Using GNU Automake with gm2

- `gm2` (from gcc-14 onwards) supports `-M`, `-MM`, `-MD`, `-MMD`, `-MP`, `-MT` and `-MQ` ⟨https://gcc.gnu.org/onlinedocs/gcc/Preprocessor-Options.html⟩

- `-M` and `-MM` are run without generating any code (in the preprocessor stage).

- `-MD`, `-MMD` are run with the compiler and therefore the command line can generate dependencies and also code.
  - the example project here uses `-MMD` as one command line can be used to generate an object file and a dependency file.

Example project

- the example project `m2-autoconf-example` is available via:

- ```
  $ git clone https://github.com/gaiusm/m2-autoconf-example
  ```

- consists of a tiny example modula-2 project containing modules:
  - `top.mod` the application level module
  - `z.def` and `z.mod` the lowest level module exporting `zproc`
  - module `a.def` and `a.mod` (also imports module `z`)
  - module `b.def` and `b.mod`
  - module `a.def` and `a.mod` (also imports module `z`)
  ```

configure and compile project

```
$ mkdir build-m2-autoconf-example
$ cd build-m2-autoconf-example
$ ../m2-autoconf-example/configure
$ make
gm2 -MMD -MT a.o -MP -MF .deps/a.d  -I../m2-autoconf-example \
    -c ../m2-autoconf-example/a.mod
gm2 -MMD -MT b.o -MP -MF .deps/b.d  -I../m2-autoconf-example \
    -c ../m2-autoconf-example/b.mod
gm2 -MMD -MT c.o -MP -MF .deps/c.d  -I../m2-autoconf-example \
    -c ../m2-autoconf-example/c.mod
gm2 -MMD -MT z.o -MP -MF .deps/z.d  -I../m2-autoconf-example \
    -c ../m2-autoconf-example/z.mod
gm2 -MMD -MT top.o -MP -MF .deps/top.d  -I../m2-autoconf-example \
    -c -fscaffold-main -fscaffold-dynamic \
    ../m2-autoconf-example/top.mod
gm2 a.o b.o c.o z.o top.o -o top
```

- modules: `a`, `b`, `c`, `z` are all compiled using the same rule.

configure and compile project

- module `top` is compiled to include the C function `main` and also the runtime scaffold to initialize all modules.
  - but the rule does not perform the link

- the final `gm2` command links and place the executable into `top`

configure and compile project

- on inspection we can see that the dependencies for module `c` are stored in file `.deps/c.d` with contents:

-
```
c.o: \
 ../m2-autoconf-example/c.mod \
 ../m2-autoconf-example/c.def \
 ../m2-autoconf-example/z.def
```

Makefile.am

```
SUFFIXES = .mod .def .o .a

DEPFLAGS=-MMD -MT $@ -MP -MF .deps/$*.d

DEPS = a.o \
       b.o \
       c.o \
       z.o \
       top.o

bin_PROGRAMS = top
top_SOURCES = top.mod \
       a.mod \
       b.mod \
       c.mod \
       z.mod
```

Makefile.am

```
top$(EXEEXT): $(DEPS)
    gm2 $(DEPS) -o $@


top.o: top.mod
    gm2 $(DEPFLAGS) -I$(srcdir) -c \
        -fscaffold-main -fscaffold-dynamic $<

%.o: %.mod .deps/%.d
    @test -z .deps || mkdir -p .deps
    gm2 $(DEPFLAGS) -I$(srcdir) -c $<
```

Makefile.am

```
DEPFILES=$(top_SOURCES:%.mod=.deps/%.d)

$(DEPFILES):

include $(wildcard $(DEPFILES))
```

rundemo

- the script `rundemo` runs `make` then modifies module `z` to export procedure `bar`. It also modifies module `c` to call `bar`
  - `make` is run again which results in a recompile of modules: `a`, `c` and `z`. It then links all modules

rundemo

```
$ bash rundemo
gm2 -MMD -MT a.o -MP -MF .deps/a.d -I../m2-autoconf-example \
    -c ../m2-autoconf-example/a.mod
gm2 -MMD -MT c.o -MP -MF .deps/c.d -I../m2-autoconf-example \
    -c ../m2-autoconf-example/c.mod
gm2 -MMD -MT z.o -MP -MF .deps/z.d -I../m2-autoconf-example \
    -c ../m2-autoconf-example/z.mod
gm2 a.o b.o c.o z.o top.o -o top
```