشت تا خوانایی افزایش پیدا کند.

Ten-point type is different from magnified five-point type.

معمولاً بهتر است تا قلم‌ها را کمی بیشتر یا کمتر از اندازه‌ای که طراحی شده‌اند، بزرگ یا کوچک کنیم مگر اینکه خروجی نهایی بعد از پردازش TEX، قرار است که بصورت تصویری کوچک شود یا مگر اینکه ما می‌خواهیم خروجی غیر عادی داشته باشیم. یک راه دیگر برای برگ‌نمایی یک قلم این است که یک فاکتور مقیاس که وابسته به اندازه‌ای که آن قلم برای آن طراحی شده است، را مشخص کنیم.

مثال ۱۱.۴. دستور

```
\font\magnifiedfiverm=cmr5 scaled 2000
```

راه دیگری برای بزرگ کردن اندازه قلم cmr5 در دو برابر اندازه عادی‌اش است.

فاکتور مقیاس به عنوان یک عدد صحیحی که نمایانگر یک نسبت بزرگ‌نمایی ضرب در ۱۰۰۰ است، مشخص می‌شود. بنابراین فاکتور مقیاس ۱۲۰۰ بزرگ‌نمایی به اندازه 1.2 را مشخص می‌کند.

پرسش ۹.۴. دو راه برای فراخوانی قلم cmr10 در نصف اندازه عادی‌اش در حافظه TEX را بیان کنید؟

پاسخ.

```
\font\squinttenrm=cmr10 at 5pt
\font\squinttenrm=cmr10 scaled 500
```

یک راه دیگر برای بزرگ‌نمایی قلم این است که بزرگ‌نمایی‌ها می‌توانند در نسبت‌های هندسی ظاهر شوند. با استفاده از این روش، قلم‌ها به اضافه اینکه در اندازه واقعی خودشان در دسترس هستند، در بزرگ‌نمایی‌هایی به اندازه‌های 1.2، $1.44 = 1.2^2$، $1.728 = 1.2^3$، و حتی بالاتر نیز در دسترس خواهند بود. بنابراین با استفاده از این روش می‌توانید هم تمام نوشتار را به اندازه 1.2 یا 1.44 بزرگ کنید و هم از مجموعه قلم‌های موجود استفاده کنید. قالب plain دستورات زیر را در اختیار شما قرار می‌دهد:

\magstep0: فاکتور مقیاس ۱۰۰۰

\magstep1: فاکتور مقیاس ۱۲۰۰

\magstep2: فاکتور مقیاس ۱۴۴۰

\magstep3: فاکتور مقیاس ۱۷۲۸

\magstep4: فاکتور مقیاس ۲۰۷۴

\magstep5: فاکتور مقیاس ۲۴۸۸

مثال ۱۲.۴. وقتی می‌گوییم:

```
\font\bigtenrm=cmr10 scaled\magstep2
```

قلم cmr10 را در 1.2×1.2 برابر اندازه عادی‌اش فراخوانی می‌کنیم.

مثال ۱۳.۴. به نمونه زیر توجه کنید:

"This is cmr10 at normal size (\magstep0)."

"This is cmr10 scaled once by 1.2 (\magstep1)."

"This is cmr10 scaled twice by 1.2 (\magstep2)."

همانطور که در نمونه بالا مشاهده می‌کنید، مقدار کمی بزرگ‌نمایی، قلم‌ها را بسیار بزرگ می‌کند.

در قالب plain هنچنین دستوری به نام \magstephalf وجود دارد که یک قلم را به اندازه $\sqrt{1.2}$ (فاکتور مقیاس ۱۰۹۵) برابر بزرگ می‌کند. همانطور که از نام این دستور پیداست، مقدار بزرگ‌نمایی آن بین (وسط) مقدار بزگ‌نمایی دستورات \magstep0 و \magstep1 می‌باشد.

در فصل‌های بعدی این نوشتار خواهیم دید که چگونه می‌توانیم کل نوشتاری را که در آن قلم‌هایی را که با هر بزرگ‌نمایی فراخوانی شده‌اند، بزرگ کنیم.

مثال ۱۴.۴. اگر قلمی را با مقیاس \magstep1 فراخوانی کرده باشیم، با قرار دادن دستور

```
\magnification=\magstep2
```

قلمی که برای حروف‌چینی به کار می‌رود مقیاسی برابر با \magstep3 خواهد داشت. بصورت کاملاً مشابه اگر قلمی را با مقیاس \magstephalf فراخوانی کرده باشیم، با قرار دادن دستور

```
\magnification=\magstephalf
```

قلمی که برای حروف‌چینی به کار می‌رود مقیاسی برابر با \magstep1 خواهد داشت.

فصل ۵

گروه‌بندی

هر از گاهی لازم است تا بخشی از نوشته‌مان را به عنوان یک واحد در نظر بگیریم بنابراین باید به نحوی مشخص کنیم که این بخش از نوشته‌مان کجا شروع و کجا تمام می‌شود. برای این منظور دو «کاراکتر گروه‌بندی» معنای خاصی برای TEX دارند که با آن‌ها (همانند escape character) بصورت کاملاً متفاوتی، از نشانه‌های عادی که تایپ می‌کنیم، برخورد می‌شود. در قالب plain } و { کاراکترهای گروه‌بندی هستند. در قسمت ۳.۴، نمونه‌هایی از کاربرد گروه‌بندی را مشاهده کردیم که تغییر قلم داخل یک گروه تأثیری در قلم‌های خارج آن گروه ندارند. همانطور که بعداً خواهیم دید، دقیقاً همین قاعده برای تقریباً هر چیز دیگری که داخل یک گروه تعریف شده باشد، صدق می‌کند.

مثال ۱.۵. اگر دستوری را داخل یک گروه تعریف کرده باشید، وقتی که گروه تمام می‌شود، تعریف آن دستور هم ناپدید می‌گردد. به این طریق شما با تغییر قراردادهای عادی TEX بصورت موقت داخل یک گروه، به راحتی می‌توانید به TEX یاد بدهید تا کاری کاملاً غیر معمول را انجام دهد. از آنجایی که تغییرات داخل یک گروه، در خارج آن گروه ناپدید می‌شوند، زمانی که ساخت و ساز غیر معمول تمام شد، اگر فراموش کردید که به قراردادهای عادی برگردید، دیگر نیازی به نگرانی درباره خراب شدن یا بر هم زدن مابقی یک نوشتار نیست.

در علوم کامپیوتر این جنبه گروه‌بندی نامی برای خودش دارد زیرا بصورت کلی یک جنبه مهم زبان‌های برنامه‌نویسی به شمار می‌رود. در علوم کامپیوتر این جنبه گروه‌بندی «ساختار کنده‌ای» (block structure) گفته می‌شود و تعریف‌هایی که تنها داخل یک گروه تأثیر دارند، تعریف‌های محلی (local) آن گروه خوانده می‌شوند.

شما ممکن است حتی زمانی که ساختار کنده‌ای برایتان مهم نیست و فقط می‌خواهید کنترل بهتری روی فاصله‌ها داشته باشید، بخواهید از گروه‌بندی استفاده کنید.

مثال ۲.۵. دستور \TeX که لوگوی TEX را حروف‌چینی می‌کند یک بار دیگر در نظر بگیریم. در زیرقسمت ۲.۲.۳

دیدیم که یک فاصله خالی بعد از این دستور قورت داده می‌شود مگر اینکه ␣\TeX\␣ را تایپ کنیم. در عین حال، زمانی که کاراکتر بعد از لوگوی TEX یک فاصله خالی نیست، تایپ کردن ␣\TeX\␣ اشتباه است. در همه موارد درست خواهد بود که از یک گروه ساده استفاده کنیم:

```
{\TeX}
```

چه کاراکتر بعد از لوگوی TEX یک فاصله باشد چه نباشد. دلیل این امر این است که {، جلوی TEX را از فرو بردن یک فاصله اختیاری در دستور \TeX، می‌گیرد. این موضوع زمانی که از یک ویرایشگر متن استفاده می‌کنیم مفید خواهد بود (زمانی که کلمه‌ای را که چندین بار در نوشته‌مان وجود دارد، با یک دستور جایگزین می‌کنیم).

یک راه دیگر این است که از یک گروه خالی برای همین منظور استفاده کنیم:

```
\TeX{}
```

{} در اینجا یک گروه بدون کاراکتر است بنابراین باعث ظاهر شدن چیزی در خروجی نمی‌شود اما باعث می‌شود تا TEX فاصله‌ها را از قلم نیندازد.

پرسش ۱.۵. بعضی اوقات لازم است که یک کلمه نادر مانند shelfful را حروف‌چینی کنیم. این کلمه بصورت shelfful (بدون لیگاتور ff) ظاهر بهتری دارد. چگونه می‌توانیم TEX را فریب بدهیم تا تصور کند دو f پشت سر هم در چنین کلمه‌ای وجود ندارد؟

پاسخ. می‌توانیم از `shelf{}ful` یا `shelf{}ful` و یا دیگر موارد مشابه استفاده کنیم یا حتی می‌توانیم از `shelf\/ful` استفاده کنیم که به جای shelffful به ما shelfful می‌دهد. در حقیقت، این ایده (وارد کردن اصلاح ایتالیک) بهتر است زیرا TEX بعد از تیره‌بندی (hyphenating) `shelf{}ful` ، لیگاتور ff را از نو خود به خود وارد می‌کند. در ؟؟ (زمانی که در مورد تیره‌بندی صحبت خواهیم کرد) خواهیم دید که لیگاتورها، داخل یک کلمه تیره‌بندی شده که درهم‌رفتگی‌های آشکار (explicit kerns) ندارد، قرار داده می‌شوند و اصلاح ایتالیک یک درهم‌رفتگی آشکار است. اما مقدار فاصله‌ای که اصلاح ایتالیک قرار می‌دهد ممکن است خیلی زیاد باشد (مخصوصاً در یک قلم ایتالیک)، بنابراین استفاده از `shelf{\kern0pt}ful` همیشه بهترین راه است.

پرسش ۲.۵. چگونه می‌توانیم سه فاصله خالی پشت سر هم بدون استفاده از ␣\␣\␣\ داشته باشیم؟

پاسخ. می‌توانیم از ␣{␣}␣ یا ␣{}␣{}␣ و سایر موارد مشابه استفاده کنیم. قالب plain، همچنین ماکرویی به نام \space دارد. بنابراین می‌توانیم

```
\space\space\space
```

را تایپ کنیم. از آنجایی که \space\space\space فاصله‌ها را با استفاده از «فاکتور فاصله» (space factor) فعلی تنظیم می‌کنند (این مطلب در فصل‌های بعدی این نوشتار توضیح داده خواهد شد)، حتماً معادل \␣\␣\␣ نیستند.

TEX همچنین از گروه‌بندی برای یک منظور کاملاً متفاوت دیگر (تعیین کردن اینکه چه مقدار از متن شما از دستورات خاصی پیروی کند) استفاده می‌کند.

مثال ۳.۵. اگر می‌خواهید چیزی را در یک سطر وسط‌چین کنید، می‌توانید از دستور \centerline که در قالب plain تعریف شده است، استفاده کنید:

```
\centerline{This information should be centered.}
```

گروه‌بندی در تعدادی از دستورالعمل‌های پیچیده‌تر TEX به کار می‌رود و داشتن گروه‌های تو در تو (گروه‌هایی که داخل گروه‌های دیگری هستند که خودشان داخل گروه‌های دیگری هستند) میسر است. هر چند گروه‌بندی پیچیده معمولاً در نوشته‌های متداول لازم نیستند بنابراین نیازی نیست نگران گروه‌بندی‌های پیچیده باشید فقط فراموش نکنید که هر گروهی را که شروع کرده‌اید، تمامش کنید چون یک { فراموش شده ممکن است مشکل‌ساز باشد.

مثال ۴.۵. این یک نمونه از کاربرد دو گروه تو در تو (یک گروه داخل گروه دیگری) است:

```
\centerline{This information should be {\it centered}.}
```

همانطور که ممکن است انتظار داشته باشید TEX یک سطر وسط‌چین که همچنین متن ایتالیک هم دارد، ایجاد می‌کند:

This information should be *centered*.

اجازه بدهید به دقت به این مثال نگاه کنیم. دستور \centerline بیرون آکولادها (curly braces) قرار دارد در صورتیکه دستور \it داخل آکولادها (curly braces) قرار دارد. چرا این دو مورد با هم فرق دارند؟ و یک مبتدی چگونه می‌تواند یاد بگیرد که به خاطر داشته باشد کدام دستور بیرون آکولادها (curly braces) قرار می‌گیرد و کدام دستور داخل آکولادها (curly braces) قرار می‌گیرد؟

\centerline دستوری است که تنها روی چیزی که بلافاصله بعد از خودش قرار می‌گیرد، به کار برده می‌شود بنابراین شما باید دور متنی که می‌خواهید آن را وسط‌چین کنید، آکولاد (barces) قرار دهید (مگر اینکه متن شما از یک نشانه یا دستور تشکیل شده باشد). به عنوان مثال برای وسط‌چین کردن لوگوی TEX در یک سطر، کافی است

```
\centerline\TeX
```

را تایپ کنیم اما برای وسطچین کردن عبارت TEX has groups به آکولاد (barces) نیاز خواهیم داشت:

```
\centerline{\TeX\␣has␣groups}
```

از طرفی دیگر، \it دستوری است که فقط «قلم فعلی را تغییر می‌دهد» و بدون اینکه متن جلوی خودش رو ببیند، عمل می‌کند بنابراین حداقل به طور بالقوه، هر متنی را که بعد از خودش قرار بگیرد تحت تأثیر قرار می‌دهد. آکولادها (braces) دستور \it را احاطه می‌کنند تا تغییر قلم را به یک ناحیه محلی، محدود کنند. به عبارت دیگر، دو جفت آکولاد (barces) در این مثال کارکرد متفاوت دارند:

- یکی برای در نظر گرفتن چندین کلمه متن به عنوان یک چیز واحد، به کار می‌رود.
- در صورتیکه دیگری یک ساختار کنده‌ای محلی را فراهم می‌کند.

پرسش ۳.۵. اگر متن زیر را تایپ کنیم، چه اتفاقی می‌افتد؟

```
\centerline{This information should be {centered}.}
\centerline So should this.
```

پاسخ. در مورد اول دقیقاً همان نتیجه‌ای را می‌گیریم که بدون وجود آکولاد (braces) داخلی می‌گرفتیم به این خاطر که ما از گروه‌بندی برای تغییر قلم یا کنترل کردن فاصله یا هر چیز دیگری، استفاده نکرده‌ایم. اگر ما وقتمان را برای ایجاد گروه‌هایی که هیچ دلیل خاصی ندارند، بیهوده تلف کنیم، از نظر TEX اشکالی ندارد.

اما در مورد دوم، آکولادهای لازم فراموش شده‌اند. حرف S به تنهایی در یک سطر وسطچین می‌شود و در سطر بعد پاراگرافی وجود خواهد داشت که با عبارت .o should this شروع می‌شود.

پرسش ۴.۵. اگر متن زیر را تایپ کنیم، چه اتفاقی می‌افتد؟

```
\centerline{This information should be \it centered.}
```

پاسخ. همان نتیجه‌ای را می‌گیریم که اگر یک جفت آکولاد دیگر دور \it centered وجود داشت، می‌گرفتیم با این استثنا که نقطه هم با قلم ایتالیک حروف‌چینی می‌شود (هر دو نقطه تقریباً شکل یکسانی دارند). قلم \it بعد از پایان \centerline اثری نخواهد داشت اما این یک چیز همرویداد است: TEX از آکولادها استفاده می‌کند تا تشخیص بدهد چه متنی قرار است وسطچین شود اما سپس این آکولادها را حذف می‌کند. دستور \centerline در قالب plain بصورت زیر تعریف شده است:

```
\def\centerline#1{\line{\hss#1\hss}}
```

بنابراین همانطور که در تعریف دستور \centerline مشاهده می‌کنید، دستور \centerline متن بدون آکولاد حاصل را داخل یک گروه دیگر قرار می‌دهد و به همین دلیل است که دستور \it بعد از پایان دستور \centerline ناپدید می‌شود (اگر متوجه این موضوع نمی‌شوید، فقط آکولادها را در حالت‌های پیچیده فراموش نکنید، و مشکلی نخواهید داشت).

پرسش ۵.۵. دستوری به نام \ital تعریف کنید تا به جای

```
{\it text\/}
```

بتوانیم از {\ital{text} استفاده کنیم. مزایا و معایب دستور \ital در مقابل دستور \it چیست؟

پاسخ.

```
\def\ital#1{{\it#1\/}}
```

مزایا: دستور \ital برای یاد گرفتن آسانتر است، مانند دستور \centerline کار می‌کند، و احتیاجی نیست که به خاطر داشته باشیم، باید از اصلاح ایتالیک استفاده کنیم.

معایب: برای اینکه از اصلاح ایتالیک قبل از ویرگول (comma) یا نقطه استفاده نکنیم، احتمالاً باید یک دستور دیگر را هم یاد بگیریم. به عنوان مثال، با استفاده از دستور \nocorr که بصورت زیر تعریف شده است:

```
\def\nocorr{\kern0pt }
```

می‌توانیم ,{\ital{comma} or \ital{period\nocorr} را تایپ کنیم. راه دیگر که برای اجتناب از اصلاح ایتالیک، ویرگول (comma) یا نقطه را با قلم ایتالیک حروف‌چینی کنیم، ظاهر به این خوبی ندارد. از آنجایی که برای از نو اسکن کردن آرگومان دستور \ital، تمام متن برای آرگومان دستور \ital باید در حافظه خوانده شود، یک دنباله طولانی از ایتالیک‌ها برای TEX کارآمد نخواهد بود.

در فصل‌های بعدی این نوشتار، در مورد بسیاری از دستورات بدوی TEX صحبت خواهیم کرد که برای آن‌ها مکان گروه‌بندی مهم است.

مثال ۵.۵. اگر یکی از پارامترهای داخلی TEX، داخل یک گروه تغییر کند،در پایان گروه محتوای قبلی این پارامتر را خواهیم داشت.

هر چند، گاهی اوقات می‌خواهیم تعریفی را انجام بدهیم که فراتر از گروه فعلی‌اش باشد. برای انجام این کار، قبل از آن تعریف، دستور \global را قرار می‌دهیم که درحقیقت پیشوند تعریفی است که می‌خواهیم انجام دهیم.

مثال ۶.۵. TEX شماره صفحه فعلی را در درجینه‌ای (register) به نام \count0 قرار می‌دهد و روالی (routine) که یک صفحه تولید می‌کند، می‌خواهد شماره صفحه را افزایش دهد. روال‌های خروجی (ouput routines) همیشه با محصور شدنشان در گروه‌ها، محافظت می‌شوند تا تصادفاً موجب به هم ریختن باقی TEX نشوند. اما اگر تغییرات \count0 داخل گروهی که روال خروجی (output routine) در آن محصور شده است، قرار بگیرند فقط مختص به این گروه خواهند بود و بعد از اتمام گروه ناپدید می‌شوند. دستور

```
\global\advance\count0 by 1
```

این مشکل را حل می‌کند. این دستور مقدار \count0 را افزایش می‌دهد و باعث می‌شود تا این مقدار در پایان روال خروجی (output routine) موجود باشد.

به طور کلی، دستور \global باعث می‌شود تا تعریفی که بلافاصله بعد از آن قرار می‌گیرد، نه تنها به داخلی‌ترین گروه بلکه به همه گروه‌های موجود تعلق داشته باشد.

پرسش ۶.۵. فرض کنیم که \c معادل =\count1، \g معادل =\global\count1، و \s معادل \showthe\count1 باشند. کد زیر چه مقادیری را نشان خواهد داد؟

```
{\c1\s\g2{\s\c3\s\g4\s\c5\s}\s\c6\s}\s
```

پاسخ.

```
{1 {2 3 4 5} 4 6} 4
```

یک راه دیگر برای بدست آوردن ساختار کنده‌ای با TEX، استفاده از دستورات بدوی \begingroup و \endgroup است. این دستورات شروع کردن گروه با یک دستور و پایان کردن گروه با دستور دیگری را آسان می‌کنند. متنی که در واقع TEX بعد از آنکه دستورات گسترش یافته‌اند، اجرا می‌کند، باید گروه‌های تو در تو کاملاً درستی داشته باشند (گروه‌هایی که با هم تداخل نداشته باشند).

مثال ۷.۵.

```
{ \begingroup } \endgroup
```

مشروع (قانونی) نیست.

پرسش ۷.۵. دستورات

\beginthe⟨block name⟩ و \endthe⟨block name⟩

که یک ساختار کنده‌ای «نامی» (named) می‌دهند، را تعریف کنید. به عبارت دیگر،

```
\beginthe{beguine}\beginthe{waltz}\endthe{waltz}\endthe{beguine}
```

باید مجاز باشد اما

```
\beginthe{beguine}\beginthe{waltz}\endthe{beguine}\endthe{waltz}
```

نباید مجاز باشد.

پاسخ. نکته کلیدی در اینجا این است که باید آرگومان دستور \beginthe و آرگومان دستور \endthe (که هر دو آرگومان، نام هستند) را با هم مقایسه کنیم. سپس چنانچه این دو نام یکی بودند دستور \endgroup را قرار دهیم اما اگر این دو نام یکسان نبودند اعلام خطا کنیم.

```
\def\beginthe#1{\begingroup\def\blockname{#1}}
\def\endthe#1{\def\test{#1}%
  \ifx\test\blockname\endgroup
  \else\errmessage{You should have said
    \string\endthe{\blockname}}\fi}
```

نمایه

کتاب‌نامه

[1] D.E. Knuth. *The TEX book*. Addison-Wesley, reprinted with corrections 2012.