

# Re: compile+link Fujitsu Linux

---

*Source:* <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2008-02/msg00369.html>

---

- *From:* Richard <[riplin@xxxxxxxxxxxxx](mailto:riplin@xxxxxxxxxxxxx)>
  - *Date:* Wed, 6 Feb 2008 20:13:38 -0800 (PST)
- 

Robert wrote:

On Wed, 6 Feb 2008 00:19:14 -0800 (PST), Richard <[riplin@xxxxxxxxxxxxx](mailto:riplin@xxxxxxxxxxxxx)> wrote:

On Feb 6, 7:01 pm, Robert <[n...@xxxxxx](mailto:n...@xxxxxx)> wrote:

Well, no it doesn't do that. Not even close. But you still haven't explained how your mechanism is 'better'. Given equal slackness in permissions and such and users can access compilers then how is a library any different ? Tools can push and pull stuff out of libraries.

My security is better because it does not depend on file system permissions. It doesn't depend on application code, either.

You claimed that your mechanism of having all the programs in one library was better. That is was more reliable, more secure, easier to deploy and faster.

It may be insignificantly faster in the CALL, but is slower on startup. Deployment would only be 'easier' if you don't bother to test. Reliability is worse because a 'missing file' stops the whole application. Now you simply change the subject to something that is completely independent of whether it uses libraries or individual programs.

The goal of security is to prevent unauthorized people from seeing or changing certain data. I do that by protecting the DATA; you do it by preventing them from running programs.

## Re: compile+link Fujitsu Linux

As an analogy, suppose we are parents who want to prevent our kids from watching sex movies on TV. You would do it by locking up the remote control, which is easily bypassed with a universal remote. I would do it by telling the cable box to play the movies only for designated users.

You claimed that your way was superior and the best that you have come up with is 'less ugly'.

Simpler deployment, can be checked by auditing tools, faster loading, more reliable because mid-run aborts due to missing program are impossible.

These are mere assertions. Having a script to do ZIP up everything is just as simple as having a list for ld. Actually the ZIP list can be \*.so, so that is simpler and more reliable because it collects everything and not just what is in the list of -ls. On unzipping it won't remove a file that failed to be included in zip, while a library short of one program will make that unavailable.

'Simple' is 'what you are used to'. It may be simpler for you because you already have it all set up. My way is simpler for me because I have mine set up. You are attempting to claim some sort of 'global universal truth' for your assertions (as usual).

As I say, if you don't test what you are sending out then it will be less reliable.

I use auditing tools, too. Programs COPY in a 'version.ws' that is recreated each run of make so that I can check that a deployment to a site has resulted in the correct versions of all programs, and this would pick up if anything is missing.

Your assertion that yours is 'more reliable' is complete nonsense. Mine is 100% and you have previously stated that your stuff goes missing.

Your claim about faster loading is also nonsense. Your startup will take longer because you also load the library(s). I load on demand and only load the one that are actually used. The startup time difference may be noticeable, the load of an individual program is not.

What makes  
you think  
that users  
have write  
access to  
the program

Re: compile+link Fujitsu Linux

directory ?  
\_I\_ can  
drop in  
another  
program.

If they have read access,  
they can copy your whole  
directory structure to their  
home  
directory, change  
permissions all they want,  
and run their local copy to  
find out the  
boss' salary.

Well, actually, no they couldn't. I already  
told you that particular  
programs can be set to be owned by the  
application administrator with  
a group id of the trusted group and  
permissions of 640. It is only if  
you are a member of the trusted group that  
give read access to run (or  
copy) that program. Users can be members  
of several groups so access  
can be very granular. Users not a member of  
that group will get a  
'program not found' if they try to select it  
from the menu.

I assumed the files had world read permission. You stopped  
that approach.

So, you would concede then that I have described a mechanism that  
would not be able to be used by your system the way that you have it  
and that this mechanism does protect the programs.

My system? I'd like to take credit for inventing Unix and ELF, but I believe Thompson and  
Ritchie did it first.

I was not referring to your \_Operating\_System\_ but to your system of  
assembling programs.

Re: compile+link Fujitsu Linux

Re: compile+link Fujitsu Linux

Security belongs in the called program, not the menu program. A user could write his own program that calls yours.

Which is not an advantage for your library because he could call that too.

So? The called program issues an Unauthorized message and exits.

And why do you think that "A user could write his own program that calls yours." and have it work ?

Instead of addressing your claimed superiority you merely ramble on about something else entirely unrelated, and straw-men, in the hope that it may bamboozle. Perhaps that is your normal 'arguing' skills, you win if they give up so it doesn't matter what the content is.

It IS relevant to you, since your security depends on the Unix file system. If I can change a group member's (hidden) profile or shell login, I can get his group permission, even though I don't belong to the group. His script could copy the main menu (or ksh) to my home, give it world execute and setgroupid. Next time I run it, I'm in the privileged group.

Well they can't.

I don't have 30,000 developers trying to take me down. The level of security is entirely appropriate for the types of business and clients that I look after.

.

Re: compile+link Fujitsu Linux