Quadrivio
General Edit

User's Guide
1.0.2 Addenda

Quadrivio Corporation
1563 Solano Avenue #360
Berkeley, California 94707

Contents

Chapter 1
Changed features

This chapter describes changes in the structure syntax that were introduced in General Edit 1.0.2. You may need to edit structures that worked in version 1.0.1 if you want to use them in version 1.0.2 or later. You should read this section to see if the changes affect any of your structure definitions.

Invalid option combination (hc) or ;hc

Previous behavior

When the ‘h’ (skip header) and ‘c’ (collapse contents) options were used together in structs, unions, or arrays, the ‘c’ option was ignored. When the “collapse contents” option is used, only the header row with a closed triangle is initially visible in the data pane. If the “skip header” option were also to be applied, nothing would be visible; for this reason, the “skip header” option was ignored when the “collapse contents” option was selected.

New behavior

The option combination is no longer legal and is treated as a syntax error.

TIFF example file

The (hc) combination was used in the example TIFF structure that shipped with versions 1.0 and 1.0.1. A corrected version is shipped with 1.0.2.

“Inheritance” of option flags in named structs, etc.

Introduction

This section discusses the effect of option flags on structs and unions that are re-used after being defined. For example:

```
struct(h) MyRec {
  Int16 alpha;
  char3 beta;
} firstUse;

struct MyRec secondUse;
```

Previous behavior

In previous versions, the effect of option flags was inconsistent when flags were used in later appearances of the struct or union.

New behavior

In version 1.0.2 and later, flags that are used in the initial definition are automatically included when the struct is used again; you should not repeat them. Flags that are included in later appearances of the struct apply to that appearance only, and they must be consistent with the original set of flags.

Example 1

Inheritance of flags from original definition:

```
struct(h) MyRec {
  Int16 alpha;
  char3 beta;
} firstUse;

struct MyRec secondUse;
```

Both structs are displayed with header line omitted (option h).

Example 2

Applying flags to later appearances of struct or union:

```
struct MyRec {
  Int16 alpha;
  char3 beta;
} firstUse;

struct(c) MyRec secondUse;
```

The second appearance of the struct appears initially closed (option c); the first does not.

Example 3

Inconsistent flags not allowed:

```
struct(h) MyRec {
  Int16 alpha;
  char3 beta;
} firstUse;

struct(c) MyRec secondUse;
```

The use of the (c) option in the second struct causes a syntax error, because (hc) is not a legal combination.

Long long format changed from %Ld to %lld

New behavior

To display a 64-bit integer, use %lld instead of %Ld in the format specifier.

Example:

```
Int64 num;
evalp("value = %lld") num + 1;
```

Floating-point values stored in variables

Old behavior

All variable defined by the var statement were integers.

Example:

```
var x;
eval x = 3.12;
evalp x;
// this displays "3" instead of 3.12
```

New behavior

The type of a variable defined by the var statement is either integer or floating-point, depending on how it is used.

Example:

```
var x;
evalp x = 100;
evalp x;
// this displays "100"

var y;
eval y = 3.12;
evalp y;
// this displays "3.120000"
```

Chapter 2
New features

This chapter describes new features introduced in version 1.0.2. (These are minor enhancements of features that existed in version 1.0.)

Look up default structures by file name extension

Introduction

This feature enhances the use of the Default Structures folder. The Default Structures folder allows General Edit automatically to select the correct structure definition when a file is opened.

In versions before 1.0.2, the selection of a structure definition file was based on the creator and/or file type of the file being opened. In version 1.0.2, the selection can also be based on the file name extension (the characters following a period in the file name). This is useful for files that originated on another operating system. (When non-Macintosh files are transferred to the Mac, the often have creator and file types of '????', and the only way to distinguish their type is by the extension.)

Feature

If the Mac OS file type and creator of a file does not have a matching structure definition file in the Default Structures folder, General Edit will look in the folder for a file whose name is the extension (up to 7 characters) of the file being opened.

Example

When General Edit opens a file named “ABCD.EFGH” and cannot find a structure definition file in the Default Structures folder that matches the file’s Mac OS file type, it then looks for a definition file named “EFGH” in the folder.

typedef struct and typedef union

Feature

The typedef statement can now be used with struct and union types.

Example

```
typedef struct MyRec {
  Int16 alpha;
  char3 beta;
};

struct MyRec aRec;
```

The typedef statement defines the struct without displaying anything or advancing the current pointer in the file.

Note that this syntax differs from that for simple types, for example:

```
typedef Int16 MyShort;
```

cstring, ustring, and char options

Multiple terminators

A cstring can be terminated by any one of several characters.

Example:

```
cstring("^$_");
```

Variable-length strings

The cstring and ustring types can have variable lengths where the length is determined by an expression.

Example:

```
const x = 15;
cstringN({x + 1});
```

ASCII-coded decimals

Character strings that contain a decimal integer in text form can evaluated, and the decimal value can be used in later expressions. The (d) flag causes the characters to be interpreted as a decimal integer.

Example:

```
char5(d) count;
Int16 value[{count}];
```

Array labels

Feature

The label text for the header line of an array can be specified by putting the text in quotes within the array brackets.

Example:

```
Int32 [100; "array label"];
```

Global use of enum constants

Feature

Constants that are defined in an enum statement can be used outside that statement if the (g) global flag is included.

Example:

```
enum(g) {
  alpha = 1,
  beta = 2} x;

if (x == alpha)
  Int32;
```

Use of constants in switch statements

Feature

Switch statement cases can be specified with symbolic constants as well as integers.

Example:

```
const alpha = 1;

switch (x) {
  case alpha: Int8;
  case 2: Int16;
};
```

Recursive structs

Introduction

Before version 1.0.2, General Edit could not handle recursive structs, i.e., structs that contained themselves. (The QuickTime file format is an example of a recursive format.) Version 1.0.2 can parse recursive structs.

Example:

```
struct Tree {
  Int8 leaf;
  if (leaf == 1)
    char;
  else
    struct Tree [2];
};
```

Local variables

If you work with recursive structs, you may need to use variables that have scope within the current recursion of a struct (and are unaffected by deeper recursions of the struct). To define such a variable, use the (L) flag in the var statement.

Example:

```
struct Tree {
  Int8 leaf;
  var(L) branch;
  eval branch = leaf == 0;
  if (leaf == 1)
    char;
  else
    struct Tree [2];
  evalp branch;
};
```

Editing and selection keys

Multiple-click drag	Double-click dragging selects whole words. Triple-click dragging selects whole lines.
Command-arrow keys	Command-left arrow moves the cursor to the beginning of a line. Command-right-arrow moves the cursor to the end of a line. Command-up-arrow moves the cursor to the beginning of a pane or data column. Command-down-arrow moves the cursor to the end of a pane or data column. Holding down the shift key when using a command-arrow key adds to the current selection.

Increased size limits for comments and symbol names

Comments	// comments may now have up to 128 characters.
Symbols	Variable and constant names may now be up to 31 characters in length.