

Re: newbie: I/O with nasm

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2006-03/msg00260.html>

- *From:* Frank Kotler <fbkotler@xxxxxxxxxxx>
 - *Date:* Wed, 08 Mar 2006 12:41:08 -0500
-

Dirk Wolfgang Glomp wrote:

TK schrieb:

Hi,

how can I do simple I/O with nasm? For example: I want to read a string from the commandline. Please keep it simple;-).

As an example, say you run a program called program and give it three arguments:

```
./program foo bar 42
```

Ah! Someone who actually read the question! He *does* say "command line". Well, he'll want to learn to take input from either the command line or the keyboard... or wherever stdin is redirected to, if it is. Might make an interesting example to check argc, and if it's "wrong", ask for input instead of printing a "usage" message and exiting...

The stack will then look as follows:

```
4  
program  
foo  
bar  
42
```

Number of arguments (argc), including the program name

Name of the program (argv[0])

Argument 1, the first real argument (argv[1])

Argument 2 (argv[2])

Argument 3 (argv[3]) (Note: this is the string "42", not the number 42)

Right. It should be noted that this applies only to a Linux program that starts with "_start". A program that starts with "main" and is linked with gcc (like some examples in this thread) has the "_start" label in the C

Re: newbie: I/O with nasm

startup code (unless we use "--nostartfiles"), and the stack is slightly different when we get control at "main". Dos is *very* different, as you point out. Windows? Gawd knows – and "GetCommandLine" talks to Him and tells us :)

Working on up the (Linux) stack, the list of command line argument pointers is terminated with a zero. Then comes a similar zero-terminated array of pointers to the environment variables (zero-terminated strings). Yet another way we could get "input" from the user!

Now lets write the program program that takes the three arguments:

```
section .text
global _start

_start:
pop eax ; Get the number of arguments
pop ebx ; Get the program name
pop ebx ; Get the first actual argument ("foo")
pop ecx ; "bar"
pop edx ; "42"

mov eax,1
mov ebx,0
int 80h ; Exit
```

After all that popping, EAX contains the number of arguments, EBX points to wherever "foo" is stored in memory, ECX points to "bar" and EDX to "42". This is obviously way more elegant and simple than in DOS. It took us just 5 lines to get the arguments and even how many there are, while in DOS it takes 14 rather complicated lines just to get one argument!

Well... I suspect 14 lines can be beaten... don't really recall. "Argv[0]", the program name, is a real PITA – hidden at the end of our environment variables, which are, in turn, off in some other segment which we have to find in the PSP... Definitely easier in Linux! Fortunately, we don't need the program name too often. If we *do* need it, dos gives us the full path to the program, while Linux gives only what was typed on the command line. If we want full path, we need a "sys_getcwd", and parse it from there – if the user typed "../..../myprog", we've got a chore to figure out where we really are! Score one for dos, there, I think!

Note that the 3rd pop overwrites the value we put in EBX with the 2nd pop (which was the program name). Unless you have a really good reason, you can usually chuck away the program name as we did here.

I've got a slightly different version that leaves the stack as is, instead of popping everything. I figured I might have forgotten something, and might want to go back later. :) If we just save the initial value of esp at the "_start:" label, we can figure everything out from any point in our program... Provided we haven't trashed that upper part of the stack. In dos, certain operations will trash the command line, so we want to get it quick. Score one for Linux. :)

Re: newbie: I/O with nasm

If what TK wants is Linux examples – or dos – we're in pretty good shape (despite the fact that "assembly in Linux" is sort of an "exotic" subject, and dos is "dead" :) Windows... I'm sure someone can help him.

Did you ever get that VBLANK thing for the framebuffer working? I gave up on it – don't think my card supports it. I did figure out some rudimentary access to the framebuffer device, thanks to your inspiration! I should get back to that, instead of fooling around with "Hello uhClem"! :)

Best,
Frank

.