
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>].

Table of Contents

Introduction	1
Usage	2
Options	2
Output	3
Batchgen language	3
Reserved words	3
Grammar	3
Building	5
Bugs	5
Acknowledgments	5

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 unnecessary 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.

Functions do not take parameters, nor do they return values. It is an error for a function definition to contain a function call, ie functions cannot be nested, owing to the extremely crude return mechanism. Functions must have been defined before they can be called.

break may only be used inside a while or switch statement. return; may be used in a function, in which case control goes back to the main program, or in the main program, in which case control returns to the operating system. return primary_expression; (eg return 42;) is a bit of a bodge; a program called el is invoked, which returns its argument as an int. Control is then given to the operating system. The errorlevel when the batch file terminates is thus set to, for example, 42. This requires that the program el.com or el.exe be available to your batch file.

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

Grammar

```
batch_file:
statement_list
;

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

function_definition:
FDEF identifier '(' ')' compound_statement
;

function_call:
function_identifier '(' ')'
;

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
    ;

function_identifier:
    func_id
    ;
```

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.