Operating Systems Lab (C+Unix)

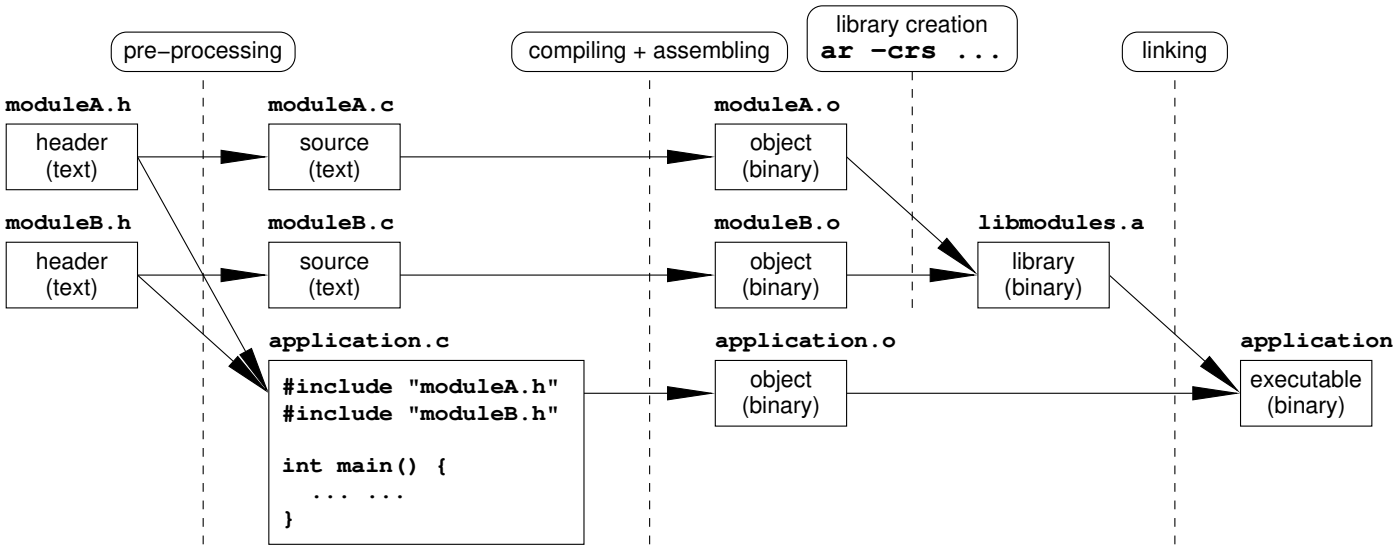
Enrico Bini

University of Turin

Outline

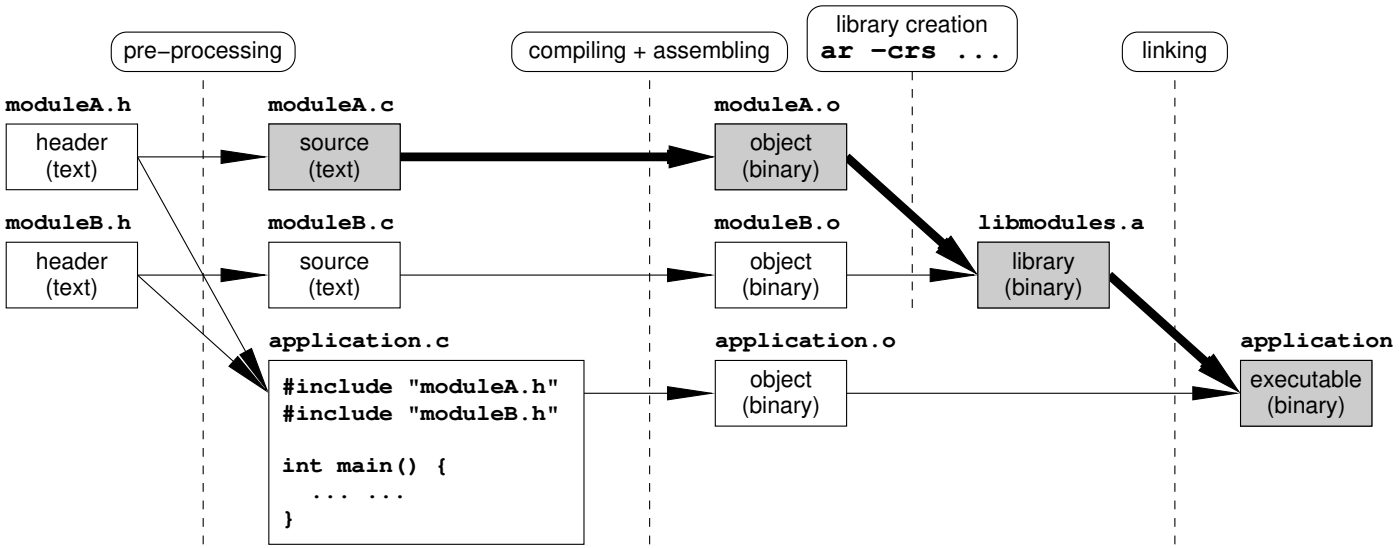
Usage of `make`: help with dependencies in projects

- In large projects, pre-processor `#include` and modules/libraries usage establish dependencies



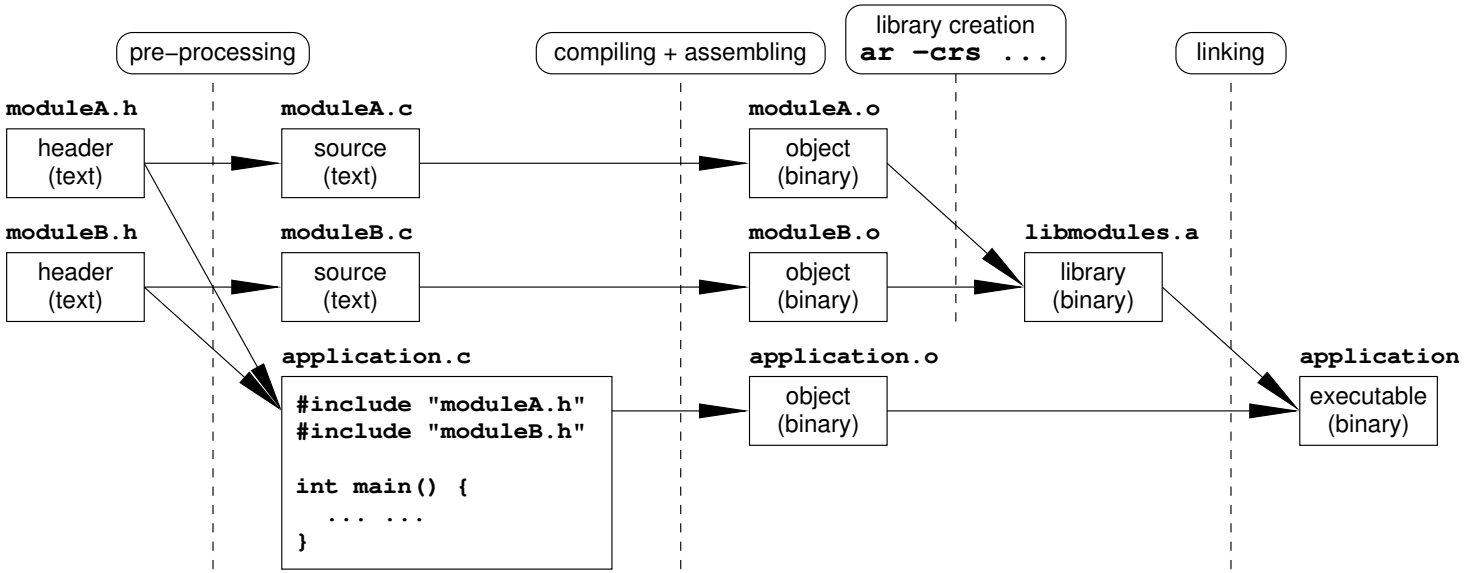
Usage of `make`: help with dependencies in projects

- In large projects, pre-processor `#include` and modules/libraries usage establish dependencies



- If the implementation in "`moduleA.c`" is changed, no need to re-compile all sources. Just the following steps
  1. compiling with `gcc -c moduleA.c`
  2. linking with
     - ★ using libraries: `ar -crs libmodules.a moduleA.o moduleB.o` + `gcc application.o -lmodules -o application`
     - ★ without libraries: `gcc application.o moduleA.o moduleB.o -o application`

Expressing dependencies with `Makefile`

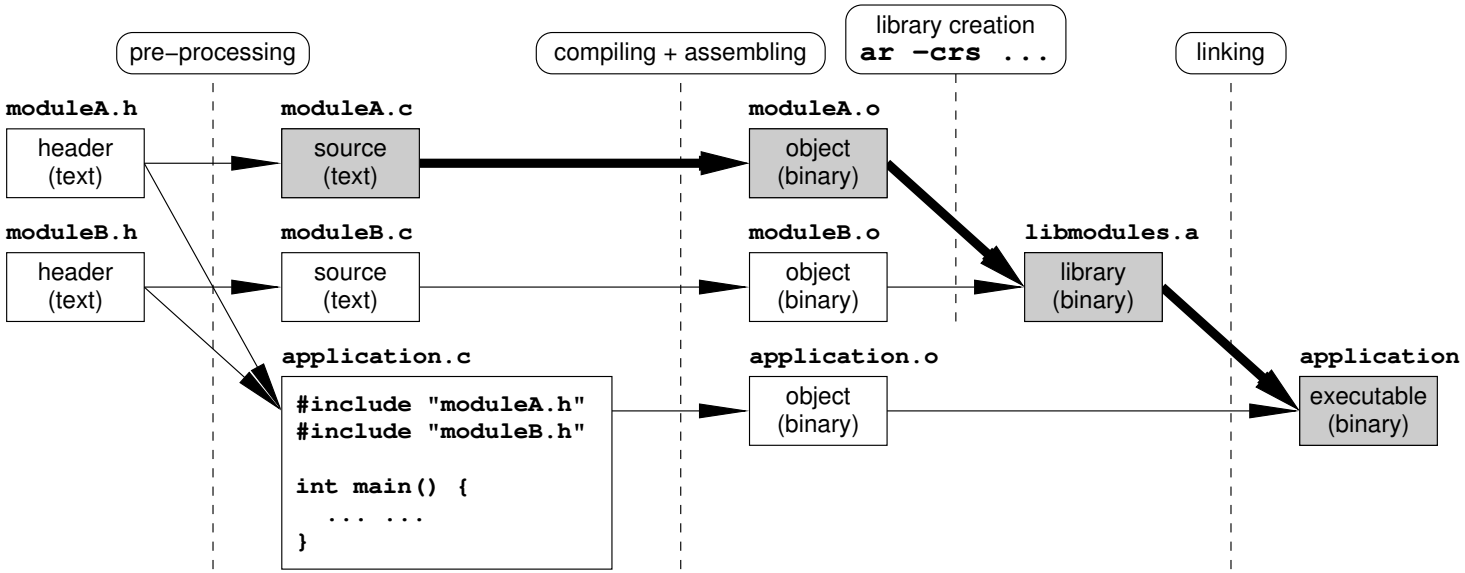


- The dependecies are given in the `Makefile` (check *`Makefile-full`*)
- Everything from the charater # until the end of line is a comment
- The format of rules in `Makefile` is:

```
target: ingredientA ingredientB     # dependencies
[TAB]   recepie-to-make-target
```

The recepie **must** start after the TAB character (not 8 spaces)

Expressing dependencies with `Makefile`



- The dependecies are given in the `Makefile` (check *`Makefile-full`*)
- Everything from the charater # until the end of line is a comment
- The format of rules in `Makefile` is:

  ```
  target: ingredientA ingredientB    # dependencies
  [TAB]   recepie-to-make-target
  ```

  The recepie **must** start after the TAB character (not 8 spaces)
- If `moduleA.c` is changed, by invoking `make application`
  1. the timestamp of "predecessors" of target `application` are checked
  2. all "recepies" from newer predecessors to `application` are made

Implicit rules

- If no *explicit* rule is set in `Makefile` for `<target>`, then by invoking `make <target>` some *implicit* rules are tried
- `make hello` uses implicit rules to produce the executable `hello`
  - If `hello.c` is **not** present, it returns an error
    ```
    make:  *** No rule to make target 'hello'.    Stop.
    ```
  - If `hello.c` is present in the current directory it compiles by
    ```
    cc hello.c -o hello
    ```
  - If `make hello` is launched again, `hello.c` is **not re-compiled** again.
    ```
    make:  'hello' is up to date.
    ```

1. `make <target>` tries to make the executable `<target>` from the object file `<target>.o` or from the source `<target>.c` by
   ```
   gcc <target>.c -o <target>
   ```
2. `make <file>.o`
   tries to make the object file from a source file `<file>.c` by
   ```
   gcc -c <file>.c
   ```

- The command `make` exploits implicit rules (check *`Makefile-simple`*)

Makefile: example of module

- Example of Makefile for the "matrix" module. Remember:
  - *matrix.h*, the header file of the module (the interface)
  - *matrix.c*, the implementation of the module
  - *application.c*, an example of code using the module
  - *Makefile*, giving instructions to make
- Everything from the charater # until the end of line is a comment
- The Makefile may optionally start with project-dependent declarations of variables
- Targets are not necessarily files to be produced, just "things to be done"
- The format of an explicit rule is:

```
target: ingredientA ingredientB     # dependencies
[TAB]   recepie-to-make-target
```

  The recepie **must** start after the TAB character (not 8 spaces)
- Try to modify some files of the "matrix" module and launch make

Makefile: invoking an explicit rule

- If the following explicit rule is listed in the `Makefile`

```
target: ingredientA ingredientB    # dependencies
[TAB]   step1
[TAB]   step2
[TAB]   step3
```

  then by launching `make target`

  - if a file "`target`" exists and is more recent than "`ingredientA`" **and** "`ingredientB`" then nothing is made (it means that "`target`" was made already)
  - otherwise `make` is invoked for any ingredient that is newer than `target`
  - when all ingredients are made, all commands (`step1`, `step2`, `step3`) are executed

- If no "ingredient" is specified, then the commands are always executed