The graphviz package*

Derek Rayside ⟨drayside@uwaterloo.ca⟩
with contributions from Ralf Hemmecke ⟨ralf@hemmecke.de⟩
Kevin Mauser ⟨kbmauser@gmail.com⟩

April 5, 2024

1 Introduction

graphviz.sty is a LaTeX package for writing graphviz/dot/neato graphs inside of LaTeX documents. graphviz.sty was inspired by a feature that Daniel Jackson added to his tagger text markup tool.

graphviz is a freely available package for doing automated graph layout from AT&T Research, distributed under the Common Public License (CPL). graphviz includes the dot and neato programs, which read a textual description of a graph and produces a graphical rendering of it. Many different graphics formats, include PostScript, are supported.

There are two main web pages for the graphviz project:

- http://www.graphviz.org
- http://www.research.att.com/sw/tools/graphviz/

graphviz.sty is provided as-is, with no warranty or claim to fitness for any purpose, use at your own risk, etc. graphviz.sty is distributed under the LaTeX Project Public License.

2 Example

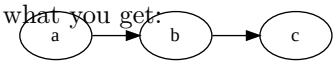
Put this in your document:

```
\digraph[scale=0.5]{abc}{rankdir=LR; a->b->c;}
```

Run these commands (only the first run needs `-shell-escape`):

```
latex -shell-escape main.tex
latex main.tex
```

And here's what you get:



*This document corresponds to graphviz v0.96, dated 2024/04/04.

3 Usage

`\digraph[⟨i⟩]{⟨n⟩}{⟨g⟩}` The `\digraph` (dot) and `\neatograph` (neato) commands take three arguments:

`[⟨i⟩]` parameters to the `\includegraphics` command that will include the PostScript file of the graph [this is optional]: eg, '`scale=0.5`'

`{⟨n⟩}` the name of the graph; a file `name.dot` is created, and a file `name.ps` is expected to be produced from dot: eg, '`MyGraph`'
`{⟨n⟩}` has to be a valid file name and a valid identifier name.

`{⟨g⟩}` the graph, specified in the dot/graphviz language:
eg, '`rankdir=LR; a->b->c;`'

4 Options

`singlefile` LaTeX has a small number of file handles (about 16 or so). So if you can't have too many digraphs in your tex file before you run out of file handles. The `singlefile` option is a work-around: it writes all of your digraphs to a single file (`tmpmaster.graphviz`), and then uses gvpr to split that file into individual dot files for processing by dot.

The GVPR commands are all written to a second file (`tmpmaster.gvpr`), which is executed once the `tmpmaster.graphviz` file has been closed.

gvpr does not seem to be packaged with the Windows version of dot.

```
\RequirePackage{kvoptions}
\RequirePackage{ifthen}
\RequirePackage{xkeyval}
\newif\ifsinglefile
\DeclareOptionX{singlefile}{
    \singlefiletrue
    \AtBeginDocument{% open a new file handle
        \newwrite\masterdotfile%
        \immediate\openout\masterdotfile=\@tmpdir0 tmpmaster.graphviz%
        \newwrite\mastergvprfile%
        \immediate\openout\mastergvprfile=\@tmpdir0 tmpmaster.gvpr}
    \AtEndDocument{% close the file
        % close the dot file and the gvpr file
        \immediate\closeout\masterdotfile%
        \immediate\closeout\mastergvprfile%
        % execute the gvpr file
        \immediate\write18{gvpr -f \@tmpdir0 tmpmaster.gvpr \@tmpdir0 tmpmaster.graphviz}%
        }
    }
```

`psfrag` The psfrag option uses the psfrag package to enable you to overlay TeX fragments over included postscript files, such as those generated via the `\digraph` command.

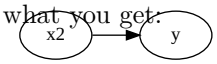
The ladot script from Brighten Godfrey uses Perl to extend the syntax of the graphviz language with TeX fragments, and psfrag to super-impose those fragments.

The psfrag option requires sed. psfrag seems to only work with dvips: ie, it is not compatible with pdflatex or dvipdfm. The PDF files produced by LaTeX/psfrag/ps2pdf seem to view ok with Acrobat, but not with gv. Oddly, the PS files produced this way work in gv.

Put this in your document:

```
\psfrag{x2}[cc][cc]{$x^2$}
\digraph{xy}{rankdir=LR; x2->y;}
```

And here's what you get:



PSfrag replacements

x^2

```
\newif\ifpsfrag
\DeclareOptionX{psfrag}{ \psfragtrue }
```

`ps` Tell Graphviz to generate Postscript files as output.

```
\newcommand{\@outext}{ps}
\newcommand{\@outextspace}{ps }
\DeclareOptionX{ps}{
    \renewcommand{\@outext}{ps}
    \renewcommand{\@outextspace}{ps }}
```

`pdf` Tell Graphviz to generate PDF files as output.

```
\DeclareOptionX{pdf}{%
    \renewcommand{\@outext}{pdf}%
    \renewcommand{\@outextspace}{pdf }}
```

`auxdir` Allow directing all generated files to auxiliary directory, whether or not tmpdir is used

```
\newcommand{\@auxdirisempty}{}
\newcommand{\@auxdirdefault}{./}
\newcommand{\@auxdir}{./}
\newcommand{\@auxdirO}{}
\DeclareOptionX{auxdir}{%
    \renewcommand{\@auxdir}{#1}
    \ifx\@auxdir\@auxdirdefault
        \renewcommand{\@auxdir}{./}
        \renewcommand{\@auxdirO}{\@auxdir}
    \else
        \ifx\@auxdir\@auxdirisempty
            \renewcommand{\@auxdir}{./}
            \renewcommand{\@auxdirO}{\@auxdir}
        \else
            \renewcommand{\@auxdir}{#1}
            \renewcommand{\@auxdirO}{./\@auxdir/}
        \fi
    \fi
}
```

`tmpdir` Write all generated files in `./tmp/` by default or in user-specified directory. Now compatible with using –aux-directory option when compiling document

```
\newcommand{\@tmpdirdefault}{tmp/}
\newcommand{\@tmpdir}{tmp/}
\newcommand{\@tmpdirO}{tmp/}
\DeclareOptionX{tmpdir}{%
    \renewcommand{\@tmpdir}{#1}
    \ifx\@tmpdir\@tmpdirdefault
        \renewcommand{\@tmpdir}{tmp/}
    \else
        \ifx\@tmpdir\@auxdirisempty
            \renewcommand{\@tmpdir}{}
        \else
            \renewcommand{\@tmpdir}{#1}
        \fi
    \fi
    \renewcommand{\@tmpdirO}{\@tmpdir}
}
```

Set the default options

```
\ExecuteOptionsX{tmpdir,auxdir,ps}
\ProcessOptionsX\relax % LaTeX class guide says it is wise to relax

%when using aux-directory set proper path for .dot file creation and accessing
\ifx\@auxdir\@auxdirisempty
    \ifx\@tmpdir\@auxdirisempty
        \renewcommand{\@tmpdirO}{./}
    \else
        \renewcommand{\@tmpdirO}{\@auxdirO\@tmpdir}
    \fi
\else
    \ifx\@tmpdir\@auxdirisempty
        \renewcommand{\@tmpdirO}{\@auxdirO}
    \else
        \ifx\@auxdir\@tmpdir
            \renewcommand{\@tmpdirO}{}
        \else
            \ifx\@tmpdir\@tmpdirdefault
                \renewcommand{\@tmpdirO}{\@tmpdir}
            \else
                \renewcommand{\@tmpdirO}{\@tmpdir}
            \fi
        \fi
    \fi
\fi
```

5 Implementation

5.1 Required Packages

This package requires graphicx to include PostScript renderings of graphs.

```
\RequirePackage{graphicx} %psfrag option
\RequirePackage{psfrag} %to process .ps into .pdf
\RequirePackage{auto-pst-pdf} %to process .ps into .pdf
\ifpsfrag \RequirePackage{psfrag} \fi
```

5.2 Command Implementation

`\digraph` This is the command the user uses for dot.

It is very important that this command is not defined with 3 parameters although it will be used with 3 parameters in the form `\digraph[OPTIONS]{FILENAME}{GRAPH}`. The reason is that the catcode for `^^M` must be changed *before* TEX reads the GRAPH argument.

The order of the command (first `\inputdigraph` then `\@digraph`) may look a bit odd, but it simplifies the code. In order to include the digraph, LATEX has to be run at least two times anyway. In the first run the file dot will be generated and only the second run the digraph will be included.

```
\newcommand{\digraph}[2][scale=1]{
  \inputdigraph[#1]{#2}{dot}%       % Include the generated ps/pdf.
  \@digraph{digraph}{#2}%           % Generate the .dot file.
}
```

`\neatograph` This is the command the user uses for neato. The syntax is the same as for `\digraph`.

```
\newcommand{\neatograph}[2][scale=1]{
  \inputdigraph[#1]{#2}{neato}%     % Include the generated ps/pdf.
  \@digraph{graph}{#2}%             % Generate the .dot file.
}
```

`\@digraph` Internal implementation.

The macro `\@digraph` prepares the actual output of the digraph to a file (which is done by `\@@digraph`) by a special treatment of the newline character. Before entering `\@@digraph`, the input newline character (`^^M`) is made active, and redefined to expand to `^^J`. Note that `\@digraph` has a `\begingroup` that is closed in `\@@digraph`.

The purpose of this is to preserve line breaks in the digraph.

```
\begingroup
  \catcode`\^^M=\active%
  \gdef\@digraph{\begingroup\catcode`\^^M=\active\def^^M{^^J}\@@digraph}%
\endgroup
```

`\@@digraph` Internal implementation.

The parameters of the macro \@@digraph are the TYPE, FILENAME and GRAPH of the initial \digraph[OPTIONS]{FILENAME}{GRAPH}. Note that if \@@digraph is entered the ^^M character is active. Thus every newline character (^^M) in the following macro is hidden through a % sign at the end of line.

```
\def\@@digraph#1#2#3{%
    \ifsinglefile% write the graph to the master file
        \expandafter\def\csname -\endcsname{\string\n}%
        \immediate\write\masterdotfile{#1 #2 {#3}}%
        \immediate\write\mastergvprfile{BEG_G { if ($.name == "#2") {writeG($G,"\@tmpdirO#2.d
    \else% open a new file handle
        \newwrite\dotfile%
        \immediate\openout\dotfile=\@tmpdirO#2.dot%
        \expandafter\def\csname -\endcsname{\string\n}%
        \immediate\write\dotfile{#1 #2 {#3}}%
        \immediate\closeout\dotfile%
    \fi%
% Here comes the closing \endgroup that closes the group opened in \@digraph.
    \endgroup}%
% Now ^^M is no longer active.
```

\inputdigraph This is usually only called by \digraph, but may be called by the user.

The purpose is to include the ps/pdf rendering of the graph if it exists, or to give instructions on how to generate it.

```
\newcommand{\inputdigraph}[3][scale=1]{
    % execute dot or neato (nb: requires latex -shell-escape)
    \immediate\write18{#3 -T\@outextspace -o \@auxdirO\@tmpdirO#2.\@outextspace \@auxdirO\@tm
    \IfFileExists{\@auxdirO\@tmpdirO#2.\@outext}{ % the postscript/pdf exists: include it
            \ifpsfrag
                % per the ladot 2.2 source code, psfrag has a problem with
                % graphviz 2.2, and some sed hackery is necessary to work around
                % Unix/Linux:
                \immediate\write18{sed -ibackup -e "s/xshow/pop show/g" \@auxdirO\@tmpdirO#2.
                % Windows:
                %\immediate\write18{get-content \@auxdirO\@tmpdirO#2.ps |  \%{\$_ -replace "x
            \fi
            \includegraphics[#1]{\@tmpdirO#2.\@outext}
        }
        % else: the postscript/pdf doesn't exist: tell the user how to create it
        {
            \fbox{ \begin{tabular}{l}
                The file \texttt{#2.\@outext} hasn't been created from \texttt{\@auxdirO\@tmp
                Run `\texttt{dot -T\@outextspace -o \@auxdirO\@tmpdirO#2.\@outextspace \\
                \@auxdirO\@tmpdirO#2.dot}' to create it. \\
                Or invoke \LaTeX\ with the \texttt{-shell-escape} option to have this done au
                \end{tabular}}
        }
}
```

5.3 Process

\digraph writes out a dot file, and then invokes dot on it.

Note: `\digraph` can only invoke dot if the LaTeX was invoked with the `-shell-escape` option, to enable execution of external programs. If you do not want to allow LaTeX to execute external programs, then you will have to invoke dot yourself. graphviz will also need to execute gvpr if the `singlefile` option has been selected, and sed if the `psfrag` option has been selected.

Here's a picture of the process (drawn with dot, naturally). The picture shows the process using dvips, but pdflatex is now also supported with the pdf option.