# 1 Memory spaces

There are two memory spaces within C. One memory space is the data memory space addressed by data pointers, e.g.

```
int* myPtr;
```

The other is a code memory space addressed by function pointers, e.g.

```
int (*myFunc)(void);
```

For portability reasons function pointers and data pointers are not compatible. They may not even have the same size and therefore even casts to seemingly innocent `void*` may yield unexpected results and compiler warnings.

This could happen on a Harvard based plattform, where code and data are stored in different memories. This is common for microcontrollers where code could be stored in flash memory. Also architectures with segmented memory may be suspectible to such differences, e.g. x86-Realmode.

Some APIs and frameworks may require untyped function pointers to support a large number of functions with different signatures. This can be archived by using a default function pointer, e.g.

```
void (*)(void)
```

and casting it to the desired target function pointer. This is done in the GLX API[3] with,

```
typedef void (*__GLXextFuncPtr)(void);
extern __GLXextFuncPtr glxGetProcAddress(const GLubyte*)
```

[2] where `GLXextFuncPtr` serves as arbitrary function pointer and `glxGetProcAddress` returns such a pointer depending on the string passed to it.

Despite the issues involved, important API calls like the `dlsym`[1] function return `void*` for function pointers and therefore rely on Neumann architectures with a single memory space for both code and data. This issues is well known and not solved. Some compilers support the necessary cast as an extension.[4].

# References

[1] The Open Group Base Specifications Issue 6, 2004
http://pubs.opengroup.org/onlinepubs/009695399/functions/dlsym.html

[2] Brian Paul, glx.h, Mesa 3-D graphics library, Version 6.5

[3] Brian Paul and Jon Leech, The Kronos Group Inc. `ARB_get_proc_address`, 3. January 2000, *http://www.opengl.org/registry/specs/ARB/get_proc_address.txt*

[4] Steve Clamage, C++ Standard Core Language Defect Reports and Accepted Issues, Revision 84, 12. Jan 2000
*http://www.open-std.org/jtc1/sc22/wg21/docs/cwg_defects.html*