

4. *Spacing within formulas.*  $\TeX$ 을 이용해서 수식을 조판할 때,  $\TeX$ 은 내부적으로 수식 내의 간격을 조절하는 규칙을 가지고 있고, 그 규칙대로 간격을 기계적으로 자동 조절 해줍니다. 사실 그 규칙은 단순하기 그지없습니다.  $\TeX$ 의 수식 내 간격 조절 규칙에 대해서는 후반부에서 자세히 알아보겠습니다. 하지만 아무리 수식 조판의 달인이라는  $\TeX$ 이라 하더라도 수 많은 수식을 우리의 입맛에 맞게 처리할 수 있을까요? 그것도 그 단순한 규칙을 가지고 모두 자동적으로? 실제로  $\TeX$ 이 규칙에 의해 기계적으로 처리하다보니, 가끔은 우리들이 보기에 약간 어색하게 보이는 구석이 있는 것도 사실입니다. 이런 경우에 수고스럽겠지만, 우리들이  $\TeX$ 을 도와주어야 합니다.  $\TeX$ 도 자신이 우리들의 도움을 받아야 한다는 것을 이미 알고 있기 때문에 친절하게도, 비교적 넓은 간격 조절에 이용되는 `\quad` 나 `\qquad` 말고도, 세밀한 간격을 조절할 수 있는 도구들을 몇가지 미리 마련해 놓았습니다.

수식 내의 세밀한 간격 조절을 위해서  $\TeX$ 이 미리 준비해 두었다는 것들은 바로 *thin spaces*, *medium spaces*, *thick spaces* 라고 불리는 간격의 기본 단위들입니다. 이 기본 단위들을 자세히 알아보기에 앞서, 도대체 어떤 것인지 감을 잡기 위해서 아래와 같은 수식을 살펴봅시다.


The Fibonacci numbers satisfy  $F_n = F_{n-1} + F_{n-2}$ ,  $n \geq 2$ .

등호 '='와 부등호 '>'의 앞 뒤에 있는 간격이 바로 *thick spaces* 이고, *medium spaces*는 + 기호 앞 뒤에서 찾아 볼 수 있습니다. *Thin spaces*는 좀 작아서 알아보기 힘듭니다. 'loglog'와 'log log'의 차이가 눈에 보이십니까? 그 차이가 바로 *thin space*입니다. 한 문단에서 단어들 사이의 간격은 바로 이 *thin space*의 거의 두배에 해당하는 간격입니다.

앞에서 이미 언급했듯이,  $\TeX$ 은 자신만의 기본적인 규칙을 가지고 *thin spaces*, *medium spaces*, *thick spaces*를 이용하여 수식 간격을 자동적으로 조절합니다. 하지만 우리들이 원하기만 한다면, 이 간격들을 직접 넣을 수도 있습니다. 넣을 때는 다음과 같은 명령어(*contro sequence*)를 이용합니다.

```
\, thin space (보통 quad의 1/6);
\> medium space (보통 quad의 2/9);
\; thick space (보통 quad의 5/18);
\! negative thin space (보통 quad의 -1/6).
```

우리들이 수식을 조판할 때, 대부분의 경우는  $\TeX$ 에게 모든 것을 맡기면 됩니다. 우리들이 직접 위의 네 개의 명령어를 사용할 기회는 매우 드물고, 그것도 일단은  $\TeX$ 이 만들어 낸 결과물을 보고서 넣을지 말지를 판단해야 합니다.

 비교적 넓은 간격을 조절 할때 사용되는 `\quad`는 수식의 스타일이나 그 수식에 사용되는 수학 폰트에 따라서 변하지 않습니다. 하지만 *thin spaces*, *medium spaces*, *thick spaces* 들은 수식의 스타일이나 폰트가 커지거나 작아지면 그에 맞추어 그 간격이 좁아지거나 넓어집니다; 이는 네 가지 기본 간격의 단위들이 수식 간격 조절을 위해서 특별히 고안된 `\muglue`라는 것을 이용해서 정의 되기 때문입니다. 사실 `\muglue`도 특별한 것이 아니어서 `\muglue`를 정할 때도 다른 보통의 `glue`를 정의 하듯이 하면 됩니다. 다만, `pt`, `cm` 같은 단위를 이용하는 것이 아니라 `math unit` 라고 하는 'mu' 만을 이용 해야 한다는


*The basic elements of space that  $\TeX$  puts into formulas*

제18장 앞 부분에서 설명한 내용입니다.


것이 다릅니다. 예를 들면, 부록 B 에서 다음과 같은 정의를 찾아 볼 수 있습니다.

```
\thinmuskip = 3mu
\medmuskip = 4mu plus 2mu minus 4mu
\thickmuskip = 5mu plus 5mu
```

위에서 볼 수 있듯이, thin spaces는 plain TeX에서는 늘어나거나 줄어들지 못하고, medium spaces는 조금 늘어날 수 있고, 그 간격이 0이 되도록 줄어 들 수 있습니다. Thick space는 많이 늘어 날 수는 있지만 줄어 들지는 못합니다.

 하나의 em은 family 2 (the math symbols family) 에서는 18 mu에 해당합니다. 다른 말로 하면, `\textfont 2`는 display 와 text styles 에서 mu에 대한 em 값을 정합니다. `\scriptfont 2`은 em script 크기에 대해서 정하고, `\scriptscriptfont 2`는 scriptscript 크기에 대해서 em을 정합니다.

em은 대문자 'M'의 크기에 해당하는 간격입니다.

 수식 내에 math glue를 넣고자 할때는 '`\mskip<mu>`'와 같은 명령어를 이용합니다. 예를 들어, '`\mskip 9mu plus 2mu`'는 이 명령어가 사용되고 있는 시점에서 쓰이고 폰트의 크기에 맞는 em 크기의 절반에 해당하는 간격을 넣는데 그 간격은 약간 늘어날 수도 있습니다. 부록 B를 보면 '`\,`'는 '`\mskip\thinmuskip`'를 뜻합니다. 비슷한 맥락으로, '`\mkern`'를 사용할 수도 있는데, 이때는 '`\kern`'이 그러하듯이 간격은 늘어나거나 줄어들지 못합니다. '`\mkern18mu`'는 이 명령어가 사용되는 시점의 환경에서 수평으로 em에 해당하는 간격을 넣어 줍니다. TeX은 `\mskip` 과 `\mkern`는 mu하고만 어울리려고 고집을 피웁니다. 그래서 `\hskip` 과 `\kern`는 수식에서도 사용될 수 있는 명령어이지만, 그들에게 mu로 된 길이 단위를 줄 수 없습니다.


미적분 수식 기호가 들어 있는 수식에서  $dx$  와  $dy$  앞에 thin space를 넣으면 보기가 훨씬 좋다고 합니다. TeX이 이것을 자동적으로 해주면 좋으려면, 해주지는 않는답니다. 따라서 그러한 수식을 조판할 때에 수식 조판에 능숙한 분들이 하듯이, 아래의 예제처럼 '`\,`'를 넣어주는 센스를 보여주시면 좋습니다.

<i>Input</i>	<i>Output</i>
<code>\int_0^\infty f(x)\,dx</code>	$\int_0^\infty f(x) dx$
<code>\,dx-x\,dy</code>	$y dx - x dy$
<code>\,dy=r\,dr\,d\theta</code>	$dx dy = r dr d\theta$
<code>\,dy/dx</code>	$x dy/dx$

위의 마지막 예에서 볼 수 있듯이 '`\,`' 뒤에 '`\,`'가 없다는 것도 눈여겨 두시기 바랍니다. 아래와 같은 경우에도 '`\,`'가 필요없는데,

`\int_1^x \frac{dt}{t}`

이는 분수에서  $dt$  가 분자로써 홀로 사용되었기 때문입니다.

 수식 중에 길이, 무게등 물리적 단위들이 사용될 때는 반드시 로마체로 쓰여져야 하고, 숫자와 그 단위 사이에 thin space가 있어야 합니다.

<code>\,mi/hr</code>	55 mi/hr
<code>\,m/sec^2</code>	$g = 9.8 m/sec^2$
<code>\,ml,cc</code>	1 ml = 1.000028 cc



수식에서 계승(factorial)을 나타낼 때 사용되는 느낌표 다음에 오는 문자가 글자, 숫자 혹은 왼쪽 괄호라면 느낌표 다음에도 thin space가 사용되어야 합니다.

$$\$(2n)!/\bigl(n!\,(n+1)!\bigr)\$ \quad (2n)!/(n!(n+1)!)$$

$$\$\$52!\over13!\,,13!\,,26!\}\$\$ \quad \frac{52!}{13!13!26!}$$

이러한 경우들 외에도, 우리들은 종종 수학 기호들이 너무 촘촘하게 묶인 수식들을 만나기도 하고, 너무 듬성듬성 묶인 수식도 만나기도 합니다. 이들은 대개 운 없게도 수학 기호들 사이에 뭐가 잘 맞지 않아서 발생한 경우입니다. 여러분이 편집하고 있는 수식의 결과를 직접 눈으로 확인하기 전에는 이런 결점을 미리 예측하는 것은 거의 불가능합니다. 따라서 일단 TeX이 조판한 결과를 보고 필요에 따라 ‘\,’ 나 ‘\!’를 넣어주어 세밀한 조정을 통해서 보다 멋진 수식을 조판해야 합니다. 제공된 기호나 여러개의 적분 기호가 겹친 경우가 바로 세밀한 조정이 필요한 대표적인 경우입니다. 아래의 예들을 한 번 살펴 봅시다:


$\$\sqrt2\,,x\$$	$\sqrt2x$
$\$\sqrt{\,,\log x}\$$	$\sqrt{\log x}$
$\$0\bigl(1/\sqrt n\,,\bigr)\$$	$O(1/\sqrt n)$
$\$[\,,0,1)\$$	$[0,1)$
$\$\log n\,,(\log\log n)^2\$$	$\log n(\log\log n)^2$
$\$x^2\!/2\$$	$x^2/2$
$\$n/\!\log n\$$	$n/\log n$
$\$\Gamma_2+\Delta^{\!2}\$$	$\Gamma_2 + \Delta^2$
$\$R_i^{\!j}\!_{\!k}\!_l\$$	$R_i^{jkl}$
$\$\int_0^x\!\int_0^y dF(u,v)\$$	$\int_0^x \int_0^y dF(u,v)$
$\$\$\int\!\!\!\int_D dx\,,dy\$\$$	$\iint_D dx dy$

위의 각각의 경우에서 \,, 나 \! 를 빼버리면, 수식들이 어딘가 어색하게 보일 것입니다.



위의 예에서 thin space가 필요한 경우는 공교롭게도 수식 기호들 사이에 궁합이 잘 안맞는 수식들이 함께 쓰여서 발생한 경우가 대부분입니다. 예를 들어  $\$x^2/2\$$ 에 사용된 윗첨자는 슬래쉬와 궁합이 맞지 않는데, 그 이유는 윗 첨자와 슬래쉬 사이에 약간의 공간이 생기게 되기 때문입니다. ( $x^2/2$ ): 이 경우에는 negative thin space로 그 간격을 조금 좁히면 됩니다. 마찬가지로,  $\$\sqrt{\,,\log x}\$$ 에 positive thin space가 사용된 이유는 제공된 기호와 로그 기호 사이의 궁합이 문제가 된 경우로 키가 큰 로그 기호와 제공된 기호가 함께 사용되면 너무 딱 붙어서 이상한 기호 처럼 부여서 로그 기호 앞에 약간의 간격이 필요했기 때문입니다. 그러나 다음의 두가지 예는 TeX이 수학을 잘 알지 못해서 생기는 경우로 TeX으로써는 어쩔 수 없는 경우라서 반드시 여러분이 TeX을 도와주어야 하는 경우입니다. (1) 수식  $\$\log n(\log\log n)^2\$$  에서, TeX은 왼쪽 괄호 앞에 필요한 약간의 간격을 넣지 않았습다. 왜냐하면, 수식  $\$\log n(x)\$$  는 로그 기호와 왼쪽 괄호 사이에 간격이 필요없는데, TeX은 이 두 수식을 동일한 경우라고 판단하기 때문입니다. (2) 수식  $\$n/\log n\$$ 에서, TeX은 log 앞에 넣지 않아야 할 간격을 자동으로 넣어줍니다. 왜냐하면, TeX의

입장에서 보면 슬래시는 보트의 수학 기호로 취급되기 때문이고, 보통의 수식 기호와 log 같은 연산자 사이에 약간의 간격이 들어간다는 T<sub>E</sub>X의 규칙 때문에 그렇습니다.


 수식 내에서 간격 조절을 위해서 T<sub>E</sub>X이 사용하는 규칙은 매우 단순합니다. 하나의 수식은 내부적으로 math list로 변환되는데, 이 math list는 여덟 가지 기본 타입의 원소로 구성됩니다: Ord (ordinary), Op (large operator), Bin (binary operation), Rel (relation), Open (opening), Close (closing), Punct (punctuation), and Inner (a delimited subformula). 다른 종류의 원소들, 예를 들면, `\overline`, `\mathaccent`, `\vcenter` 들은 모두 Ord 타입이고, 분수는 Inner 타입입니다. 아래에 나오는 표는 이웃하는 원소들 사이에 어떤 간격을 넣어야하는지를 결정할 때 T<sub>E</sub>X이 참조하는 표입니다.

*A formula is converted to a math list as described at the end of Chapter 17*

*Right atom*

	Ord	Op	Bin	Rel	Open	Close	Punct	Inner
Ord	0	1	(2)	(3)	0	0	0	(1)
Op	1	1	*	(3)	0	0	0	(1)
Bin	(2)	(2)	*	*	(2)	*	*	(2)
<i>Left atom</i> Rel	(3)	(3)	*	0	(3)	0	0	(3)
Open	0	0	*	0	0	0	0	0
Close	0	1	(2)	(3)	0	0	0	(1)
Punct	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
Inner	(1)	1	(2)	(3)	(1)	0	(1)	(1)

여기서 0, 1, 2, 3은 각각 no space, thin space, medium space, thick space를 나타냅니다. 표에서 괄호 안에 있는 것은 간격은 그 간격은 display 와 text styles 에서만 사용될 수 있다는 뜻입니다. 예를 들면, Rel행과 Rel열에 나오는 대부분의 간격은 ‘(3)’입니다; 이는 ‘=’와 같은 관계 기호 앞, 뒤에는 thick spaces가 나온다는 의미입니다. 이 간격이 subscripts에 나올때는 아무런 간격도 들어가지 않습니다. 표를 자세히 살펴보면, 일부는 ‘\*’로 나타냅니다; 이 경우는 결코 발생하지 않는 경우를 나타냅니다. 왜냐하면 이항 연산자 앞과 뒤에는 그 연산자에 의미적으로 맞는 변수나 수학 기호가 나와야지, 영동한 기호가 나오지 않는다는 의미입니다. 부록 G는 math list가 어떻게 horizontal list로 변환 되는지 자세히 설명합니다; 이 변환은 당연히 T<sub>E</sub>X이 수학 모드(math mode)에서 빠져나오자마자 끝나고, 바로 그순간 inter-atomic 간격 조절이 들어갑니다.

 예를 들어, 아래와 같은 displayed 수식 표현은

$$x + y = \max\{x, y\} + \min\{x, y\}$$

다음과 같은 원소들로 변환되는데,

$$\text{Op} \text{ Bin} \text{ Op} \text{ Bin} \text{ Ord} \text{ Rel} \text{ Op} \text{ Open} \text{ Ord} \text{ Punct} \text{ Ord} \text{ Close} \text{ Bin} \text{ Op} \text{ Open} \text{ Ord} \text{ Punct} \text{ Ord} \text{ Close}$$

이는 각각 Ord, Bin, Ord, Rel, Op, Open, Ord, Punct, Ord, Close, Bin, Op, Open, Ord, Punct, Ord, Close 타입입니다. 따라서, 앞의 표에 의해서 간격은 다음과 같이 주어지고,

$$\text{Ord} \backslash \text{Bin} \backslash \text{Ord} \ ; \ \text{Rel} \ ; \ \text{Op} \ \text{Open} \ \text{Ord} \ \text{Punct} \ \backslash, \ \text{Ord} \ \text{Close} \ \backslash \text{Bin} \ \backslash \text{Op} \ \text{Open} \ \text{Ord} \ \text{Punct} \ \backslash, \ \text{Ord} \ \text{Close}$$


최종 결과는 다음과 같습니다.

$$x \text{ Bin} \text{ Op} \text{ Bin} \text{ Ord} \text{ Rel} \text{ Op} \text{ Open} \text{ Ord} \text{ Punct} \text{ Ord} \text{ Close} \text{ Bin} \text{ Op} \text{ Open} \text{ Ord} \text{ Punct} \text{ Ord} \text{ Close}$$


즉,

$$x + y = \max\{x, y\} + \min\{x, y\} .$$

이 예제는 아랫 첨자나 윗 첨자를 포함하고 있지는 않지만, 아랫 첨자나 윗 첨자 들은 그저 단순히 간격없이 각 원소들이 서로 서로 붙어서 어울립니다.

 Plain TeX macro인 `\bigl`, `\bigr`, `\bigrm`, `\big`는 모두 동일한 수학 기호를 만들어 내는데, 이들 사이의 유일한 차이점은 이들이 서로 다른 타입에 속하기 때문에 서로 다른 간격을 만들어 낸다는 것입니다: `\bigl`은 Open, `\bigr`은 Close, `\bigrm`은 Rel, `\big`은 Ord 원소를 만들어 냅니다. 반면에, 수식이 `\left` 와 `\right` 사이에 나타날 때, 그 수식은 Inner 타입이 됩니다. 따라서 `\left` 와 `\right`로 둘러 싸인 수식은 `\bigl` 와 `\bigr`로 둘러 쌓일때 보다 그 수식과 괄호 사이에 좀 더 넓은 간격을 갖게 됩니다. 예를 들어, Inner (from `\left`) 앞에 나오는 Ord는 thin space 간격을 갖게 되지만, Open (from `\bigl`) 앞에 나오는 Ord는 그렇지 않습니다.

영어 'delimiter'를 적당히 번역한 단어를 몰라서 그냥 단순히 '괄호' 혹은 그에 상응하는 수학 기호로 번역합니다.

 TeX의 간격 조절 규칙은 가끔 엉터리 일때가 있습니다. 수식 내에 'l' 와 'r' 가 사용될 때가 그때입니다. 이는 l 와 r 가 괄호(delimiters)가 아닌 Ord 타입으로 취급되기 때문입니다. 예를 들어, 다음 수식을 살펴봅시다.

<code>\$ -x = +x \$</code>	$ -x  =  +x $
<code>\$\$\left -x\right =\left +x\right \$\$</code>	$ -x  =  +x $
<code>\$\$\lfloor-x\rfloor=-\lceil+x\rceil\$\$</code>	$\lfloor -x \rfloor = -\lceil +x \rceil$

첫번째 경우 간격 조절이 잘못되어 있습니다. 왜냐하면 TeX이 'l'를 Ord 타입으로 취급해서 'l' 와 'x' 를 더하는 수식으로 판단하기 때문입니다. 두번째 예에서 첫번째 경우의 잘못을 `\left` 와 `\right`를 사용해서 바로 잡았습니다. 세번째 예는 다른 delimiter들에 대해서는 앞에서와 같은 교정이 필요없다는 것을 보여줍니다. TeX은 이미 세번째 예에서 사용된 delimiter들이 opening 인지 closing인지를 알고 있기 때문입니다.