Subject: Some MAKE_DLL questions
Posted by natha on Mon, 19 Oct 2015 13:30:16 GMT
View Forum Message <> Reply to Message

Hi guys,

I've never created a sharable library from C code to be used by IDL with CALL_EXTERNAL.
I was wondering if it would be possible to create some parallel code using C + OpenMP and then generate de DLL. I don't know if this is possible since the documentation says:
"Not every possible option supported by the C compiler or system linker is addressed, only those commonly needed by IDL-related C code."

I have some more questions related to the DLLs:
- Are they platform independent? Can they be used in Mac, Win, Linux?
- How is the memory managed? Are the variables copied to a different block?

Thank you for all your help and clarifications,
Nata

Subject: Re: Some MAKE_DLL questions
Posted by Russell[1] on Mon, 19 Oct 2015 15:04:24 GMT
View Forum Message <> Reply to Message

I never got make_dll to produce a shared library that used OpenMP. If you make it work, I'd love to know how you did it. I did all this on Mac OS.

-Russell

On Monday, October 19, 2015 at 9:31:26 AM UTC-4, nata wrote:
> Hi guys,
>
> I've never created a sharable library from C code to be used by IDL with CALL_EXTERNAL.
> I was wondering if it would be possible to create some parallel code using C + OpenMP and then generate de DLL. I don't know if this is possible since the documentation says:
> "Not every possible option supported by the C compiler or system linker is addressed, only those commonly needed by IDL-related C code."
>
> I have some more questions related to the DLLs:
> - Are they platform independent? Can they be used in Mac, Win, Linux?
> - How is the memory managed? Are the variables copied to a different block?
>
> Thank you for all your help and clarifications,
> Nata

Subject: Re: Some MAKE_DLL questions
Posted by natha on Mon, 19 Oct 2015 15:44:49 GMT
View Forum Message <> Reply to Message

I guess is not possible. What about my other 2 questions?

Subject: Re: Some MAKE_DLL questions
Posted by Michael Galloy on Mon, 19 Oct 2015 16:58:10 GMT
View Forum Message <> Reply to Message

On 10/19/15 7:30 AM, nata wrote:
> Hi guys,
>
> I've never created a sharable library from C code to be used by IDL with CALL_EXTERNAL.
> I was wondering if it would be possible to create some parallel code using C + OpenMP and then generate de DLL. I don't know if this is possible since the documentation says:
> "Not every possible option supported by the C compiler or system linker is addressed, only those commonly needed by IDL-related C code."

You can pass flags to the EXTRA_CFLAGS and EXTRA_LFLAGS keywords. I've not used OpenMP, are there just special flags to use? You will also have to make sure MAKE_DLL is set to use the correct compiler.

> I have some more questions related to the DLLs:
> - Are they platform independent? Can they be used in Mac, Win, Linux?

Absolutely not. But use the PLATFORM_EXTENSION keyword with MAKE_DLL (or just use the correct string in the name) to make many shared objects, each one valid and named for a particular platform. IDL will use the correct one.

> - How is the memory managed? Are the variables copied to a different block?

I'm not sure what you mean.

> Thank you for all your help and clarifications,
> Nata
>

--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)

Subject: Re: Some MAKE_DLL questions
Posted by natha on Mon, 19 Oct 2015 17:36:22 GMT
View Forum Message <> Reply to Message

Thank you Michael for your reply!

About my third question; I just want to know if the variables are copied (duplicated) when they are passed to the C code.
Thank you,
Nata

Subject: Re: Some MAKE_DLL questions
Posted by Jim Pendleton on Tue, 20 Oct 2015 01:40:56 GMT
View Forum Message <> Reply to Message

On Monday, October 19, 2015 at 11:36:39 AM UTC-6, nata wrote:
> Thank you Michael for your reply!
>
> About my third question; I just want to know if the variables are copied (duplicated) when they are passed to the C code.
> Thank you,
> Nata

By default, scalar arguments are passed by reference. There's a VALUE keyword to CALL_EXTERNAL to control this. Arrays are always passed by reference.

Strings in particular are a special case. You'll want to understand the differences. (It's generally easier to convert them to byte arrays instead and cast the pointers to char * on the C side.)

As the online help says, "See Chapter 3, "Using CALL_EXTERNAL" (External Development Guide in the help/pdf directory of your IDL installation).

Jim P

Subject: Re: Some MAKE_DLL questions
Posted by natha on Tue, 20 Oct 2015 15:13:59 GMT
View Forum Message <> Reply to Message

OK, so it seems to work...
I took the example code of Michael's book and I wrote the following OpenMP version:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

float callex_total(float arr[], int *n) {
```

```
  int nthreads, i, tid;
  float total = 0.;

  #pragma omp parallel shared(arr) private(i,tid)
  {
  tid = omp_get_thread_num();
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }

  printf("Thread %d starting...\n",tid);

  #pragma omp for
  for (i = 0; i < *n; i++)
    {
    total += arr[i];
    printf("Thread %d: total += arr[%d]\n",tid,i);
    }

  }  /* end of parallel section */

  return(total);
}
```

Then, I had to play a lot with the compiler but at the end I've found the way:

```
  cfile='callex_total.c'
  cfile_noext=file_basename(cfile, '.c')

  srcdir = file_dirname(file_expand_path(cfile))

  cc='gcc -fopenmp -lgomp -fPIC %c -c -o %o'
  ld='gcc -fopenmp -shared -o %L %O %X'

  make_dll, cfile_noext, 'IDL_Load', input_directory=srcdir, output_directory=srcdir, /verbose,
/show_all_output, cc=cc, ld=ld ;;extra_cflag='-fopenmp -c'

  result = call_external(cfile_noext+'.so', cfile_noext, findgen(10), 10, /f_value, /auto_glue)
  print, result
```

And here is the output:

```
Thread 6 starting...
Thread 4 starting...
Thread 4: total += arr[8]
Thread 4: total += arr[9]
```

```
Thread 2 starting...
Thread 2: total += arr[4]
Thread 2: total += arr[5]
Thread 7 starting...
Thread 5 starting...
Thread 1 starting...
Thread 1: total += arr[2]
Thread 1: total += arr[3]
Thread 3 starting...
Thread 3: total += arr[6]
Thread 3: total += arr[7]
Number of threads = 8
Thread 0 starting...
Thread 0: total += arr[0]
Thread 0: total += arr[1]
      45.0000
```

I am very glad to know that we can link OpenMP to IDL.
