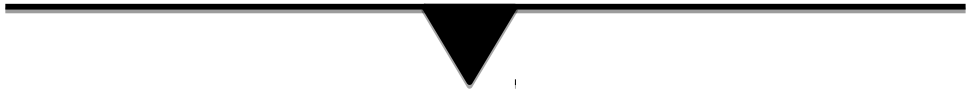
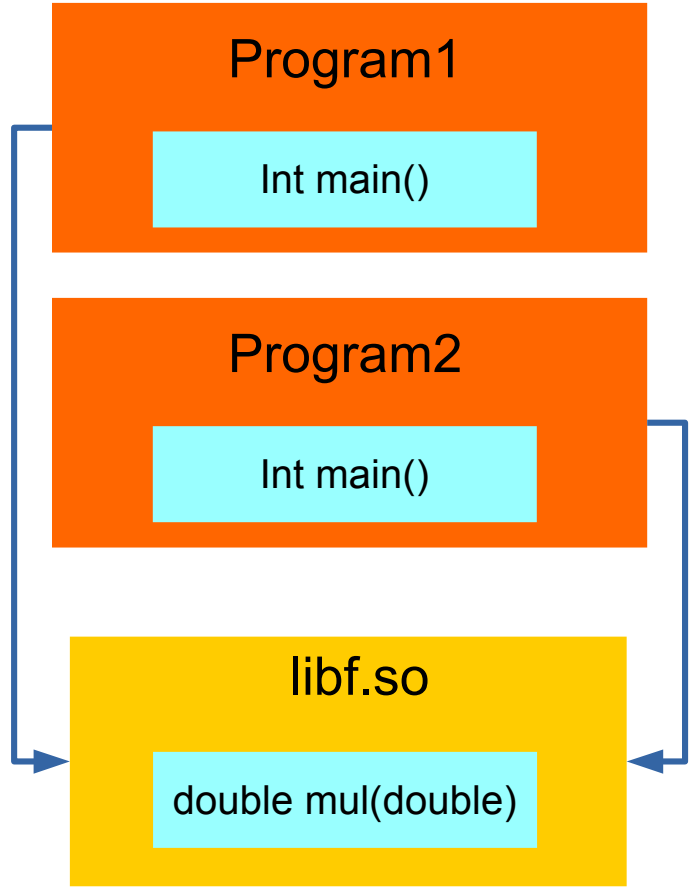
Shared library

- Collection of machine code (function and/or classes)
- Packed in ELF file
- NO main function
- Naming convection: lib<name>.so



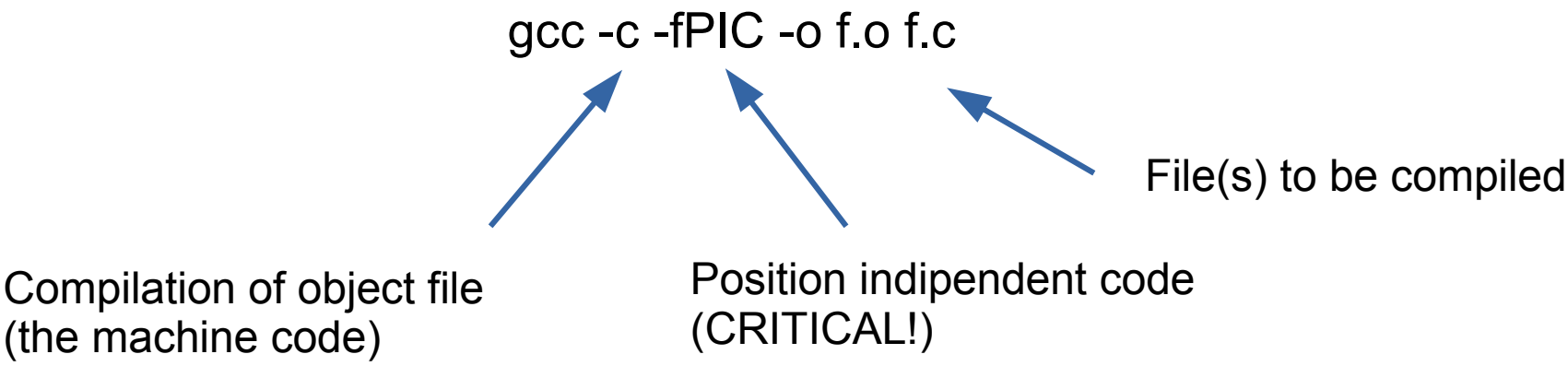
- The shared library can be called by programs
- Several programs access the same library
- Cannot be directly executed



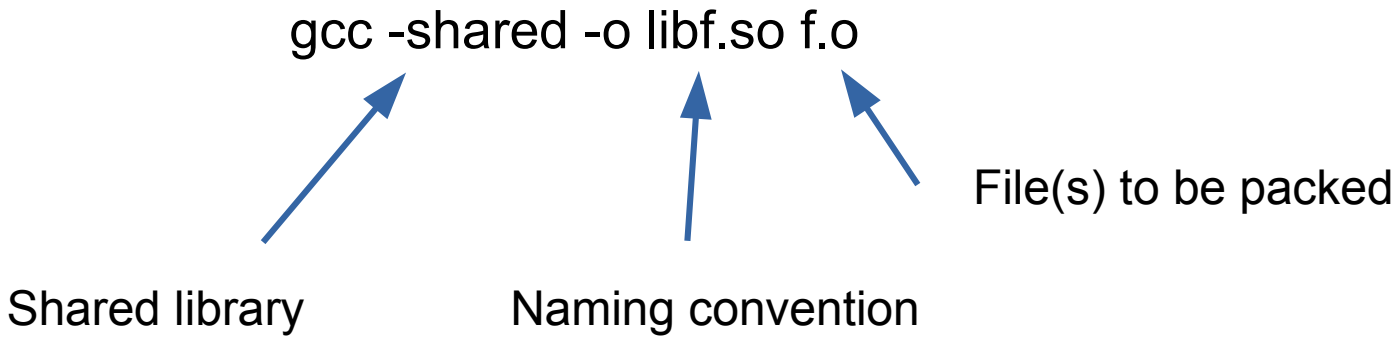
Compiling a shared library

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1. Compiling the machine code



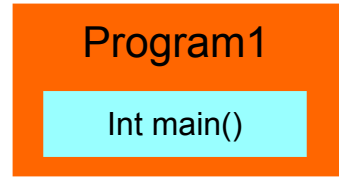
- 2. Packing in a shared library ELF file
  - Machine code will be not changed
  - Rebuild of library does not require recompilation (unless code is changed, of course)



Link to a shared library

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1. Compiling the machine code



gcc -c -o main.o main.c

Compilation of object file (the machine code)

File(s) to be compiled

- 2. Packing in an executable ELF file
  - Machine code will be not changed
  - Change in library does not require rebuild of program
  - Do not forget LD_LIBRARY_PATH

gcc -L. -o main.x main.o -lf

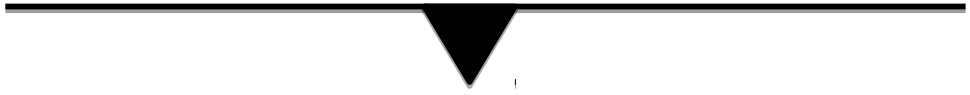
Shared library search directory

File(s) to be packed

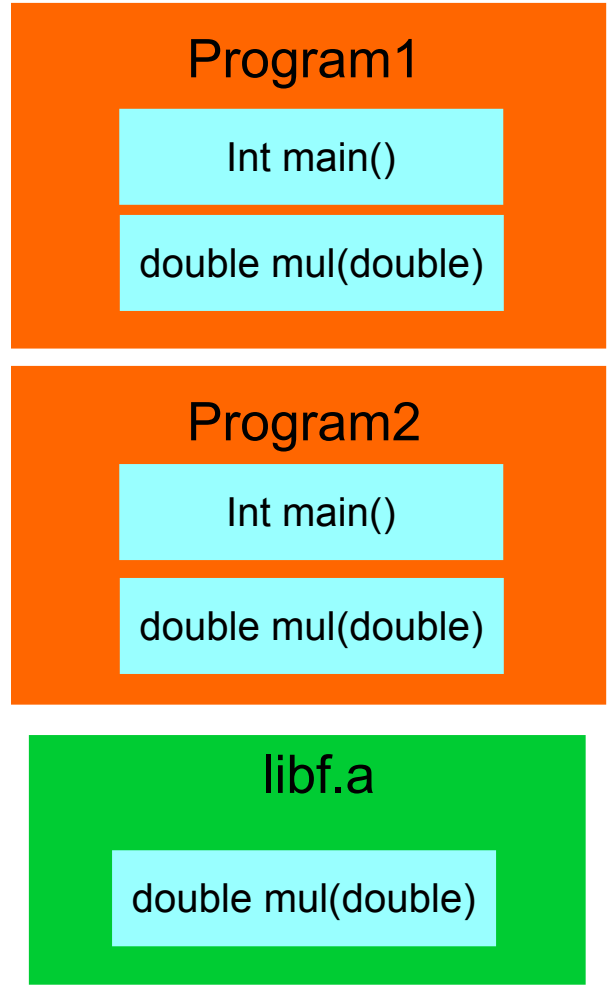
Libs to be linked

Static library

- Shared library
- Collection of machine code (function and/or classes)
- Packed in ELF file
- NO main function
- Naming convection: lib<name>.a



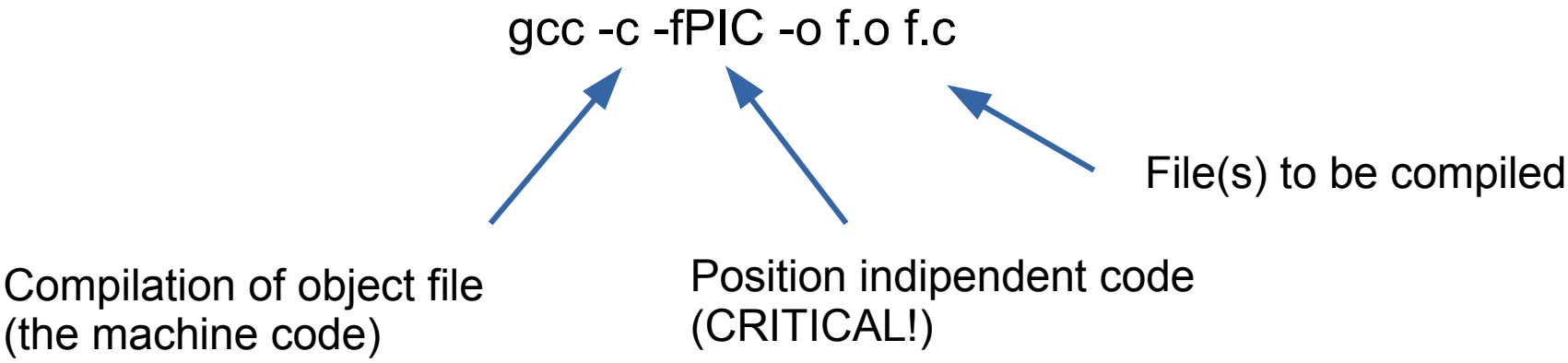
- The static library are included in the program
- Every program has his own copy of the library
- Library cannot be directly executed



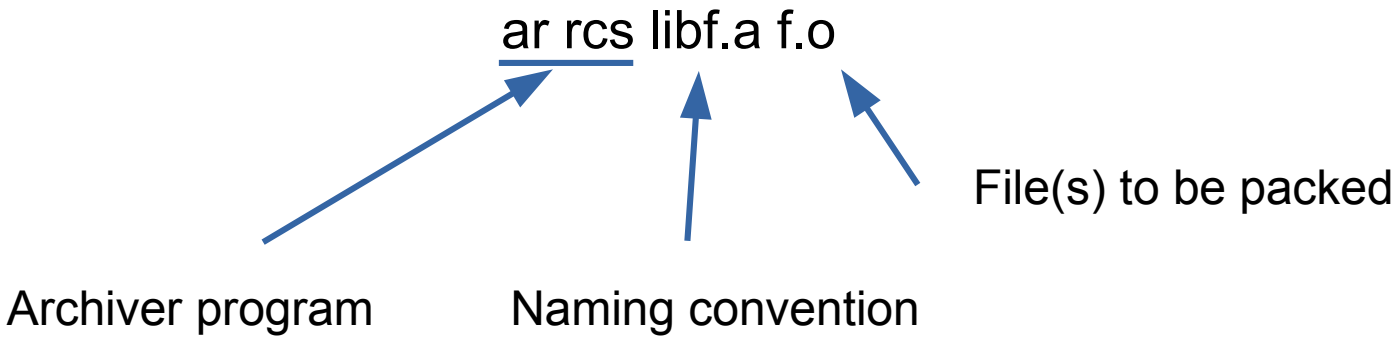
Compiling a static library

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1. Compiling the machine code (this part is common to shared library process)



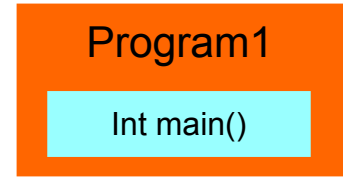
- 2. Archiving in a static library file
  - Machine code will be not changed
  - Rebuild of library does not require recompilation (unless code is changed, of course)



Link to a static library

TECHNISCHE UNIVERSITÄT DARMSTADT

- Exactly the same process of shared library
- 1. Compiling the machine code




```
gcc -c -o main.o main.c
```

Compilation of object file (the machine code)

File(s) to be compiled

- 2. Packing in an executable ELF file
  - Machine code will be not changed
  - Change in library does not require rebuild of program

```
gcc -L. -o main.x main.o -lf
```

Library search directory

Input file(s)

Libs to be linked

Notes

- Building of executable is exactly the same
- At linking gcc automatically recognize if the library is static or dynamic
  - In case you have both dynamic and static library, it is recommended to pass the complete library name

```
gcc -L. -o main.x main.o -l:libf.a
gcc -L. -o main.x main.o -l:libf.so
```

- Executable linking process in gcc is the same for static and dynamic library
- Compiling process (creation of object file) is the same for static and dynamic library
- Flag -fPIC is mandatory to build object that can be packed in a library
- Do not forget LD_LIBRARY_PATH to load shared library

Exercise

- Compile both a static and a dynamic library from the same files (f.c)
- Compare the symbol table, which are the differences?
- Read the elf info of the both libraries which are the differences?
- Compile the main file linking first time to a static and second time to a dynamic library
  - Compare the two symbol table
  - Compare the shared dependency library
- Try to execute the program, if there is a problem try to solve it

- If you do not remember which program print the symbol table, the elf info and the shared dependency go back to lesson slides.