Subject: CParser do not check for invalid strings that span lines
Posted by mingodad on Sat, 12 Apr 2014 10:48:34 GMT
View Forum Message <> Reply to Message

Hello !

Working with witz templates I found error messages a bit misleading when I forgot to put an end quote on one string, following the problem I found several problems (it got better but the error message has a wrong column info on my case it is 2 bytes upfront):

1- CParser do not check for invalid strings that spam multiple lines (see fix bellow):
/////////

```
--- /home/mingo/upp/uppsrc/Core/parser.cpp
+++ /tmp/parser-0.cpp
@@ -327,7 +327,7 @@
    }
   }
   else {
-   if(*term < ' ') {
+   if(*term < ' ' || *term == '\n') {
     if(chkend) {
      ThrowError("Unterminated string");
      return result;
```

////////

2 - Witz parser do not ask to check string end, also when calling "Block()" recursivelly line count get reseted (fix bellow):
////////

```
--- /home/mingo/upp/uppsrc/Skylark/Compile.cpp
+++ /tmp/Compile-0.cpp
@@ -199,7 +199,7 @@
   if(p.Char('0'))
    c.value = int(p.IsNumber() ? p.ReadNumber(8) : 0);
   else
-   c.value = p.IsString() ? Value(p.ReadString()) : Value(p.ReadDouble());
+   c.value = p.IsString() ? Value(p.ReadString(true)) : Value(p.ReadDouble());
  }
  return result;
 }
@@ -356,16 +356,17 @@
  ExeBlock& blk = result.Create<ExeBlock>();
  const char *s = p.GetSpacePtr();
  const char *b = s;
- int line = 1;
```

```
  while(*s) {
   if(*s == '$') {
    if(s[1] == '$') {
     blk.AddText(b, s + 1);
-    p.Set(s + 2, NULL, line);
+    p.Set(s + 2, NULL, current_line);
     b = s = p.GetSpacePtr();
    }
    else {
     blk.AddText(b, s);
-    p.Set(s + 1, NULL, line);
+    p.Set(s + 1, NULL, current_line);
     if(p.Id("if")) {
      ExeCond& c = blk.item.Add().Create<ExeCond>();
      p.PassChar('(');
@@ -403,15 +404,15 @@
     else
      blk.item.AddPick(Prim());
     b = s = p.GetSpacePtr();
-    line = p.GetLine();
+    current_line = p.GetLine();
    }
   }
   else
   if(*s++ == '\n')
-   line++;
+   current_line++;
  }
  blk.AddText(b, s);
- p.Set(s, NULL, line);
+ p.Set(s, NULL, current_line);
  return result;
 }

////////

--- /home/mingo/upp/uppsrc/Skylark/Witz.h
+++ /tmp/Witz-0.h
@@ -194,7 +194,6 @@
  Vector<bool>  forvar;
  bool          optimized;
  int           count_node;
- int     current_line;

  int ForVar(String id, int i);

@@ -231,7 +230,7 @@
```

```
  static void Register(const String& id, Value (*fn)(const Vector<Value>&, const Renderer *));

- Compiler(const char *code, const Index<String>& var) : p(code), var(var, 1) {
forvar.SetCount(var.GetCount(), false); }
+ Compiler(const char *code, const Index<String>& var) : p(code), var(var, 1) { current_line = 0;
forvar.SetCount(var.GetCount(), false); }
 };

 One<Exe> Compile(const char *code, const Index<String>& vars);

//////////
```

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by nlneilson on Sat, 12 Apr 2014 17:18:25 GMT
View Forum Message <> Reply to Message

Do you mean 'span' lines rather than 'spam' ??

I often parse character strings a character at a time.
The 'line' as represented by \n is just another character.
The \0 is the end of the string regardless of how many \n there are.

I don't recall working with 'witz' but the CParser code seemed OK when I have used it.
In theIDE don't you get an indication the "..." string does not have the ending "

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mirek on Sat, 12 Apr 2014 17:45:31 GMT
View Forum Message <> Reply to Message

mingodad wrote on Sat, 12 April 2014 10:48Hello !

Working with witz templates I found error messages a bit misleading when I forgot to put an end quote on one string, following the problem I found several problems (it got better but the error message has a wrong column info on my case it is 2 bytes upfront):

1- CParser do not check for invalid strings that spam multiple lines (see fix bellow):
/////////

```
--- /home/mingo/upp/uppsrc/Core/parser.cpp
+++ /tmp/parser-0.cpp
@@ -327,7 +327,7 @@
    }
   }
   else {
-   if(*term < ' ') {
```

+ if(*term < ' ' || *term == '\n') {

As '\n' is clearly < ' ', this is unlikely to change anything :)

Anyway, the problem was something else: ReadString has bool parameter that allows it to ignore unterminated String errors. It was added in order to be able to deal with some broken files situations, unfortunately it was set to 'ignore problems' as default.

This was kind of stupid decision, so I have been bold enought to do a backward incompatible change, making checking mode the default. I hope it is very unlikely that this would cause any problems...

(will check witz stuff later)

Mirek

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mingodad on Sat, 12 Apr 2014 19:21:27 GMT
View Forum Message <> Reply to Message

You are right here I didn't noticed the "<", but the problem is with not checking the end of string by default.

I've update to trunk and now I got on one of my existing projects an error message about unterminated string, it seems that utf8 characters are not handled correctly, I have this line on my upp project file and clicking ignoring zeroed the upp file !!!!!!!!;
---
description "Archivo y búsqueda del Boletin Oficial de la Provincia de Málaga\377";

I'll continue to check the witz parser, it got better error messages about line numbers, but when we have merged witz files and the error is in one of the merged ones the line number on the error message is for the merged file, ideally it should indicate the line on the individual inserted file.

Anyway thanks for all of you that helped so far !

File Attachments
1) bug-utf8.png, downloaded 406 times

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mirek on Sun, 13 Apr 2014 06:20:44 GMT
View Forum Message <> Reply to Message

mingodad wrote on Sat, 12 April 2014 19:21You are right here I didn't noticed the "<", but the problem is with not checking the end of string by default.

I've update to trunk and now I got on one of my existing projects an error message about unterminated string, it seems that utf8 characters are not handled correctly, I have this line on my upp project file and clicking ignoring zeroed the upp file !!!!!!!!;
---
description "Archivo y búsqueda del Boletin Oficial de la Provincia de Málaga\377";

I'll continue to check the witz parser, it got better error messages about line numbers, but when we have merged witz files and the error is in one of the merged ones the line number on the error message is for the merged file, ideally it should indicate the line on the individual inserted file.

Anyway thanks for all of you that helped so far !

Ops, looks like fixing one bug exposed another. OK, now fixed.

Mirek

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mingodad on Sun, 13 Apr 2014 07:42:48 GMT
View Forum Message <> Reply to Message

The cast to byte solved it, but I'm not sure if return to ignoring the check end of string by default is a good default !

And continuing working with witz why not register handler with a class method ?

This is something I did for FLTK:

```
class Fl_AnyClass {};

/** Default member callback type definition for all fltk widgets (by far the most used) */
typedef void (Fl_AnyClass::*Fl_MCallback)(Fl_Widget*, void*);
/** Default member callback type pointer definition for all fltk widgets */
typedef Fl_MCallback* Fl_MCallback_p; // needed for BORLAND
/** Zero parameter member callback type definition passing only the widget */
typedef void (Fl_AnyClass::*Fl_MCallback0)();
/** One parameter member callback type definition passing only the widget */
typedef void (Fl_AnyClass::*Fl_MCallback1)(Fl_Widget*);
/** Member callback type definition passing the widget and a long data value */
typedef void (Fl_AnyClass::*Fl_MCallback2)(Fl_Widget*, long);
/** Member callback type definition passing the widget and a long data value */
typedef void (Fl_AnyClass::*Fl_MCallback3)(Fl_Widget*, double);
```

```
#define MCALLBACK(wdg, mf) (wdg)->mcallback((Fl_AnyClass*)this, (Fl_MCallback)&mf)
#define THISMBACK(wdg, mf) (wdg)->mcallback((Fl_AnyClass*)this,
(Fl_MCallback)&THISCLASS::mf)


...
class FL_EXPORT Fl_Widget : public Fl_Rectangle {
  friend class Fl_Group;

  Fl_Group* parent_;
  union {
     Fl_Callback* callback_;
     Fl_MCallback mcallback_;
  };
  void* user_data_;
  //int x_,y_,w_,h_;  //Using Fl_Rectangle
  Fl_AnyClass *any_class_mcb_;
...
 /** class that holds a method for callbacks*/
  Fl_AnyClass *any_class_mcb() {return any_class_mcb_;};

  /** Gets the current class method callback function for the widget.
     Each widget has a single class method callback.
     \return current mcallback
   */
  Fl_MCallback mcallback() const {return mcallback_;}

  /** Sets the current class method callback function for the widget.
     Each widget has a single class method callback.
     \param[in] cb new class method callback
     \param[in] p user data
   */
  void mcallback(Fl_AnyClass *klass, Fl_MCallback cb, void* p) {
     any_class_mcb_ = klass;
     mcallback_=cb;
     user_data_=p;
  }

  /** Sets the current class method callback function for the widget.
     Each widget has a single callback.
     \param[in] cb new method callback
   */
  void mcallback(Fl_AnyClass *klass, Fl_MCallback cb) {
     any_class_mcb_ = klass;
     mcallback_=cb;
  }

  /** Sets the current callback method function for the widget.
     Each widget has a single method callback.
```

```
      \param[in] cb new callback
   */
  void mcallback(Fl_AnyClass *klass, Fl_MCallback0 cb) {
        any_class_mcb_ = klass;
        mcallback_=(Fl_MCallback)cb;
  }

  /** Sets the current callback method function for the widget.
      Each widget has a single method callback.
      \param[in] cb new callback
   */
  void mcallback(Fl_AnyClass *klass, Fl_MCallback1 cb) {
      any_class_mcb_ = klass;
      mcallback_=(Fl_MCallback)cb;
  }

  /** Sets the current callback method function for the widget.
      Each widget has a single method callback.
      \param[in] cb new callback
      \param[in] p user data
   */
  void mcallback(Fl_AnyClass *klass, Fl_MCallback2 cb, long p=0) {
      any_class_mcb_ = klass;
      mcallback_=(Fl_MCallback)cb;
      user_data_=(void*)p;
  }
...
/** Calls the widget callback.

  Causes a widget to invoke its callback function with arbitrary arguments.

  \param[in] o call the callback with \p o as the widget argument
  \param[in] arg use \p arg as the user data argument
  \see callback()
*/
void
Fl_Widget::do_callback(Fl_Widget* o,void* arg) {
  Fl_Widget_Tracker wp(this);
  if(any_class_mcb_) (*any_class_mcb_.*mcallback_)(o,arg);
  else callback_(o,arg);
  if (wp.deleted()) return;
  if (callback_ != default_callback)
    clear_changed();
}
```

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mirek on Sun, 13 Apr 2014 17:44:16 GMT
View Forum Message <> Reply to Message

mingodad wrote on Sun, 13 April 2014 07:42The cast to byte solved it, but I'm not sure if return to ignoring the check end of string by default is a good default !

Not sure what you are speaking about here, default stays at "do not ignore"...

Quote:
And continuing working with witz why not register handler with a class method ?

This is something I did for FLTK:

Have to say I am again confused. The code you have posted seems unrelated to witz (which is Skylark templating system).

Mirek

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mingodad on Sun, 13 Apr 2014 17:53:31 GMT
View Forum Message <> Reply to Message

I'm getting a bit crazy, I saw this commit on https://code.google.com/p/upp-mirror/source/diff?spec=svn7237&r=7237&format=side&path=/trunk/uppsrc/Core/Par ser.h and maybe because I usually view diffs in after/before order I misunderstood it.

Sorry about that!

And the code I pasted the is about something I did on FLTK using the ideas from U++ to have class methods callbacks and was suggesting to have the same for registering SKYLARK handlers.

Subject: Re: CParser do not check for invalid strings that span lines
Posted by mirek on Sun, 13 Apr 2014 18:46:17 GMT
View Forum Message <> Reply to Message

I hope that issue with error line numbers in .with should be now fixed.

Subject: Re: CParser do not check for invalid strings that spam lines

Posted by mirek on Sun, 13 Apr 2014 18:49:52 GMT
View Forum Message <> Reply to Message

mingodad wrote on Sun, 13 April 2014 17:53I'm getting a bit crazy, I saw this commit on https://code.google.com/p/upp-mirror/source/diff?spec=svn723 7&r=7237&format=side&path=/trunk/uppsrc/Core/Par ser.h and maybe because I usually view diffs in after/before order I misunderstood it.

Sorry about that!

And the code I pasted the is about something I did on FLTK using the ideas from U++ to have class methods callbacks and was suggesting to have the same for registering SKYLARK handlers.

Well, the problem is that Skylark handlers are not (and cannot be) associated with C++ classes. The issue is that "classical" web development is naturally "stateless", C++ objects generally do not live between requests (heck, the request even can end on completely different machine, right?).

BTW, I just wonder if understand the term "class method" correctly: do you mean 'static' methods too?

Mirek

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mingodad on Sun, 13 Apr 2014 19:19:30 GMT
View Forum Message <> Reply to Message

When I mean class methods I really want to say instance methods, We have database definition on main SkylarkApp so for me seem natural to handle requests within the main app itself so we can use all elements directly instead of casting http.App().

Subject: Re: CParser do not check for invalid strings that span lines
Posted by mingodad on Mon, 14 Apr 2014 10:16:28 GMT
View Forum Message <> Reply to Message

As I mentioned before the latest fixes made the line numbers on error messages a bit better, if the witz template do not include other templates the line number is correct but if it includes other witz templates then the line number is for the whole preprocessed witz template and is meaningless, ideally it should say:

Internal server error
witz_file_name(107,30): Unterminated string <<<< where witz_file_name can be any included witz template that has an error
---

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by mirek on Mon, 14 Apr 2014 11:36:45 GMT
View Forum Message <> Reply to Message

mingodad wrote on Sun, 13 April 2014 19:19When I mean class methods I really want to say instance methods, We have database definition on main SkylarkApp so for me seem natural to handle requests within the main app itself so we can use all elements directly instead of casting http.App().

Well, but as I said, unfortunately there is no instance...

Mirek

Subject: Re: CParser do not check for invalid strings that span lines
Posted by mirek on Mon, 14 Apr 2014 11:37:20 GMT
View Forum Message <> Reply to Message

mingodad wrote on Mon, 14 April 2014 10:16As I mentioned before the latest fixes made the line numbers on error messages a bit better, if the witz template do not include other templates the line number is correct but if it includes other witz templates then the line number is for the whole preprocessed witz template and is meaningless, ideally it should say:

Internal server error
witz_file_name(107,30): Unterminated string <<<< where witz_file_name can be any included witz template that has an error
---

OK, that is something missed so far and something that must be fixed.

Mirek

Subject: Re: CParser do not check for invalid strings that spam lines
Posted by dolik.rce on Mon, 14 Apr 2014 13:15:45 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 14 April 2014 13:36mingodad wrote on Sun, 13 April 2014 19:19When I mean class methods I really want to say instance methods, We have database definition on main SkylarkApp so for me seem natural to handle requests within the main app itself so we can use all elements directly instead of casting http.App().

Well, but as I said, unfortunately there is no instance...

Mirek

I actually like the idea... It is kind of awkward when you have to resort to casting only to get something from the main app class (database definition or other configuration).

What about making the SKYLARK macro define the handler method as a member of AppClass? The only ugly thing is, that it would be programmers responsibility to provide declaration in the AppClass (perhaps via another macro), but I could live with that...

Honza

Subject: Re: CParser do not check for invalid strings that span lines
Posted by mirek on Sat, 26 Apr 2014 10:00:44 GMT
View Forum Message <> Reply to Message

mingodad wrote on Mon, 14 April 2014 12:16As I mentioned before the latest fixes made the line numbers on error messages a bit better, if the witz template do not include other templates the line number is correct but if it includes other witz templates then the line number is for the whole preprocessed witz template and is meaningless, ideally it should say:

Internal server error
witz_file_name(107,30): Unterminated string <<<< where witz_file_name can be any included witz template that has an error
---

Problem fixed (http://www.ultimatepp.org/forums/index.php?t=msg&goto=43 040&#msg_43040).

Mirek
