

noweavepp

Bob Matlin

<http://matlinsoftwareservices.com>

March 1, 2003

Contents

1	Introduction	2
2	The Plan	2
3	The Code	2
3.1	The Class	3
3.2	break	5
3.2.1	Long Lines	6
3.3	breakOneLine	7
4	Test	9

1 Introduction

nowavepp is a preprocessor for noweb documents, and noweb is a literate programming tool. Briefly, literate programming is a style of programming introduced by Donald Knuth that recognizes that programs are meant to be read both by humans and computers. It makes it easy to create a document that intersperses explanatory text and code. For more information on noweb, see <http://www.eecs.harvard.edu/~nr/noweb/>; for more information on literate programming, see <http://www.literateprogramming.com>.

The reason for this program is that when a noweb document is weaved into a finished document, code lines can run beyond the end of the visible page (at least when producing a latex document). To keep the original just as it was (because of coding requirements or just for my convenience) and still produce a readable document, I decided to preprocess the noweb source to break long code lines into multiple lines.

2 The Plan

To produce the desired output, we will read in one source line at a time. The only lines we really need to reformat are code lines. Code is identified by the start of a code tag, which looks like <<filename>>=. Once a code tag is spotted, we start looking at line lengths. If a line is too long, we break it into multiple lines by breaking at spaces if possible and adding a line continuation character, then moving the start of the next line over a bit.

To facilitate testing, the code will be implemented as a class. To use the class in a script, we will add a main that calls the class to do the work.

3 The Code

First we want to be able to call this as a script directly:

```
2  <nowavepp.pm 2>≡ 3>
   #!/usr/bin/perl
   use FileHandle;

   if ($ARGV[0] eq "main") {
       #this is a main program
       my($infile) = $ARGV[1];
       my($maxLength) = $ARGV[2];
       my($pp) = Nowavepp->new($infile,$maxLength);
       $pp->break();
   }
```

3.1 The Class

The first few portions of this section are fairly simple and will not be commented on outside of the code documentation.

```
3  <noweavepp.pm 2>+≡ <2 5>
    =head1 PACKAGE Noweavepp

    =head2 SYNOPSIS

        Noweavepp (noweave preprocessor) is intended to be used
        as a filter before calling noweave, part of the noweb
        collection of literate programming tools. It breaks
        code lines into shorter pieces that can be properly
        printed.

    =cut

    package Noweavepp;

    1;

    =head1 new

        Create a new Noweavepp.

    =over

    =item INPUT

        pkg - the package name (Noweavepp)

        infile - the name of the file to process

        maxLength - the maximum line length to allow

    =cut

    sub new($$$) {
        my($pkg) = shift(@_);
        my($infile) = shift(@_);
        my($maxLength) = shift(@_);

        if (! -e $infile) {
            print STDERR "(noweavepp) Cannot find $infile.\n";
            exit(1);
        }
    }
```

```
if ($maxLength <= 0) {
    print STDERR "(noweavepp) maxLength is $maxLength but must be >= \
        0.\n";
    exit(1);
}

my($fh) = new FileHandle($infile);
if (! defined($fh)) {
    print STDERR "(noweavepp) Cannot open $infile for reading.\n";
    exit(1);
}

my $r_pp = bless {
    "infileName" => $infile,
    "fh" => $fh,
    "maxLength" => $maxLength
}, $pkg;

return($r_pp);
}
```

3.2 break

break does the work of identifying long lines.

```
5  <noweavepp.pm 2>+≡                                     <3 6a>
    =head1 break

    =over

    =item SYNOPSIS

        break finds lines that are too long to be properly printed.
        It only modifies code lines since document lines are handled
        properly in the document.

    =cut

sub break($) {
    my($this) = shift(@_);

    my($l);
    my($incode) = 0;
    my($fh) = $this->{fh};
    my($maxLength) = $this->{maxLength};
    while ($l = $fh->getline()) {
        #only a single @ on a line should exit code section
        if ($incode && $l =~ /^@\s+$/) {
            print $l;
            $incode = 0;
            next;
        }

        if ($l =~ /^<<(\w|[\s])+>>/) {
            print $l;
            $incode = 1;
            next;
        }

        if (! $incode) {
            print $l;
            next;
        }
    }
}
```

3.2.1 Long Lines

At this point we know that this is a code section. If the current line is not too long, we just print it.

```
6a  <noweavepp.pm 2>+≡ <5 6b>
      if (length($1) < $maxLength) {
          print $1;
          next;
      }
```

Now we know the line is too long, so we will break it into pieces and print the result.

```
6b  <noweavepp.pm 2>+≡ <6a 7>
      my($initSp) = "";
      $1 =~ /[^\t\r\n\b]/g;
      $initSp = substr($1,0,pos($1)-1);
      $initSp .= " ";
      my($s) = $this->breakOneLine($1,$initSp,1);
      print $s;
  }

  close($fh);
}
```

3.3 breakOneLine

breakOneLine does the actual splitting and reformatting.

```
7  <noweavepp.pm 2>+≡                                     <6b 8a>

=head1 breakOneLine

    Breaks one line into two pieces if it is too long.
    Calls itself recursively, so at the end the line may
    be broken into more than two pieces.

=over

=input

    this - a Noweavepp object

    l - the line to break

    initSp - the initial space to prepend the new line

    firstTime - if this is the first pass through this line

=item RETURN

    The initial string broken into multiple lines.

=cut

sub breakOneLine($$$$) {
    my($this,$l,$initSp,$firstTime) = @_;

    my($l1) = $l;
    $l1 =~ s/\s+$/; #ignore trailing white space

    my($maxLength) = $this->{maxLength};

    if (length($l1) < $maxLength) {
        if ($firstTime) {
            return($l);
        } else {
            return($initSp . $l);
        }
    }

    my($b1) = "";
```

```

#check that there is a space;
my(@a) = split(/\s/,$1);
$1l = $1;

```

If the line has no spaces, it is just arbitrarily broken. This will not produce an optimum result (it might be nice to break at a punctuation mark, for instance), but this program is not being designed to produce an optimal output.

```

8a  <noweavepp.pm 2>+≡ <7 8b>
      if ($#a <= 0) {
          #we know there are no spaces, so just split
          my($s) = substr($1l,0,$maxLength);
          $bl .= $s . " \\n";
          $1l = substr($1l,$maxLength);
      } else {

```

Now we know there are spaces. We find the last space closest to the max length and break at that point. I do not know of a way to get the first match from the end, so we will walk through the matches to find the last space matched. If this is not the first time the line is being broken (i.e., if it is a substring of the original line), we prepend some space to visually separate it from a normal line.

```

8b  <noweavepp.pm 2>+≡ <8a
      #use white space to split;
      my($s) = substr($1l,0,$maxLength);
      my($p) = 0;
      while ($s =~ /\s+/g) {
          $p = pos($s);
      }
      $p = 0 if (! $p);
      $s = substr($1l,0,$p);
      $bl = $s . " \\n";
      $1l = substr($1l,$p);
  }

  $bl = $initSp . $bl if (! $firstTime);
  $bl .= $this->breakOneLine($1l,$initSp,0);
  return($bl);
}

```

4 Test

Sample input file.

```
9 <noweaveppExample.nw 9>≡
  <<someCode.pl>>=
    $exp->throw($exp->setMessage("Cannot open directory " . \
      $this->{dirName} . " for reading.));
    abcd efg hij klm nop qrs tuv wxy z 1abcd 1efg 1hij 1klm 1nop 1qrs \
      1tuv 1wxy 1z 2abcd 2efg 2hij
  Path: \
    pics1-atl!c01.atl2!newsfeeds-atl1!sjc70.webusenet.com!chi1.webusenet.c \
    om!news.webusenet.com!newsfeed-east.nntpserver.com!nntpserver.com!bord \
    er1.nntp.aus1.giganews.com!nntp.giganews.com!nntp3.aus1.giganews.com!n \
    ntp.jorsm.com!news.jorsm.com.POSTED!not-for-mail
```

```

10  <noweaveppTest.pm 10>≡
    package noweaveppTest;

    use base qw(Test::Unit::TestCase);
    use perlx::Exception;
    use noweavepp;

    1;

    sub new {
        my($this) = shift()->SUPER::new(@_);
        return($this);
    }

    sub set_up {
        my($tc) = shift(@_);
        my($pp) = Noweavepp->new("noweaveppExample.nw",60);
        $tc->{pp} = $pp;
    }

    sub testBreak {
        my($tc) = shift(@_);
        my($pp) = $tc->{pp};
        my($outfile) = "test/break.out";
        my($e) = open(OUTFILE,">$outfile");
        if ($e) {
            select(OUTFILE);
        }
        $pp->break();
        my($ref) = '<<someCode.pl>>=
            $exp->throw($exp->setMessage("Cannot open directory \
                " . $this->{dirName} . " for reading."));
            abcd efg hij klm nop qrs tuv wxy z 1abcd 1efg 1hij 1klm \
                1nop 1qrs 1tuv 1wxy 1z 2abcd 2efg 2hij
Path: \
    pics1-atl!c01.atl2!newsfeeds-atl1!sjc70.webusenet.com!chi1.w \
    ebusenet.com!news.webusenet.com!newsfeed-east.ntpserver.com \
    !ntpserver.com!border1.nttp.aus1.giganews.com!nttp.giganews \
    .com!nttp3.aus1.giganews.com!nttp.jorsm.com!news.jorsm.com.P \
    OSTED!not-for-mail
';
        select(STDOUT);

        close(OUTFILE);
        #read the generated output
        $e = open(INFILE,$outfile);

```

```
my($s) = "";
my($l);
while ($l = <INFILE>) {
    $s .= $l;
}
close(INFILE);
$ref =~ s/\s/SP/g;
$s =~ s/\s/SP/g;
my($v) = ($ref eq $s);
$tc->assert_equals(1,$v);
}
```