From markwa Mon May 15 09:01:25 1989
To: davidw mikedr
Cc: bobgu philba
Subject: Re: alternate startup code
Date: Fri Nov 08 10:05:35 PDT 1991

Does this mean that that the following functions are to be documented in the Reference manual?

| | - the APIs INITTASK, INITAPP, and WAITEVENT

| >From davidw Fri May 12 16:56:01 1989
| To: bobgu markwa mikedr philba
| Subject: alternate startup code
| Date: Fri May 12 16:55:43 1989
|
| No big complaints here, just put in the usual caveats,
| and don't explain in detail what any of those things do.
| This OK by you bob?

| | >From mikedr Fri May 12 12:19:23 1989
| |
| | We would like to add to the SDK a bit of sample code based on CraigC's
| | reduced startup code for the spooler, to allow people to get the
| | memory savings they can if they don't need any of the C runtime startup
| | features. Including this as a sample (as an ASM file and an OBJ file)
| | is the simplest way to do this. However, this will effectively document
| | a few things that have previously been hidden:
| |
| | - the APIs INITTASK, INITAPP, and WAITEVENT
| | - the register setup for entry into a Windows program
| |
| | I doubt that any of this is particularly sensitive, but I thought I'd
| | ask if anyone objects to releasing this. Let me know if you do.

From bobgu Mon May 15 09:33:27 1989
To: davidw markwa mikedr
Subject: Re: alternate startup code
Date: Fri Nov 08 10:05:36 PDT 1991

Fine with me.

| >From davidw Fri May 12 16:55:49 1989
| To: bobgu markwa mikedr philba
| Subject: alternate startup code
| Date: Fri May 12 16:55:43 1989
|
| No big complaints here, just put in the usual caveats,
| and don't explain in detail what any of those things do.
| This OK by you bob?

| | >From mikedr Fri May 12 12:19:23 1989

CONFIDENTIAL

X130963

> We would like to add to the SDK a bit of sample code based on CraigC's reduced startup code for the spooler, to allow people to get the memory savings they can if they don't need any of the C runtime startup features. Including this as a sample (as an ASM file and an OBJ file) is the simplest way to do this. However, this will effectively document a few things that have previously been hidden:
>
> - the APIs INITTASK, INITAPP, and WAITEVENT
> - the register setup for entry into a Windows program
>
> I doubt that any of this is particularly sensitive, but I thought I'd ask if anyone objects to releasing this. Let me know if you do.

From davidw Mon May 15 10:24:44 1989
To: markwa mikedr
Cc: bobgu philba
Subject: alternate startup code
Date: Fri Nov 08 10:05:40 PDT 1991

It means they are NOT to be documented in the reference manual. The usual caveats mean that these are version specific entry points and are guaranteed to change. No documentation is given for this reason. They are to be viewed as black boxes.

> >From markwa Mon May 15 10:09:34 1989
>
> Does this mean that that the following functions are to be documented in the Reference manual?
>
> > - the APIs INITTASK, INITAPP, and WAITEVENT
>
> >From davidw Fri May 12 16:56:01 1989
> To: bobgu markwa mikedr philba
> Subject: alternate startup code
> Date: Fri May 12 16:55:43 1989
>
> No big complaints here, just put in the usual caveats, and don't explain in detail what any of those things do. This OK by you bob?
>
> X130964
>
> CONFIDENTIAL
>
> > >From mikedr Fri May 12 12:19:23 1989
> >
> > We would like to add to the SDK a bit of sample code based on CraigC's reduced startup code for the spooler, to allow people to get the memory savings they can if they don't need any of the C runtime startup features. Including this as a sample (as an ASM file and an OBJ file) is the simplest way to do this. However, this will effectively document a few things that have previously been hidden:
> >
> > - the APIs INITTASK, INITAPP, and WAITEVENT
> > - the register setup for entry into a Windows program

> > > I doubt that any of this is particularly sensitive, but I thought I'd
> > > ask if anyone objects to releasing this. Let me know if you do.

From mikedr Mon May 15 10:26:44 1989
To: markwa
Subject: Re: alternate startup code
Date: Fri Nov 08 10:05:44 PDT 1991

No, they should remain UNdocumented. In fact, we can circumvent the issue by releasing the alternate startup only as an OBJ.

> >From markwa Mon May 15 10:08:36 1989
> To: davidw mikedr
> Cc: bobgu philba
> Subject: Re: alternate startup code
> Date: Mon May 15 09:01:24 1989
>
> Does this mean that that the following functions are to be documented
> in the Reference manual?
>
> > - the APIs INITTASK, INITAPP, and WAITEVENT

From markwa Mon May 15 11:10:47 1989
To: mikedr
Cc: bobgu davidw philba
Subject: Re: alternate startup code
Date: Fri Nov 08 10:05:45 PDT 1991

Releasing the alternative startup only as an OBJ (no source) seems like a very good solution to avoid disclosing the APIs mentioned below. Let's do that.

> >From mikedr Mon May 15 10:26:44 1989
> To: markwa
> Subject: Re: alternate startup code
> Date: Mon May 15 10:21:16 1989
>
> No, they should remain UNdocumented. In fact, we can circumvent the
> issue by releasing the alternate startup only as an OBJ.
>
> > >From markwa Mon May 15 10:08:36 1989
> > To: davidw mikedr
> > Cc: bobgu philba
> > Subject: Re: alternate startup code
> > Date: Mon May 15 09:01:24 1989
> >
> > Does this mean that that the following functions are to be documented
> > in the Reference manual?
> >
> > > - the APIs INITTASK, INITAPP, and WAITEVENT

From markwa Fri Jun 23 10:07:00 1989
To: andyp bobgu
Subject: In C6.0, NULL:= (void*)0
Date: Fri Nov 08 10:05:49 PDT 1991

What is going to happen when C6.0 finds two different #define's for NULL? Do we need to do an #ifdef in Windows.h?

```
#ifdef c6
#define NULL ((void*)0)
#else
#define NULL 0
#endif
```

Also, are we ready to test out this NULL stuff by recompiling USER, as you suggested earlier?

>From markwa Tue May 30 18:13:03 1989
To: andyp bobgu johnen philba
Cc: jodys
Subject: Re: windows.h (and NULL)
Date: Tue May 30 18:13:02 1989

If Languages has possibly made a decision regarding NULL pointers that needs to be revised because of adverse effect on Windows, then we need to analyze the problem and search for a solution before it is too late for Languages to implement the required change in C6.0.

Can Languages wait until we do a test recompilation of USER in the middle of June to determine whether the C6.0 NULL strategy needs to be revised in any way?

Is this testing job one for Windows Development, Windows Testing, or Languages Development or Testing?

X1309E6

CONFIDENTIAL

>From bobgu Tue May 30 17:52:43 1989
To: andyp markwa
Cc: davidw greglo markro
Subject: Re: windows.h
Date: Tue May 30 17:50:46 1989

My comment to davidw was made after reading floyd's mail. After reading his mail I got the impression that a test like "lpstr == NULL" would not work since NULL was interpreted as ds:0 not 0:0. If that is not the case, and from reading your mail, I don't think we have a problem. However, the burden of proof is on you guys, not on us. Running USER through C 6.0 should be a reasonable test. The problem is that it is not easy to set up your environment to build USER (or any part of the core for that matter). Mark, if you want to do it on your system, fine. We (the USER group) do not have the time to do this until after code complete (middle of June).

- BobGu

>From andyp Tue May 30 17:36:30 1989
To: bobgu markwa
Subject: windows.h
Cc: davidw greglo markro
Date: Tue May 30 17:33:44 1989

bobgu: davidw mentioned that you felt WIN might be broken if NULL gets redefined. can you please give me some details.

markwa: sounds like you're involved in this too.

thanks,
//andy

to help you make some informed comments, here are some facts about C 6.0:
1. NULL will be defined as (void *) 0.
2. doing "char far * p = NULL" will give you 0:0.
   c5 sometimes gave you 0:0, sometimes gave you ds:0.
   if it gave you ds:0 when you were passing in a WinProc addr, you would blow up (or non-WinProcs, for that matter).
3. doing "int n = NULL" will give you a warning and 0.
   c5 did the same thing, silently.
4. doing "long n = NULL" will give you a warning and 0:0.
   c5 did the same thing, silently.
5. doing "*p = NULL" will give you a warning, and the correct result.
   c5 did the same thing, silently.
6. doing "case NULL:" will give you an error. you must change your source.
   c5 let you do it, silently.
7. doing "NULL & value" will give you an error. you must change your source.
   c5 let you do it, silently.
8. if one is willing to give up certain big-win optimizations (or get incorrect code), one can go back to the old definition of NULL.
9. we will modify RC/RCPP to handle the new NULL.

could you give me an indication of what this might break in WIN (both the WIN kernel and in various APPs).

also, can we arrange to have you or somebody in your group run your sources through our compiler to get some empirical data on what breaks at either compile or run-time.

CONFIDENTIAL

From markwa Fri Jul 07 10:19:59 1989
To: betsyt
Cc: gunterz mikedr
Subject: C compiler chapter
Date: Fri Nov 08 10:05:55 PDT 1991

X130967

The C compiler chapter is missing some categories of information