TEX 토큰과 확장: 입력 문자열 처리

김 도 현 (동국대)

2016. 1. 30.

2016 한국텍학회 정기 학술대회

입력 문자열 처리

Tokenization 입력 문자열을 읽어서 토큰열을 만든다.

Expansion 더이상 확장할 수 없을 때까지 토큰들을 다른 토큰들로 대치한다.

Tokenization

읽기의 시작

- 한 줄씩 읽어들인다.
- 줄끝문자<CR> or <LF>부터 모든 것들을 버린다.
- 줄끝의 스페이스 문자들도 버린다.
- 줄끝에 <CR>을 첨가한다.
  - 정확히는 \endlinechar (통상 <CR>) 첨가한다.
  - \endlinechar=-1 이면 아무 것도 첨가하지 않는다.

Category Code

0	\	escape	7	^	superscript
1	{	begin group	8	_	subscript
2	}	end group	10	␣	space
3	$	math shift	11	[A-Za-z]	letter
4	&	alignment tab	12	기타	other
5	<CR>	end of line	13	~	active
6	#	parameter	14	%	comment

문자열을 문자로

- ^^ (catcode 7) + ASCII 문자
  - ^^I → <TAB> 문자 $73(\texttt{I}) - 64 = 9$
  - ^^M → <CR> 문자 $77(\texttt{M}) - 64 = 13$
- ^^ + 16진수 소문자 두개
  - ^^b7 → U+00B7 (·) 가운뎃점
- XeTeX, LuaTeX
  - ^^^^ac00 → U+AC00 (가)
  - ^^^^^^020000 → U+20000 (𠀀)

문자열을 토큰열로

category code에 따라 문자열을 읽어서

문자 토큰	$\texttt{a}_{11}$, $\texttt{1}_{12}$, $\texttt{가}_{11}$, $\texttt{␣}_{10}$
명령 토큰	`hskip`, `par`
파라미터 토큰	`#1`, `#9`

으로 구성된 토큰열을 만든다.

스페이스 ␣

- catcode 10
- 줄 머리의 스페이스는 모두 사라진다.
- 스페이스 다음의 스페이스도 모두 사라진다.

※ `\newcount\n \n=1␣` 에서의 스페이스는 나중에 실행단계에서 사라진다.

줄끝문자 <CR>

- `^^M` (catcode 5)
- 이후의 모든 문자열을 버린다.
- 직전에 문자가 있었다면 스페이스를 삽입한다.
- 빈줄이었다면 `\par` 를 삽입한다.

※ 실행단계에서 수평모드 진행 중 수직모드 명령을 만날 때도 `\par`가 삽입된다. `a\vskip10pt`

※ 당연히 `\par`는 재정의될 수 있다.
`\def\par{\ifhmode\hfill ♥\fi\endgraf}`

주석문자 %

- catcode 14
- 이후의 모든 문자열을 버린다.
- 따라서 줄끝문자가 스페이스로 바뀌지 않는다.

```
\def\aa{%
  bb
}
```

단어명령 control word

`\letters`

- `\[A-Za-z]+`
- catcode 0 + catcode 11 ⋯
- 뒤따르는 스페이스는 모두 사라진다.

문자명령 control character

`\non-letter`

- `\[^A-Za-z]`
- catcode 0 + not catcode 11
- `\$` `\{` `\}` `\#` `\%` `\\` …
- 뒤따르는 스페이스는 존중된다.
- 단, `\␣` (control space) 이후의 스페이스는 사라진다.

파라미터

- catcode 6
- #1 ⋯ #9
- 명령——단어명령, 문자명령, 활성문자——확장시 인자를 지시한다.
  - { } 쌍에 둘러싸인 토큰열
  - 토큰 하나
- 확장시 인자와 인자 사이의 스페이스는 무시된다.

```
\def\a#1#2{#1(#2)}
\a {bbb} {ccc} → bbb(ccc)
\a b c → b(c)
```

문자열을 토큰열로: 예제

```
^^^^ac00\hskip 2 em    나% <LF>
<LF>
a~\% . <LF>
    <LF>
```

- 가$_{11}$ hskip 2_{12} ␣$_{10}$ e$_{11}$ m$_{11}$ ␣$_{10}$ 나$_{11}$
- par
- a$_{11}$ ~$_{13}$ %␣$_{10}$ ·$_{12}$ ␣$_{10}$
- par

(ko.TEX 부르지 않은 경우)

catcode 불변의 법칙

한 번 읽어들인 토큰의 catcode는 바뀌지 않는다.

```
\def\a#1{\begingroup\catcode`\$=12
  #1\endgroup}
\a{$50} ← Error

\def\a{\begingroup\catcode`\$=12 \b}
\def\b#1{#1\endgroup}
\a{$50} ← Good
```

\string : 문자 토큰 강제하기

- \string\hskip → $\_{12}$ h$_{12}$ s$_{12}$ k$_{12}$ i$_{12}$ p$_{12}$
- \expandafter\string\csname a b\endcsname → $\_{12}$ a$_{12}$ ␣$_{10}$ b$_{12}$
- \escapechar=-1 이면 $\_{12}$ 가 들어가지 않는다.
- \string~ → ~$_{12}$
- \def\a#1{\string #1}
  \a{$50} ← Good

Expansion

확장이란?

```
\def\aa{\bb 1} \def\bb{cc}
\aa2\relax
```

- aa 2_{12} relax
- bb 1_{12} 2_{12} relax
- c_{11} c_{11} 1_{12} 2_{12} relax

※ \relax는 확장 불가능한 원시명령이다.

확장 가능 expandable 한 것들 ——몇 가지 예

- 매크로
- 조건문
- \number, \romannumeral
  - \number"0D → $1_{12}\ 3_{12}$
- \string, \the
  - \the\linewidth → $260.29036\text{pt}_{(12)}$
- \csname ⋯ \endcsname
  - \csname a@b\endcsname → a@b

확장 안 되는 경우 ——몇 가지 예

- `\let\a\b`
- `\def\a{\b}`
- `\gdef\a{\b}`
- `\toks0={\a\b}`

- 이상 모두에서 `\a`, `\b` 모두 확장 안 됨
- `\meaning\a` → `macro:->\b`

확장 안 되는 경우 ——몇 가지 예

- \uppercase, \lowercase

```
\def\a{a}
\def\A{A}
\uppercase{\a b} → \a B → aB
\lowercase{\A B} → \A b → Ab
```

확장 안 되는 경우 ——몇 가지 예

- 조건문의 false 영역
  - `\iftrue` 확장되고`\else` 확장 안 되고`\fi`
  - `\iffalse` 확장 안 되고`\else` 확장되고`\fi`

```
\def\xx{xx\else yy\fi}
\ifx ** \xx ← Good
\ifx *a \xx ← Error
```

\edef, \xdef

- 인자를 완전히 확장하여 매크로를 정의한다.
- \edef\a{\b}, \xdef\a{\b}
  - \a 확장 안 됨. \b 는 확장

```
\count255=3
\def \a{\the\count255 }
\edef\b{\the\count255 }
\count255=4
\a → 4
\b → 3
```

\noexpand

- 다음 토큰을 잠시 \relax로 만든다.
- 따라서 다음 토큰의 확장을 방지한다.

```
\edef\x{%
  \def\xx{\the\count255}%
} ← Error
```

```
\edef\x{%
  \def\noexpand\xx{\the\count255}%
} ← Good
```

\the 토큰 변수

- \the\toks0
  - \edef 안에서 토큰 변수는 한 번만 확장된다.

```
\def\b{b}
\toks0={\b}
\edef\a{\b\the\toks0 }
\meaning\a → macro:->b\b
```

\expandafter

- 다음 토큰을 다음 다음 토큰보다 나중에 확장한다.
- 이때 다음 다음 토큰은 한 번만 확장된다.

```
\def\a{ab}
\fbox\a → ab
\expandafter\fbox\a → ab
```

\expandafter를 이용한 꼬리재귀

```
\def\dofor#1{%
  \ifx#1\stopfor
  \else
    \fbox{#1}
    \expandafter\dofor
  \fi
}
\dofor 12345\stopfor
```

\afterassignment

- 할당이나 정의를 먼저 수행한 후 다음 토큰을 확장한다.

```
\def\newfont{%
  \afterassignment\myfont\font\myfont
}
\newfont="Times" at 16pt
Typeset in Times.
```

\aftergroup

- 다음 토큰을 현재 그룹이 닫힌 후 확장한다.

  {\aftergroup\a\aftergroup\b}
  는 \a\b 와 같다.

- \aftergroup\aftergroup\aftergroup\a 는 두 단계 상위 그룹에서 \a 를 확장한다.

\protected — ε-TEX 원시명령

- 확장에 대한 저항력을 가진 robust 매크로를 만든다.

```
\def\a{a}
\protected\def\b{b}
\edef\x{\a\b}
\meaning\x → macro:->a\b
```

참고

- Donald E. Knuth. *The TEXbook*. Addison-Wesley. 1984.
- Victor Eijkhout. *TEX by Topic*. `texdoc texbytopic` or `http://eijkhout.net/texbytopic/`.
- Peter Breitenlohner. *The ε-TEX manual*. `texdoc etex`.