More details about Makefiles

Files arising when compiling a C program

- `<file>.c`: Source code.
  In the example, the file `lib/input.c` contains the source code for the library, and the files `src/program.c` and `src/output.c` contains the source code for the program.
- `<file>h`: Include files.
  These file contain the function declarations of the libraries which are used by the program. In the example, this is `include/input.h`. They need to be provided as part of the library.

- `<file>.o`: Object files
  These files contain assembly code for all the functions defined in `<file>.c` They are produced by the compiler from the source code files. In the example, they are `lib/input.o`, `src/program.o` and `src/output.c` contains the source code for the program.
- `<file>.so` Shared library files
  These files contain assembly code which is suitable for inclusion in several programs. These files are produced by the compiler from the object files containing all functions.

Conventions for Makefiles

- `$(NAME)` uses the variable `NAME`. The value of a variable is set with `<VariableName>=<value>`. An example would be `TARGETS=program`. The variable `CC` is pre-defined as `cc`, which is the C-compiler.
- A rule has the form

  ```
  <target>: <file1> ... <filen>
          <command>
  ```

  Meaning of rule: If file `target` does not exist or any of `file1...filen` are newer than `target`, the command `command` is executed

- Special variables used in rules: `$@` means target, `$<` means `file1`.
- Special case of rule specification:

  ```
  %.: %.c
      <command>
  ```

  means that every file ending in `.o` depends on the corresponding file ending in `.c` and may be created via `command`