

Learning Outcome: Students will gain experience in creating and using customized dynamic link library (DLL) and debugging.

1. (Creating customized DLL) Create dynamic link library written in C++.

- Create a DLL project named **Lab9DLL** in Visual Studio. If you don't know how to do it, please refer to <https://docs.microsoft.com/en-us/cpp/build/walkthrough-creating-and-using-a-dynamic-link-library-cpp?view=msvc-160>

- Add the following dll functions to the **dllmain.cpp** file:

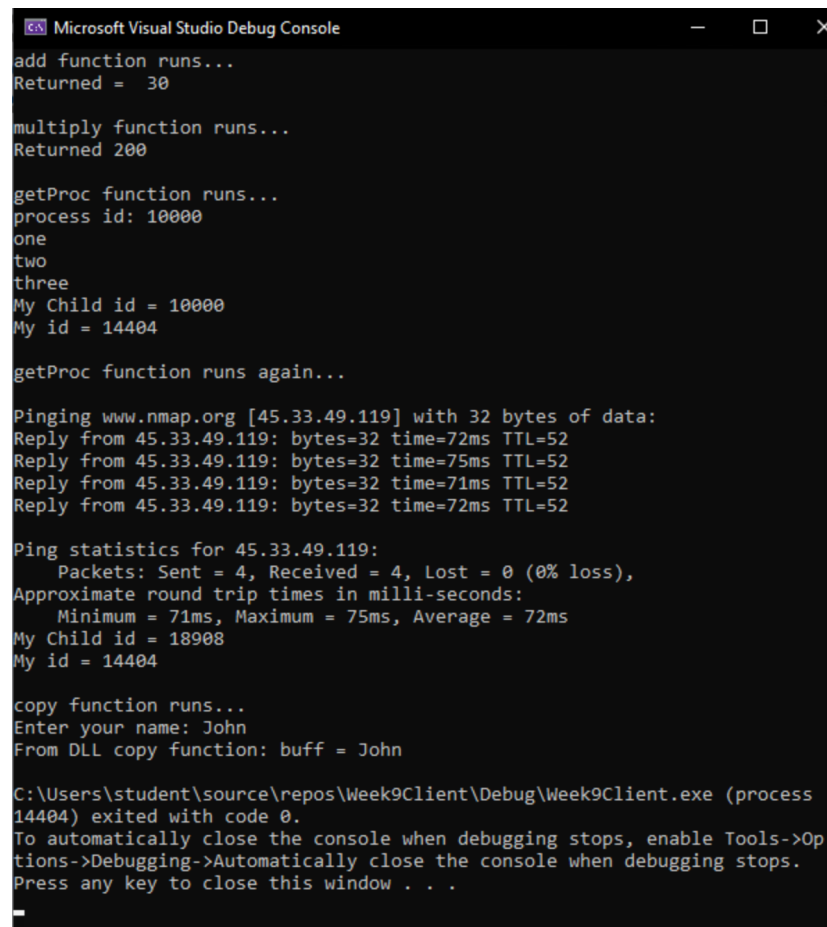
```
1) int add(int x, int y){
    Return x + y;
}
2) int multiply(int x, int y){
    return x * y;
}
3) void copy(char *name){
    char buff[30];
    strcpy(buff, name);
    printf("From copy dll: buff = %s\n", buff);
}
4) void createProc(char *applicationName, char *cmdline)
```

- Creates a child process that will execute the `applicationName` module. The `cmdline` is a string containing command line arguments separated by blank space that is to be passed into the application. The `applicationName` parameter can be **NULL**. In that case, the module name must be the first token in the `cmdline` string.

You will need to use the `CreateProcess` function. You can find information at <https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa>

- The parent process must wait until the child process is terminated and then print its own pid and child pid.
- Don't forget to use keywords `extern "C" __declspec(dllexport)` in the function headers. Without the `extern "C"` keyword, application written in C may not be linked to your dll functions. The `__declspec(dllexport)` modifier is used to export data, function, and classes.
- Click Build->Build Solution to create **Lab9DLL.dll**. You may need to check these settings in Project Properties -> C/C++ -> Code Generation.
 - Security Check should be set to "Disable Security Check"

- Basic Runtime Checks should be set to “Default”
2. (Using DLL) Create an empty project named **Lab9Client** that uses the dll functions you wrote in question. Add the provided **lab9ClientMain.c** file to the project. Modify the C file so that it uses the DLL functions. Your final output should look like:



```
Microsoft Visual Studio Debug Console

add function runs...
Returned = 30

multiply function runs...
Returned 200

getProc function runs...
process id: 10000
one
two
three
My Child id = 10000
My id = 14404

getProc function runs again...

Pinging www.nmap.org [45.33.49.119] with 32 bytes of data:
Reply from 45.33.49.119: bytes=32 time=72ms TTL=52
Reply from 45.33.49.119: bytes=32 time=75ms TTL=52
Reply from 45.33.49.119: bytes=32 time=71ms TTL=52
Reply from 45.33.49.119: bytes=32 time=72ms TTL=52

Ping statistics for 45.33.49.119:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 71ms, Maximum = 75ms, Average = 72ms
My Child id = 18908
My id = 14404

copy function runs...
Enter your name: John
From DLL copy function: buff = John

C:\Users\student\source\repos\Week9Client\Debug\Week9Client.exe (process
14404) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Op
tions->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

- You will need to create .exe file from the provided **cmdArgs.c** file so that the binary file can be executed when the **createProc** dll function runs.
- Recall/note that to explicitly link to a DLL, your client program must:
 1. Call **LoadLibrary()** to load the DLL and obtain a module handle.
 2. Call **GetProcAddress()** to obtain a function pointer to each exported function that the application will use.
 - To do this, you must have a typedef that defines the call signature of the exported functions.
 3. Call **FreeLibrary()** when done with the DLL.

If you need further information or sample code, feel free to google. You may want to read the “How to Explicitly Link to DLL” section from <https://docs.microsoft.com/en-us/cpp/build/linking-an-executable-to-a-dll?view=vs-2019>

3. (Revisiting buffer overflow) You will need to use the VS debugger (or Immunity) for this part. Reuse the **lab9ClientMain.c** file to run the `dll copy` function. Enter a string to be used in the copy function so that you can set the `EBP` register to “AAAA” and `EIP` to “BBBB”. Your program will crash eventually. Include your string at the bottom of the source file `Lab9ClientMain.c`.

```
EIP = 42424242 ESP = 012FFC78 EBP = 41414141
```

4. (Intro to Assembly) Write assembly program named **lab9.asm**:

```
var = 15    //use the data section to declare global variable var
EBX = (x%4 + 2) * 5;
```

Deliverables:

You must submit source files (`dllmain.cpp`, `Lab9ClientMain.c`, and `lab9.asm`) to the GitHub Assignment09 repository before 10/27/2021 at 7 PM.

NAME:

Each signature is worth $1/N$ of your lab grade where N is the number of signatures.

1. Student could create DLL.
2. Student wrote the `createProc` function as specified in the lab document.
3. Student could write an application program that uses DLL.
4. Student could set the EBP and EIP to 41414141 and 42424242, respectively.
5. Student could write arithmetic instructions in x86 assembly.