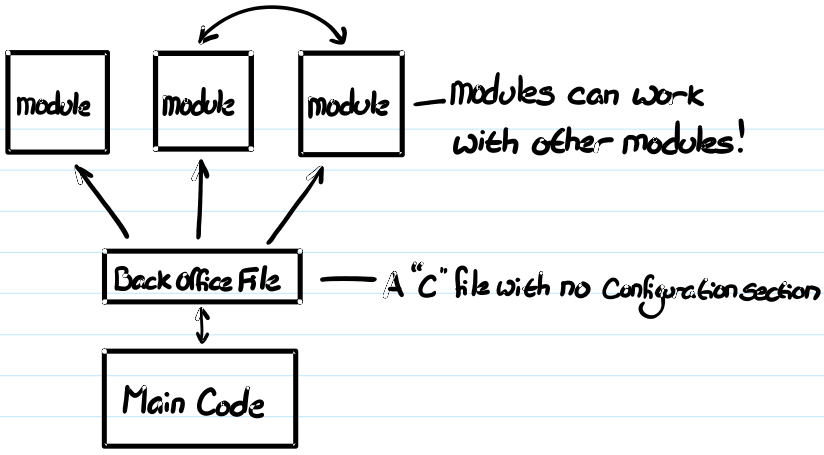

Modules in nesC lets you compartmentalize your code for easier readability and usability. Your main file code will be cleaner, and allows for reusability. This document explains the basic form of a module and how it works.

Each module has a "job" to do and includes events/commands ("tasks") to do it

Think of events/commands as functions

What does a module look like?

A module is comprised of three files:

- A Interface file taskname.nc
- A "P" file tasknameP.nc
- A "C" file tasknameC.nc

[In the skeleton code, you can find module files in the "lib" folder]

File Breakdown

Taskname.nc

Acts similarly to a .h or header file. It holds the names of commands or events you wane module to execute.

ex. in CommandHandler.nc

```
interface CommandHandler {
    event void printNeighbors();
}
```

ex in SimpleSend.nc

```
interface SimpleSend {
```

1

```
interface SimpleSend {
    command error_t send(pack_msg, uint16_t dest);
}
```

TasknameP.nc

The "P" file of the module is considered the true "module" file, and is the private component of all these files. Here is where your code implementations are stored. This file has two sections:

← Think about "P" as "private"!

```
Module TasknameP {
    provides interface Taskname;
}
```

The line provides interface allows the use of our declared commands and events in our "header" Taskname.nc

```
implementation {
}
```

Implementation allows you to write out your command and event code within its brackets

Note: Using uses interface within the module bracket allows the use of other interfaces that have been written in your code

TasknameC.nc

The "C" file is the public face of the module. It allows you to connect your module to your main code (or other modules!). This file has two sections.

← "C" seems to stand for "Configuration" but I tend to remember it as "Connection" or "Connector"

```
Configuration TasknameC {
    provides interface Taskname;
}
```

Configuration is the public component that allows wiring to other interfaces. A "C" file with a empty Configuration is a main file that only wires to other "C" files. See NodeC.nc

```
Implementation {
    Components TasknameP as Task;
}
```

Implementation allows the actual link to other components, letting you use Commands in modules with your main code and other modules.