PlainTEX Verbatim

Benjamin T. Shepard

July 1, 2022

An inline verbatim such as `this is \inline` can be made with the code `|...|`. If visible spaces are desired, use `@|...|` instead, which will produce `this␣is␣\inline`. A display verbatim such as

```
this is \display
```

can be made with `||...||`. If visible spaces are desired, use `@||...||` instead, which will produce

`this␣is␣\display`.

Every normal TEX character (except `|`) as well as all of the special characters can be used inside this environment, as shown in the following example:

```
!$@$#%^&*()-+=\ /.,;'‘~th@<is$ ]\bye *is{ #a ’‘1\rho > *\tes<t* \undefined .??["
```

Normally, the pipe character `|` can only be used inside the display verbatim `||...||`. However, the macros `\makepipeother` and `\makepipeactive` turn the character `|` into an innocent character and back into an active character, respectively. This allows for use of the pipe literal in horizontal mode. There is also the macro `\pipe` which expands to

```
 \bgroup\string|\egroup
```

displaying the literal `|`. These macros can be used to, for example, execute

```
{\tt th\pipe{}s is a pipe}
```

or, as an alternative,

```
\makepipeother{\tt th|s is a pipe}\makepipeactive
```

which will each display `th|s is a pipe`. Of course, these macros cannot be used inside the `|...|` or `||...||` environments. The character `@` is able to be used normally; TEX code such as

```
this is a t@st
@

this is a test@
\bye
```

will compile as expected. These verbatim environments also obey spaces, lines, and blank lines; this allows for indented code insertion such as

```
void dfs(int p){
    if(o[p]) return;
    o[p]=1;

    c[p] = t++;
    for(int i=0; i<s[p].size(); ++i){
        dfs(s[p][i]);
    }
}
```

which will also compile as expected.