Static Libraries

Static libraries have the extension **.a**. When a program is compiled, full copies of any loaded library routines are incorporated as part of the executable.

The following tools are used for maintaining static libraries:

> **ar** creates, updates, lists and extracts files from the library. The command:

```
$ ar rv libsubs.a *.o
```

will create **libsubs.a** if it does not exist, and insert or update any object files in the current directory.

> **ranlib** generates, and stores within an archive, an index to its contents. It lists each symbol defined by the relocatable object files in the archive. This index speeds up linking to the library. The command:

```
$ ranlib libsubs.a
```

is completely equivalent to running **ar -s libsubs.a**. While running **ranlib** is essential under some UNIX implementations, under Linux it is not strictly necessary, but it is a good habit to get into.

> **nm** lists symbols from object files or libraries. The command:

```
$ nm -s libsubs.a
```

gives useful information. **nm** has a lot of other options.

Modern applications generally prefer to use shared libraries, as it is more efficient and conserves memory. However, there are at least two circumstances where static libraries are still used:

1. For programs that are used early in system startup, before the tools to work with shared libraries are fully operational.

2. For programs that want to be completely self-contained and not have to deal with potential problems from system updates of libraries that are utilized by the application. This is typically done by either proprietary (and even closed source) application suppliers, or by other large vendors, such as Google or Mozilla. This is always controversial, because it is up to the vendor to make sure that any security holes or other bugs are fixed when they are discovered upstream in the included libraries.