MKWHAT: A Compiler-Development Tool for Fast Keyword Recognition

Peter M. Maurer
Department of Computer Science
Baylor University
Waco, TX 76798

1 Abstract

Although compiler keywords can be recognized directly in LEX (or flex) these tools are rather slow. MKWHAT is a tool that can be used to generate a keyword recognizer that will recognize key words using a minimal number of comparisons. Keywords may be either case sensitive or case insensitive.

2 Description

MKWHAT creates a function that can be called from the lexical analyzer to identify a keyword. An example of the use of this tool in a LEX specification is as follows.

```
[A-Za-z][A-Za-z0-9]*   { return KeyWordDetect(yytext); }
```

The function KeyWordDetect is generated by the MKWHAT tool using a MKWHAT definition file.

The MKWHAT definition file contains configuration commands, keyword specification commands, and comments. Any blank line is considered a comment. There are four configuration commands. The command name must begin at the first character of the line.

#include …
: This line will be copied intact into the output. This is assumed to be the name of an include file. The files will appear at the beginning of the output in the order specified. It is common to include one or more y.tab.h files from yacc output.

#default TOKEN_NAME
: This line defines the token that will be returned if the input does not mach any keyword. This specification is typically something like “VARIABLE_NAME”.

#sub SubroutineName
: MKWHAT generates a single function. This command specifies the name of the function. It will be defined using the following prototype.

```
int SubroutineName(char *x);
```

#class ClassName

The default behavior of MKWHAT is to generate a global function. If you want it to generate a member of a class instead, specify this command with the name of the class of which the function is to be a member.

Any other line beginning with a # character is treated as a comment.

Keyword specification commands have a very rigid format. The keyword must start at the first character of the line. It must be followed by a single tab character. Following the tab character is a numeric specification giving the token value for the keyword. This can be anything recognizable by C++ as an integer constant, or a #define symbol taken from an #include file. A newline character must follow the numeric specification. If the last line in a MKWHAT definition file is a keyword specification command, you must be careful to guarantee that the file ends with a newline character. The sequence \r\n is considered equivalent to \n.

3 Invoking MKWHAT

MKWHAT can be invoked using one of two commands:

mkwhat specification-file output-file

or

mkulwhat specification-file output-file

The command mkwhat provides for case-sensitive keywords while mkulwhat provides for case-insensitive keywords. With the exception of one parameter, the two commands can be used interchangeably. The output-file may have either a .c or a .cpp suffix.

These commands have the following parameters

-x This parameter is used ONLY with mkulwhat. To do the case-conversion, mkulwhat uses a character conversion table. If you have more than one mkulwhat generated function in your project, all but the first must be generated with the –x option to prevent duplicate definition of the case-conversion table.

-d TOKEN_NAME This parameter is equivalent to the #define command. This specifies the default token on the command line instead of in the file itself. This can be used if the same definition file is used with different default tokens at different times.

-s SubroutineName This parameter is equivalent to the #sub command.

-i IncludeFile This parameter is roughly equivalent to the #include command. The file name must be specified without quotes or angle-brackets. It will be enclosed in quotes when the #include statement for it is generated. This parameter can be specified as many times as desired. The –i is required for each file name.

-o ClassName This parameter is equivalent to the #sub command.

-a This parameter indicates that the output file is to be used with a Visual C++ precompiled header. If this parameter is specified, the first line of the generated file will be:

#include “StdAfx.h”

The inclusion of stdio.h (which will be included otherwise) will be suppressed.

Both programs mkwhat and mkulwhat will compile under both Linux (i.e. gcc) and Visual C++. The distribution file contains a Makefile for Linux and Visual C++ project files. (Version 8).