

Re: A snapshot of the LuxAsm developments

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2004-12/1178.html>

From: Beth (*BethStone21_at_hotmail.NOSPICEDHAM.com*)

Date: 12/20/04

Date: Mon, 20 Dec 2004 14:40:15 GMT

wolfgang kern wrote:

- > *The Half skrev:*
- > *| Please explain. What oportunity is missed ????*
- >
- > *I really don't know much about Linux, but what I saw so far,*
- > *it doesn't allow (beside other safety caused detours):*
- > *The unrestricted, undelayed direct I/O access, and*
- > *I mean it should always have IOPL3 for all rings.*
- > *After comparing a few drivers, it looks like some 'new' Linux*
- > *drivers are just translated from the windoze stuff.*

Linux doesn't allow it by "default" because it's a "multi-user" system...you know, for "security", it won't allow any old user to do whatever they like...just in case it's some malicious user trying to install viruses or something like that...especially remembering that UNIX was originally an OS for `_mainframes_` so such things did have to be controlled, as the mainframe could be being accessed for many rooms in some big business or university building, for example, by hundreds of users simultaneously...

Also, we can't allow any one user to crash the mainframe because that would screw up everyone else's work on the same mainframe...this is the real reason for "protections" (Intel did, in effect, "move down" their mainframe designs into the x86 chip to add on "protections" when it was clear the PC was going to be used for multi-user, multi-tasking things more and more...as, indeed, the `_replacement_` to mainframes)...

But if you log in as "superuser" ("root" :) then, as this user is the "owner" of the entire system (in the OS's eyes, anyway), this user account is allowed to do things that the other users can't...that is, "root" has "permissions" to do anything with "security" (which is why you shouldn't be logged on as "root" unless you really need to be...better to log on as an ordinary user and temporarily get "superuser" permissions with the "su" command, which allows you to temporarily switch to "superuser", do whatever you need to do and then switch back out of "superuser" to an ordinary user :)...)

alt.lang.asm: Re: A snapshot of the LuxAsm developments

As "superuser" (with "root privileges"), there are a few syscalls available to you for this kind of thing...for example:

NAME

ioperm – set port input/output permissions

...

DESCRIPTION

Ioperm sets the port access permission bits for the process for 'num' bytes starting from port address 'from' to the value 'turn_on'. The use of ioperm requires root privileges.

Only the first 0x3FF I/O ports can be specified in this manner. For more ports, the 'iopl' function must be used.

...

So, you CAN use syscalls like this to ask it to "drop" protections on certain port ranges...then "in" and "out" should work happily with those port ranges (it's obviously using the "I/O permissions bitmap" thingy to be so "selective" in doing this...but then that's why Intel added that device to the CPU to allow these kinds of things :)...

As you may guess from what it's doing, this is "Linux / x86 specific" stuff...so, it IS there for use, wolfgang...if you don't hear much mention of it, then that's just because so many people out there want "portability" that they would avoid this syscall completely...but it does exist to allow the kinds of access you're talking about...

The difference is just that Linux "by default" operates with the "protections"...and only "root" is given the security permissions to ask for those "protections" to be dropped...but if this is your own home installation of Linux, then you ARE going to be "root", aren't you?

But what you could do is create your program which does direct I/O port access and it calls "ioperm" (or there are some other I/O port related syscalls, as the above suggests, too :)...and then grant it "root privileges"...that's just a "security" rather than "protections" thing...you just need to be logged on as a user who's "trusted" to make these kinds of changes...then you can ask it to "drop" the permissions on the I/O ports you want to access...you should then be okay to do your "direct access" to the I/O ports...

The "iopl" syscall referred to above, by the way, actually changes the IOPL of the process...so, this one gets you unrestricted access 100% (indeed, the "man" page for the call does warn that you gain enough privilege to

alt.lang.asm: Re: A snapshot of the LuxAsm developments

switch off interrupts too...which it notes is "not recommended" – but it doesn't say "can't be done" – because switching off all interrupts would, of course, mean Linux itself would probably crash :)...

There are also some syscalls called "io_setup", "io_cancel" and such...I don't have "man" pages for these as they are new syscalls in a newer kernel version than I think I have installed here...but the fact they start with "io" could be a good sign that they are more syscalls for direct access stuff too (maybe not...I'm just guessing from the name ;)...

Although, "iop1" allows you to ask for full I/O permissions, so that should be more than enough for "in" and "out" direct port access there (indeed, it's kind of too much when you can switch off interrupts and probably screw up Linux's scheduler doing so...mind you, hmmm, let's see: Of course, you can call "sched_setscheduler" to one of its more drastic "real-time" settings which effectively switches off the scheduler...then you should be okay to access interrupts too...if you're "careful" about it all, of course...although, if you care not to be able to return to "ordinary operations" afterwards, I suppose you don't even have to be "careful" either...screw it all up, if you plan not to switch the scheduler back on at any point...Linux is also used in embedded / real-time systems too, you see, so it does have to allow this kind of thing...there is some bare bones "embedded Linux" package out there which ignores all the desktop stuff for the more "bare bones" systems you like, wolfgang :)...

It just operates "protections" by default for "security" reasons, is all...so that some naughty user can't walk up and screw up your machine when you're not looking...but, with the right permissions ("root"), you can tell it to switch off protections on I/O ports, set the scheduler to "real-time" (which is effectively like switching off pre-emption completely) and so on and so forth...

If those "new" Linux drivers are just translated from Windows, then that's more to do with Microsoft's "unfair advantage" and the growing trend not to give out "specifications" for anything anymore...or just "lazy" coding, of course...

To be honest, looking at Linux, the system itself is reasonably good...the problems it has are more to do with the programmers it attracts...like, for instance, I asked on a Linux newsgroup about the "int 80h" interface (just in case someone has already made some kind of "table" or something with the latest, full information about it...because if they have, that saves us a lot of effort making one of our own for LuxAsm :)...and, yeah, a C / C++ coder appears and says "NEVER use the syscalls directly...always use libc!"...I'm surprised I also wasn't told off for using assembly language too ("ALWAYS use C!!" ;)...

No, it wasn't that bad, really...like I say, not criticised for using assembly language so better than the "usual" responses...but you know the deal: They use C, they use "libraries", they consider ASM, direct access and other things "verboten!" in all circumstances...they don't care for

alt.lang.asm: Re: A snapshot of the LuxAsm developments

"bloat" or "bad performance" because they believe "buy more RAM!"...and so on and so forth...such coders can screw up any system, really...

Like, for instance, the thing that's always struck me as weird is "XFree86" running on an iMac...ummm, why the "86" in the name, if, in fact, it can run on systems other than 86 systems? I'm thinking lots of "portability" and such is getting in the way, which is why the X systems do run rather crap and slow...the design of X itself doesn't mandate this, really (it is "portable" but it does this by having a "standard" for the messages it sends...you know, like the higher internet protocols...the data packets have "standard" information and that's how they are "portable"...and with a GUI, you're usually sending things that are "device-independent", anyway – screen co-ordinates (just integers), "events" (just standard ID codes to say things have happened), etc. – that this should be a problem, except for "pixmap" (X distinguishes, in its terminology, "bitmaps" – which are literal bit maps (i.e. "monochrome") – from "pixmap" which can have many bits per pixel)...ah, it's an old design (still has "bitplanes"...mostly redundant these days on all systems because graphics have gone "beyond" that kind of device) but it's a reasonably good one...could do with an "update" but the basic design is very solid and good, I'd say)...so, if it's slow and crap, that's more to do with "implementation" than design...indeed, while XFree86 runs horribly on my machine here at times, I never had any such problems with X using it on HP workstations nor Sparcs and such before...

Of course, if you want to do "direct" things then DOS-like stuff (or "roll your own" OSes :) would be better, even if only from the perspective that you need to do anything before you can do "direct access"...but Linux isn't quite as bad as Windows...please wash your mouth out with soap and water for saying so! ;)

Beth :)