Linking

"Linking is the process of collecting and combining various pieces of code and data into a single file that can be loaded (copied) into memory and executed. Linking can be performed at compile time, when the source code is translated into machine code; at load time, when the program is loaded into memory and executed by the loader; and even at run time, by application programs."

"Linkers play a crucial role in software development because they enable separate compilation. Instead of organizing a large application as one monolithic source file, we can decompose it into smaller, more manageable modules that can be modified and compiled separately. When we change one of these modules, we simply recompile it and relink the application, without having to recompile the other files."

Compiling is actually a multi-step process that ultimately produces relocatable object files, designated as .o files.

Example: `box-print.c`

Compile into a program:

```
gcc -o box-print box-print.c
```

Separate the printing functionality from `main` in source code:

```
box-print-func.c
box-print-test.c
```

Create object files:

```
gcc -c box-print-test.c
gcc -c box-print-func.c
```

View the object files with relocation information:

```
objdump -dr box-print-test.o
objdump -dr box-print-func.o
```

Link the object files together into an executable file:

```
gcc -o box-print box-print-test.o box-print-func.o
```

View the executable file:

```
objdump -d box-print
```

Now, archive the `print_box` function into a static library:

```
ar rs box-print-func.a box-print-func.o # add the object file
```

Now, compile `box-print-test.c` using the library:

```
gcc -o box-print box-print-test.c box-print-func.a
```

A library of relocatable objects can be created so that it can be used for linking at run time (load time) or during execution. Such as library is known as a shared library.

Shared libraries on Unix use the extension .so by convention. The extension for shared libraries on Windows is .dll.

Example of linking a shared library at runtime:

```
unix> gcc -o p2 main2.c ./libvector.so
```

In the above example, the compiler copies symbol table and relocation information from the shared library for use at runtime.

A shared library can be loaded at runtime by use of library functions. On Unix, these functions include:

```
dlopen -- loads a shared library into memory
dlsym  -- obtain a reference to an object
  (such as a function) within the library
dlclose
dlerror-- returns a description of the last
  error that occurred as a result of calling
  the above-functions.
```

The primary advantage of a shared library is that it saves on storage space and execution space (because multiple executables can share the code of a shared library at runtime).