

Forum:[Hack Forums](#)

Thread: [C++ static linking, packaging DLLs, etc](#)

Hackers - October 23, 2023, 8:36 pm

Hey all, recently started teaching myself C++ and I've made a few things so far. However, I need to learn more about self contained libraries, static linking, and basically just general release packaging to distribute my programs. No, this is not for hacking or spreading - I just don't want my coworkers to have to go find drivers or other files manually.

I made a collection of tools for Windows in 1 program which has a starter program (to trigger admin perms request on a target exe), and a custom packager that runs my compiler and build commands. Then I decided to setup CMake which also works.

I did one more command line program that has 50 questions for a specific psychology test. For this one I just converted some Python code I had written to be a C++ CLI script instead, also in an exe.

I thought I had it covered with the compiling but discovered the issue of not packing DLLs properly when my other computer couldn't run.

Now I'm basically weighing my options for packaging the DLL alongside the Exe in a zip versus packaging it in the exe itself. I think I'd rather do the latter, but to be honest I'm not opposed to zipping if there's a reason to do that instead. The other idea I had was to host the dependency DLLs and create a Downloader, / install.exe but that would require its own packed DLLs. My overall programs are quite small so a self contained exe won't get too large and if it did I could create an Installer if I had to later.

All in all, I'm taking things slow. I really only had Fri, Sat, Sunday this past weekend to focus on this so I am hoping to get some recommendations here before I kickoff another learning session.

I'm challenging myself to basically write C++ every day so if anyone has any ideas for what's best to practice, let me know. Currently I'm working on integrating OpenSSL with a single file header (that someone else made - [http://lib.h](#)) for HTTPS requests, but struggling hard on this. Learning how to pack DLLs will assist with this later as I'll have to figure out how to dynamically link to a user's SSL library, or pack it in myself somehow.

The ones I'm packaging now are far more common - mostly std:: funcs for filesystem, cout, etc with command line prompt getline / cin, and some recursive file/folder funcs - few env variables being read, but overall nothing too crazy.

If anyone has more info or recommendations on this that would be great (personal experience please, I can read Stack, Gpt, Google for documentation and general information)

Tentco - October 23, 2023, 9:14 pm

From what I understood you are able to link CRT DLLs but not the external libs you're using and you don't want to dynamically include them.

Why don't you write an MSIL based Dropper/Installer that would just drop your resources at a specific location?

It's strange though using /MT flag while compiling should already include all external libs into a single binary, Have you tried it?

stdio - October 23, 2023, 11:16 pm

(Oct 23, 2023 08:36 PM)nodes Wrote: [\[CITE\]](#) Hey all, recently started teaching myself C++ and I've made a few things so far. However, I need to learn more about self contained libraries, static linking, and basically just general release packaging to distribute my programs. No, this is not for hacking or spreading - I just don't want my coworkers to have to go find drivers or other files manually.

I made a collection of tools for Windows in 1 program which has a starter program (to trigger admin perms request on a target exe), and a custom packager that runs my compiler and build commands. Then I decided to setup CMake which also works.

I did one more command line program that has 50 questions for a specific psychology test. For this one I just converted some Python code I had written to be a C++ CLI script instead, also in an exe.

I thought I had it covered with the compiling but discovered the issue of not packing DLLs properly when my other computer couldn't run.

Now I'm basically weighing my options for packaging the DLL alongside the Exe in a zip versus packaging it in the exe itself. I think I'd rather do the latter, but to be honest I'm not opposed to zipping if there's a reason to do that instead. The other idea I had was to host the dependency DLLs and create a Downloader, / install.exe but that would require its own packed DLLs. My overall programs are quite small so a self contained exe won't get too large and if it did I could create an Installer if I had to later.

All in all, I'm taking things slow. I really only had Fri, Sat, Sunday this past weekend to focus on this so I am hoping to get some recommendations here before I kickoff another learning session.

I'm challenging myself to basically write C++ every day so if anyone has any ideas for what's best to practice, let me know. Currently I'm working on integrating OpenSSL with a single file header (that someone else made - httpLib.h) for HTTPS requests, but struggling hard on this. Learning how to pack DLLs will assist with this later as I'll have to figure out how to dynamically link to a user's SSL library, or pack it in myself somehow.

The ones I'm packaging now are far more common - mostly std:: funcs for filesystem, cout, etc with command line prompt getline / cin, and some recursive file/folder funcs - few env variables being read, but overall nothing too crazy.

If anyone has more info or recommendations on this that would be great (personal experience please, I can read Stack, Gpt, Google for documentation and general information)

Good to see someone going deep into C++, a good challenge is to research on alternate entry points ;) yes I mean entry point besides main()

Hackers - October 24, 2023, 3:12 am

(Oct 23, 2023 09:14 PM)Tentco Wrote: [\[CITE\]](#) From what I understood you are able to link CRT DLLs but not the external libs you're using and you don't want to dynamically include them.

Why don't you write an MSIL based Dropper/Installer that would just drop your resources at a specific location?

It's strange though using /MT flag while compiling should already include all external libs into a single binary, Have you tried it?

<https://learn.microsoft.com/en-us/cpp/build/reference/md-mt-ld-use-run-time-library?view=msvc-170>
libgcc_s_seh-1.dll was the missing file on my test exe I sent a coworker. It was just a standard program using std::cout, cerr, cin, getline, with literally nothing really complex inside. I was surprised

something so minimal would require packaging, but maybe I've compiled wrong with g++ or something. There's so many options, I'm sure I am just missing a step or configuring it wrong, like not using a flag or cmake for something.

I am packing my own internal library just fine, I have multiple lib/folderHere/main.cpp and main.h files, it's only these external ones not packing.

I will look up what MT flag does, thanks!

(Oct 23, 2023 11:16 PM)stdio Wrote: [\[CITE\]](#) Good to see someone going deep into C++, a good challenge is to research on alternate entry points ;) yes I mean entry point besides main()
Sounds good, and yes I'm only using int main currently with some custom functions in a lib folder. I'm not sure of other options here but I'll do some Googling on this soon.

My goal is to learn as much as possible with standard programs, then I'm going to take a stab at my own Node module for node.js import, and then I plan to work on a Raspberry Pi. Then from there I'll probably work with Qt (used for other langs in past) to make some GUIs. After that I'm not sure, but my goal is to be as fluent in C++ as I am with Python, PHP, and JS. Then after I imagine I'll probably go lower with standard C or go to C# for some NET experience.

Thanks for encouragement, I love programming and I try to learn a new language every couple years basically.

Tentco - October 24, 2023, 3:24 am

(Oct 24, 2023 03:12 AM)nodes Wrote: [\[CITE\]](#) libgcc_s_seh-1.dll was the missing file on my test exe I sent a coworker. It was just a standard program using std::cout, cerr, cin, getline, with literally nothing really complex inside. I was surprised something so minimal would require packaging, but maybe I've compiled wrong with g++ or something. There's so many options, I'm sure I am just missing a step or configuring it wrong, like not using a flag or cmake for something.

I am packing my own internal library just fine, I have multiple lib/folderHere/main.cpp and main.h files, it's only these external ones not packing.

I will look up what MT flag does, thanks!

Not sure about g++, if you compile with MSVC using /MT flag you will get a standalone binary with all dependencies linked.

Also If I remember while compiling with gcc or g++ you need to statically link your lib from args like -static-libstdc++ -static-libgcc

Hackers - October 24, 2023, 4:02 am

(Oct 24, 2023 03:24 AM)Tentco Wrote: [\[CITE\]](#) Not sure about g++, if you compile with MSVC using /MT flag you will get a standalone binary with all dependencies linked.

Also If I remember while compiling with gcc or g++ you need to statically link your lib from args like -static-libstdc++ -static-libgcc

Yeah, that's where I got to but haven't had a chance to test out the static links basically. I'll read up on MSVC, I really went with gcc because I had tried another compiler and had issues but gcc worked the first time so I didn't keep researching basically. That said, I struggled so hard with Cmake at first but it was a breeze a few days later once I understood the concepts better. I'll try out MSVC, thanks!

Tentco - October 24, 2023, 4:06 am

(Oct 24, 2023 04:02 AM)nodes Wrote: [\[CITE\]](#) Yeah, that's where I got to but haven't had a chance to test out the static links basically. Is MSVC recommended / more popular than gcc or others?

Depends Upon Personal Preference and factors, MSVC is popular and friendly for users comes integrated with VS Studio.

MSVC is commonly used for developing Windows desktop applications and drivers etc...

You don't really need to parse any args in MSVC as VS studio is optimized to do so.

While you could say GCC is used mostly in cross-platform development atleast that's what I've heard from others and is present in Linux ecosystem.

I personally use MSVC so I don't know much about gcc.

Hackers - October 24, 2023, 4:16 am

(Oct 24, 2023 04:06 AM)Tentco Wrote: [\[CITE\]](#) Depends Upon Personal Preference and factors, MSVC is popular and friendly for users comes integrated with VS Studio.

MSVC is commonly used for developing Windows desktop applications and drivers etc...

You don't really need to parse any args in MSVC as VS studio is optimized to do so.

While you could say GCC is used mostly in cross-platform development atleast that's what I've heard from others and is present in Linux ecosystem.

I personally use MSVC so I don't know much about gcc.

Ah, that makes sense. I'm currently just using VS Code with some extensions for syntax highlighting and then using GCC from Powershell and with CMake I'm using the GUI for binary build folder mixed with some command line for the actual final make command.

I'll look into VS Studio / MSVC. So far I'm happy with GCC and manually compiling because it's teaching me a lot, so I'll keep going and try to use the static links but if that doesn't work I'll swap. Ultimately my goal is to learn the most, but I don't want to gimp myself by not using available tools that may be easier lol.

Tentco - October 24, 2023, 4:21 am

(Oct 24, 2023 04:16 AM)nodes Wrote: [\[CITE\]](#) Ah, that makes sense. I'm currently just using VS Code with some extensions for syntax highlighting and then using GCC from Powershell and with CMake I'm using the GUI for binary build folder mixed with some command line for the actual final make command.

I'll look into VS Studio / MSVC. So far I'm happy with GCC and manually compiling because it's teaching me a lot, so I'll keep going and try to use the static links but if that doesn't work I'll swap. Ultimately my goal is to learn the most, but I don't want to gimp myself by not using available tools that may be easier lol.

I really suggest you to try VS studio if you're on windows.

Hackers - October 24, 2023, 4:39 am

(Oct 24, 2023 04:21 AM)Tentco Wrote: [\[CITE\]](#) I really suggest you to try VS studio if you're on windows.

Yeah, saw a lot online about it. I used to have it, I'll get it again and give it a go.

I am on Windows but spend much more time in Linux servers / CLI than Windows. I don't currently have a working Linux personal computer, but might wipe an old laptop and install a distro for testing purposes, or I can run it on a VPS. I do have QA testing Macs at work and 1 macbook air I could test cross platform programs on.

Ultimately my goal is only to start with Windows but to actually learn the Mac and Linux versions of everything too so I can build completely cross platform.

Do you know if it's recommended to have Mac and Windows with if statements for system environment detection, or is it better to compile CPP programs completely separately per environment?

I can't tell if it's better to use the if statements in the file, or to have two versions of the file in same dir, or to have two completely separate projects.

Tentco - October 24, 2023, 4:42 am

(Oct 24, 2023 04:39 AM)nodes Wrote: [\[CITE\]](#) Yeah, saw a lot online about it. I used to have it, I'll get it again and give it a go.

I am on Windows but spend much more time in Linux servers / CLI than Windows. I don't currently have a working Linux personal computer, but might wipe an old laptop and install a distro for testing purposes, or I can run it on a VPS. I do have QA testing Macs at work and 1 macbook air I could test cross platform programs on.

Ultimately my goal is only to start with Windows but to actually learn the Mac versions of everything too.

Do you know if it's recommended to have Mac and Windows with if statements for system environment detection, or is it better to compile CPP programs completely separately per environment?

I can't tell if it's better to use the if statements in the file, or to have two versions of the file in same dir, or to have two completely separate projects.

I'm confused by if-else statements do you mean checking if the program is running on windows or mac?

That's kind of useless since you won't be able to compile or run mac based code on windows and windows based code on mac, the difference due to libs and frameworks.

Better compile windows programs for windows and mac for mac.

Hackers - October 24, 2023, 4:53 am

(Oct 24, 2023 04:42 AM)Tentco Wrote: [\[CITE\]](#) I'm confused by if-else statements do you mean checking if the program is running on windows or mac?

That's kind of useless since you won't be able to compile or run mac based code on windows and windows based code on mac, the difference due to libs and frameworks.

Better compile windows programs for windows and mac for mac.

Yeah, that's what I meant, like system detection. I found some if statement formats that appear to only compile the code when it's on the target env, rather than a normal if statement.

If I'm remembering correctly it was #if with a hash tag in the example for conditionals during compiling. I tried it out, it didn't cause any errors, but it seems unnatural to have two sets of code in 1 file. The only reason I'm stuck on this though is because theoretically if I build out my own library of functions, the lib funcs could have the system conditions in the if statements and my primary files could use those funcs without caring what platform it's compiled on. I'm assuming that's the point of the #if syntax, but really really am out of my depth on this one for best practices in CPP

Thanks for all your help by the way, you're awesome.

Tentco - October 24, 2023, 4:56 am

(Oct 24, 2023 04:53 AM)nodes Wrote: [\[CITE\]](#) Yeah, that's what I meant, like system detection. I found some if statement formats that appear to only compile the code when it's on the target env, rather than a normal if statement.

If I'm remembering correctly it was `#if` with a hash tag in the example for conditionals during compiling. I tried it out, it didn't cause any errors, but it seems unnatural to have two sets of code in 1 file. The only reason I'm stuck on this though is because theoretically if I build out my own library of functions, the lib funcs could have the system conditions in the if statements and my primary files could use those funcs without caring what platform it's compiled on. I'm assuming that's the point of the `#if` syntax, but really really am out of my depth on this one for best practices in CPP

I still don't see the point of using it you won't be able to use WinAPI or any windows based code on mac or linux.

And you can build your own lib which contains Conditional statements since they are part of the C++ language itself and are not specific to any particular operating system, what you can't use is wrappers,libs that are made for one specific arch etc...

Hackers - October 24, 2023, 5:18 am

(Oct 24, 2023 04:56 AM)Tentco Wrote: [\[CITE\]](#) I still don't see the point of using it you won't be able to use WinAPI or any windows based code on mac or linux.

And you can build your own lib which contains Conditional statements since they are part of the C++ language itself and are not specific to any particular operating system, what you can't use is wrappers,libs that are made for one specific arch etc...

Ah, understood, but yeah not to run the code on others, i meant more like a way to have 1 codebase serve all. I guess I was thinking of things like this basically:

```
lib/someFunction/blah.cpp:
if Windows, use Win API or windows specific code
If Mac, use the Mac version of that code
If Linux, use that version
```

Then the blah.h header would have same input/output, and then in main.cpp I'd use blah. Then when I compile main.cpp in Windows, blah would use WIN API, but if I compiled on Mac it would use theirs, etc.

The idea was basically if I create functions that are cross platform with `#if` to only compile the code when it's on the target env, eventually I could code new programs without thinking of the actual environment at all in main.cpp because everything would be abstracted away in my library funcs to be able to compile on every target platform.

I'm assuming there are already libraries that do this for specific functions, and realistically I'm months away from even starting cross platform tests as I really want to have a firm baseline in 1 platform first.

Hackers - October 24, 2023, 5:19 am

(Oct 24, 2023 04:56 AM)Tentco Wrote: [\[CITE\]](#) I still don't see the point of using it you won't be able to use WinAPI or any windows based code on mac or linux.

And you can build your own lib which contains Conditional statements since they are part of the C++ language itself and are not specific to any particular operating system, what you can't use is wrappers,libs that are made for one specific arch etc...

Ah, understood, but yeah not to run the code on others, i meant more like a way to have 1 codebase serve all. I guess I was thinking of things like this basically:

```
lib/someFunction/blah.cpp:
```

if Windows, use Win API or windows specific code
If Mac, use the Mac version of that code
If Linux, use that version

Then the blah.h header would have same input/output, and then in main.cpp I'd use blah. Then when I compile main.cpp in Windows, blah would use WIN API, but if I compiled on Mac it would use theirs, etc.

The idea was basically if I create functions that are cross platform with #if to only compile the code when it's on the target env, eventually I could code new programs without thinking of the actual environment at all in main.cpp because everything would be abstracted away in my library funcs to be able to compile on every target platform.

I'm assuming there are already libraries that do this for specific functions, and realistically I'm months away from even starting cross platform tests as I really want to have a firm baseline in 1 platform first.

The -static linking worked by the way. Took my test exe from 148kb to 2.5mb but still better than Electron or Java lol!

Hackers - October 24, 2023, 5:20 am

(Oct 24, 2023 04:56 AM)Tentco Wrote: [CITE](#) I still don't see the point of using it you won't be able to use WinAPI or any windows based code on mac or linux.

And you can build your own lib which contains Conditional statements since they are part of the C++ language itself and are not specific to any particular operating system, what you can't use is wrappers,libs that are made for one specific arch etc...

Ah, understood, but yeah not to run the code on others, i meant more like a way to have 1 codebase serve all. I guess I was thinking of things like this basically:

```
lib/someFunction/blah.cpp:  
if Windows, use Win API or windows specific code  
If Mac, use the Mac version of that code  
If Linux, use that version
```

Then the blah.h header would have same input/output, and then in main.cpp I'd use blah. Then when I compile main.cpp in Windows, blah would use WIN API, but if I compiled on Mac it would use theirs, etc.

The idea was basically if I create functions that are cross platform with #if to only compile the code when it's on the target env, eventually I could code new programs without thinking of the actual environment at all in main.cpp because everything would be abstracted away in my library funcs to be able to compile on every target platform.

I'm assuming there are already libraries that do this for specific functions, and realistically I'm months away from even starting cross platform tests as I really want to have a firm baseline in 1 platform first.

By the way, the -static linking worked. Took my test exe from 150kb to 2.5mb though lol. Still better than Electron or Java though!

Tentco - October 24, 2023, 5:24 am

(Oct 24, 2023 05:19 AM)nodes Wrote: [\[CITE\]](#) Ah, understood, but yeah not to run the code on others, i meant more like a way to have 1 codebase serve all. I guess I was thinking of things like this basically:

```
lib/someFunction/blah.cpp:  
if Windows, use Win API or windows specific code  
If Mac, use the Mac version of that code  
If Linux, use that version
```

Then the blah.h header would have same input/output, and then in main.cpp I'd use blah. Then when I compile main.cpp in Windows, blah would use WIN API, but if I compiled on Mac it would use theirs, etc.

The idea was basically if I create functions that are cross platform with `#if` to only compile the code when it's on the target env, eventually I could code new programs without thinking of the actual environment at all in main.cpp because everything would be abstracted away in my library funcs to be able to compile on every target platform.

I'm assuming there are already libraries that do this for specific functions, and realistically I'm months away from even starting cross platform tests as I really want to have a firm baseline in 1 platform first.

That's what I am telling you, you CANNOT add different arch code in one codebase the compiler won't let you debug/build your binary.

Your question is theoretical.

Have you heard about IntelliSense? Try one IDE with this integrated and try linking different arch libs and compile, this will result in error.

And different arch has different executable extensions.

However you could create a dropper or some website that checks the type of arch you're running the program or visiting the website on and download compiled version of your program for that specific arch from some hosted server.

Hackers - October 24, 2023, 5:30 am

(Oct 24, 2023 05:24 AM)Tentco Wrote: [\[CITE\]](#) That's what I am telling you, you CANNOT add different arch code in one codebase the compiler won't let you debug/build your binary.

Your question is theoretical.

However you could create a dropper that checks the type of arch you're running the program and download compiled version of your program for that specific arch from some hosted service.

I guess I'm confused then. I've read a lot about using `#ifdef`, a thing called probe, and other methods to dynamically include headers or to only compile code in main.cpp based on platform, which would allow you to have a single main.cpp able to be compiled cross platform. I might be misunderstanding, and I really don't know enough here to know, but there are a few articles that say it is possible. BTW, I don't mean compiling for Mac on Windows, I just mean having the same repo able to be compiled on both but separately.

AFAIK, the compiler would ignore the other platforms, requiring me to debug and compile the program on the other platforms individually, but essentially using the same codebase as long as everything is scoped properly per system.

Tentco - October 24, 2023, 5:34 am

(Oct 24, 2023 05:30 AM)nodes Wrote: [\[CITE\]](#) I guess I'm confused then. I've read a lot about using `#ifdef`, a thing called probe, and other methods to dynamically include headers based on platform, which would allow you to have a single main.cpp able to be compiled cross platform. I might be

misunderstanding, and I really don't know enough here to know, but there are a few articles that say it is possible. BTW, I don't mean compiling for Mac on Windows, I just mean having the same repo able to be compiled on both but separately.

AFAIK, the compiler would ignore the other platforms, requiring me to debug and compile the program on the other platforms individually, but essentially using the same codebase as long as everything is scoped properly per system.

```
#ifdef _WIN32
// Windows-specific code goes here if you're on windows then no problem.
#else
// Non-Windows (e.g., Linux or macOS) perform file I/O, memory r/w,
// there are limitations, What you cannot do is:
//Execute Platform-Specific Binaries
//Get Low-Level Privileges
//No libs support(libraries are typically platform-specific)
//If you try to link any other arch related frameworks/libs the code won't compile.
#endif
```

Hackers - October 24, 2023, 5:38 am

(Oct 24, 2023 05:34 AM)Tentco Wrote: [\[CITE\]](#) #ifdef _WIN32
// Windows-specific code goes here if you're in windows then no problem.
#else
// Non-Windows (e.g., Linux or macOS) perform file I/O, memory r/w,
// there are limitations, What you cannot do is:
Execute Platform-Specific Binaries
Get Low-Level Privileges
No libs support(libraries are typically platform-specific)
If you try to link any other arch related frameworks/libs the code won't compile.
#endif
Okay, yeah that looks similar to one I tried and is where my thoughts came from.

I'll trust you on this one for sure & I'll be sticking to Windows for a while anyways, but this is good to know. I'll probably come back later to make Linux or Mac versions for things I build in Windows, but glad I didn't go down this path too far trying to do this. Sounds like it's a complete rewrite anyways, so better to have it in its own repo.

Cheers

Tentco - October 24, 2023, 5:39 am

(Oct 24, 2023 05:38 AM)nodes Wrote: [\[CITE\]](#) Okay, yeah that looks similar to one I tried and is where my thoughts came from.

I'll trust you on this one for sure & I'll be sticking to Windows for a while anyways, but this is good to know. I'll probably come back later to make Linux or Mac versions for things I build in Windows, but glad I didn't go down this path too far trying to do this. Cheers

At some point you will need to release compiled version of your software for different OS.
Good Luck.

stdio - October 24, 2023, 11:25 pm

(Oct 24, 2023 04:21 AM)Tentco Wrote: [\[CITE\]](#) I really suggest you to try VS studio if you're on windows.

I agree to this, VS is best option for targetting windows.. GCC good for cross platform

Also VS 2022 allows building for Linux as well now, and If you move to C# .net later on you can create cross platform apps easily. I used to code C++ only but have switched to C# for majority work. I think its way better and easier unless you really really need low level programming.

Hackers - October 25, 2023, 12:10 am

(Oct 24, 2023 11:25 PM)stdio Wrote: [\[CITE\]](#) I agree to this, VS is best option for targetting windows.. GCC good for cross platform

Also VS 2022 allows building for Linux as well now, and If you move to C# .net later on you can create cross platform apps easily. I used to code C++ only but have switched to C# for majority work. I think its way better and easier unless you really really need low level programming.

I don't really need low level immediately to be honest, but want to learn it for the sake of learning it. However, it's wicked fast and I do a lot of work with various blockchains, AI, and want to get into custom devices so C++ feels like it may be a better choice.

The main reason I skipped over thinking about C# is because I feel like it may be better for me to learn C++ and then C# after rather than vice versa, but if they're not even close in syntax or funcs then maybe it doesn't matter.

That's cool that VS 2022 has Linux compiling, will definitely check that out, but for now am sticking to Windows. That said, I imagine 99% of my future dev will be on Linux servers/devices so I might rethink my plans and actually start with Linux.

I do want to release at least one full piece of software for Windows first for portfolio reasons & to fulfill my childhood dream of doing so (used Visual Basic about 17-18 years ago but was a script kiddy at the time and didn't actually write anything from scratch in VB). I've done it with Java, Electron, and Python, but I've basically always wanted to learn C++ or C to build out a Windows program.

Not gonna lie, I am so so excited by CPP and it just feels so natural while coding in it. I'll investigate C# some more, but idk if I'll be able to kick this feeling that C++ will serve me better longterm than C# will.

Thanks for all the help everyone, I was able to get my fully packaged exe working ?

Tentco - October 25, 2023, 12:19 am

(Oct 25, 2023 12:10 AM)nodes Wrote: [\[CITE\]](#) I don't really need low level immediately to be honest, but want to learn it for the sake of learning it. However, it's wicked fast and I do a lot of work with various blockchains, AI, and want to get into custom devices so C++ feels like it may be a better choice.

The main reason I skipped over thinking about C# is because I feel like it may be better for me to learn C++ and then C# after rather than vice versa, but if they're not even close in syntax or funcs then maybe it doesn't matter.

That's cool that VS 2022 has Linux compiling, will definitely check that out, but for now am sticking to Windows. That said, I imagine 99% of my future dev will be on Linux servers/devices so I might rethink my plans and actually start with Linux.

I do want to release at least one full piece of software for Windows first for portfolio reasons & to fulfill my childhood dream of doing so (used Visual Basic about 17-18 years ago but was a script kiddy at the time and didn't actually write anything from scratch in VB). I've done it with Java, Electron, and Python, but I've basically always wanted to learn C++ or C to build out a Windows program.

Not gonna lie, I am so so excited by CPP and it just feels so natural while coding in it. I'll investigate C# some more, but idk if I'll be able to kick this feeling that C++ will serve me better longterm than C# will.

Thanks for all the help everyone, I was able to get my fully packaged exe working ?
I've programmed in C# for 1 year with JIT compilation and it's a good language for small software afterall, performance wise on same terms with Java, Python.
The only thing it lacks is better anti-analysis/protection of your software because anyone can convert it to CIL, I would switch over again if AOT ever becomes fully compatible with windows.
The package manager called Nuget also has rich libs/resources just like pypi.

Hackers - October 25, 2023, 12:22 am

(Oct 25, 2023 12:19 AM)Tentco Wrote: [\[CITE\]](#) I've programmed in C# for 1 year with JIT compilation and it's a good language for small software afterall, performance wise on same terms with Java, Python.

The only thing it lacks is better anti-analysis/protection of your software because anyone can convert it to CIL, I would switch over again if AOT ever becomes fully compatible with windows.
The package manager called Nuget also has rich libs/resources just like pypi.

Ahhh, yeah, that's what I am worried about. I already know Python, I dabbled in Java, if C# is similar in performance then I'd choose Python every time, especially because of how simple it is to use. You probably just hammered the nail in the C# coffin for me to be honest lol, unless there are reasons to use it over Python besides performance.

Tentco - October 25, 2023, 12:30 am

(Oct 25, 2023 12:22 AM)nodes Wrote: [\[CITE\]](#) Ahhh, yeah, that's what I am worried about. I already know Python, I dabbled in Java, if C# is similar in performance then I'd choose Python every time, especially because of how simple it is to use. You probably just hammered the nail in the C# coffin for me to be honest lol, unless there are reasons to use it over Python besides performance.

Well C# is static and python is dynamic, there's a little difference between performance but if you use PyObjects or use nuitka, ofcourse you can match same speed as .NET, choosing between them is tricky.

I would suggest you already know python, just go with c++ for now.

Hackers - January 8, 2024, 7:17 am

(Oct 25, 2023 12:30 AM)Tentco Wrote: [\[CITE\]](#) Well C# is static and python is dynamic, there's a little difference between performance but if you use PyObjects or use nuitka, ofcourse you can match same speed as .NET, choosing between them is tricky.

I would suggest you already know python, just go with c++ for now.

Coming back to this talk about static linking - is there a good way to bring in a full package of many headers, but then strip out any that aren't used? For example, if I bring over a folder of 30 headers and only use 2 of them that only then include 4 more as dependencies, is there a tool that would accurately find the 6 total headers & not compile with the other 24? I guess my real question is whether compiling C++ takes into account used vs unused files in the source or not.