Batchgen - a batch file generator

Nick James

Revision History

Revision 1	30-7-2007
Revision 2	31-10-2008

Having noticed that this program has been consistently downloaded over the last 4 years, and in the belief that someone may actually be using it, I've tweaked the documentation. In addition the improvements to the (previously arcane) file upload process on sourceforge [https://sourceforge.net/projects/batchgen/] make it possible to upload the version 1.2 zip, as well as hosting it on my own site [http://www.njames.co.uk/desktop.html].

Revision 3	12-2-2010

Version 2.0 This has improved sub routine functionality, owing to improvements in the winXP command processor. This version may not do what you want if you are not using XP.

Table of Contents

Warning

This software contains known and unknown bugs: use it at your own risk.

batchgen is a program to generate dos batch files from a higher level language, making it easier to use boolean logic in batch files.

Introduction

One of the failings of DOS/command.com/cmd.exe is that the only logical operator is "not". The other big failing is that it can't add up. batchgen is an attempt to bring boolean logic to DOS. You need to be quite happy writing batch files before using batchgen, it doesn't quite write them for you. If you have experience of C life will be easier.

The big idea is that we use batchgen to do all the flow of control transformations, whilst we enclose normal DOS commands [# rem like this #]. batchgen turns the flow of control statements into legal DOS commands, and copies text enclosed in [##] verbatim. (this can lead to problems, see Bugs, below).

If we have the following as input:

```
if (%1==%2 && ! %3==%4 || exist %5)
    [# rem this will be copied verbatim
       @echo it's true #]
```

batchgen's output is:

```
@goto main
:main
if "%1" == "%2" goto AT0
goto ORF1
:AT0
if "%3" == "%4" goto ORF1
goto IF0
:ORF1
if exist %5 goto IF0
goto EI0
:IF0

     rem this will be copied verbatim
     @echo it's true

rem endif
:EI0
:batch_end
```

if we ask for optimization we get:

```
if not "%1" == "%2" goto ORF1
if not "%3" == "%4" goto IF0
:ORF1
if not exist %5 goto EI0
:IF0
     @echo it's true
:EI0
```

If you ask me, that's pretty slick, and it saves a lot of unecessary thinking. batchgen is a run of the mill lex/yacc parser. The lexemes are defined in batchgen.l [../source/batchgen.l], the grammar and it's associated semantic actions are defined in batchgen.y [../source/batchgen.y]. The parser does not parse DOS commands.

Usage

```
batchgen [-o[r|b]] [filename]
```

In the absence of a filename, batchgen will read input from the command line, terminated with ctrl-z if you're using the keyboard.

```
batchgen filename
```

is equivalent to

```
batchgen < filename
```

is equivalent to

```
type filename | batchgen
```

and all three of these command lines will work. batchgen file1 file2 .. is equivalent to batchgen file1. There are no warnings or errors.

Options

The options are as follows:

- -o optimizes but leaves lines beginning with rem and blank lines
- -or optimizes as for -o and also removes rems
- -ob optimizes as for -o and also removes blank lines

- -orb = -obr is full on optimization
- -h prints a usage string

Output

Output is to bg.bat in the current directory. If you've asked for optimization, your output is in bg.bat and the optimized output is in bgo.bat. If optimization fails (see Bugs, below) bgo.bat is written, with code that makes it return immediately wrapped round it, so as not to execute potentially damaging instructions..

Batchgen language

The syntax is pretty much C like.

Version 2 handles functions by generating code that uses the call :label syntax, see this [http://technet.microsoft.com/en-us/library/bb490873.aspx]. There are no changes in the grammar, but the generated code won't work unles you're on XP or later.

break may only be used inside a while or switch statement. return; returns control to the operating system. return primary_expression; (eg return 42;) is rewritten as exit /b primary_expression, see this [http://technet.microsoft.com/en-us/library/bb490902.aspx]. The errorlevel when the batch file terminates is thus set to, for example, 42.

case and default are only legal inside a switch statement, despite the grammar.

Blocks of dos code to be passed through are enclosed by [# and #]. This text is copied verbatim to the output batch file. This should, therefore, be legal DOS commands. batchgen does not parse this text.

Reserved words

if case else fdef switch default while return break || && == != >= "" errorlevel ERRORLEVEL exist EXIST

Grammar

```
batch_file:
statement_list
;

statement:
  dos_command
| compound_statement
| selection_statement
| while_statement
| labelled_statement
| jump_statement
|
;

labelled_statement:
  CASE primary_expression ':' statement
| DEFAULT ':' statement
;

compound_statement:
  '{' statement_list '}'
| '{' '}'
;
```

```
statement_list:
  statement
| statement_list statement
;

selection_statement:
  IF '(' dos_expression ')' statement
| IF '(' dos_expression ')' statement ELSE statement
| SWITCH '(' primary_expression ')' statement
;

while_statement:
WHILE '(' dos_expression ')' statement
;

jump_statement:
  BREAK ';'
| RETURN ';'
| RETURN primary_expression ';'
;

dos_expression:
  logical_AND_expression
| dos_expression LOG_OR logical_AND_expression
;

logical_AND_expression:
  unary_expression
| logical_AND_expression LOG_AND unary_expression
;

unary_expression:
  dos_logical_expression
| '(' dos_expression ')'
| '!' unary_expression
;

dos_logical_expression:
  ERRORLEVEL GE primary_expression
| ERRORLEVEL < primary_expression
| ERRORLEVEL EQ primary_expression
| ERRORLEVEL NE primary_expression
| EXIST primary_expression
| primary_expression EQ primary_expression
| primary_expression NE primary_expression
;

primary_expression:
  identifier
| primary_expression identifier
| en_var
| primary_expression en_var
| param
| primary_expression param
| number
| primary_expression number
| empty
;
```

The non terminals identifier, en_var and param are defined as follows:

- identifier `({letter}|{digit}|\\|\.|\-)([a-zA-Z$\.\\_\*:]|{digit})*` - this should boil down to a dos directory name
- en_var - is an identifier surrounded by %%
- param - is a parameter prefixed by %

Building

There is a generated guide to the source in doc\doxy\index.html.

You need:

- a C++ compiler
- lex and yacc binaries

compile.bat (in bin) will compile from the command line. Using MinGW produces a buggy program, you're better using Visual Studio Express 2010 (free) and the batchgen.sln file (in vce2010).

The VC project doesn't run lex and yacc: bin\doLexYacc.bat calls lex and yacc (I use the GnuWin32 [http://gnuwin32.sourceforge.net/] ones). Running yacc on batchgen.y gives the following error:

```
state contains 1 shift/reduce conflict.
```

This is the if-else ambiguity, we accept yacc's default resolution.

There are some test files in tests, these should all generate and optimize witjout error.

Bugs

- The optimizer cannot do agressive optimization without removing comments and blank lines.
- The contents of input and output files are held in main memory, consequently you could run out of memory.
- Trailing space on environmental variables. This can be a real pain.

  ```
  [# set VAR=99 #]
  ```

  when translated and executed by the OS will give VAR the value "99 ". You will be saying something like

  ```
  if "VAR"=="99" do something
  ```

  and your program won't do something, it will do nothing. This is because VAR has the value "99 ", not "99".
- It's not written in C++. Some of the code is pretty nasty in retrospect (2011) but it more or less works.
- function parameters not supported. This doesn't stop you defining a function that uses parameters, but you will have to call it using something like [# call func arg1 arg2 #] rather than func(arg2, arg2).

Acknowledgments

This program is pretty much lifted from Compilers: Principles, Techniques and Tools by Aho, Sethi and Ullman, pub Addison Wesley 1986.

The grammar is pretty much lifted from C.