A ferramenta make

A - Introdução

- O C é orientado para a produção de aplicações informáticas de grande dimensão

- Na geração de aplicações informáticas há ficheiros gerados a partir de outros: logo, os segundos têm obrigatoriamente data posterior aos primeiros.
  - Primeiro: ficheiros fonte (extensão *.c* e *.h*)
  - Segundo: ficheiros objecto (extensão *.o*)
  - Terceiro: programa executável

A ferramenta make

- As funções logicamente relacionadas devem ser colocadas no mesmo ficheiro (módulo).
- Os ficheiros de inclusão do programador devem identificados em primeiro lugar e incluídos apenas nos módulos que usam as declarações.

- O make gera um programa coerente, compilando apenas os ficheiros fonte alterados.
- Os ficheiros envolvidos e comandos de construção são indicados no ficheiro `makefile` (ou `Makefile`).

A ferramenta make

B - Sintaxe

- lista de comandos, iniciados na primeira coluna, separados por linhas em branco na forma

  Id : {Id}* \n

  { \t ComandoUnix \n}*

  { }

  – A lista de identificadores de ficheiros após : pode prosseguir na linha seguinte, se precedido por \\
  – O primeiro comando deve conter o ficheiro final (programa)
  – Comentários iniciados por #

A ferramenta make

C - Reconstrução de ficheiros

- Comando unix na foma

  make

- Pesquisa em profundidade os ficheiros. Por omissão, começa no primeiro comando do `makefile`.
- Se algum ficheiro tiver data posterior ao ficheiro indicado no lado esquerdo de `:`, executa os comandos Unix indicados nas linhas seguintes até à linha em branco.

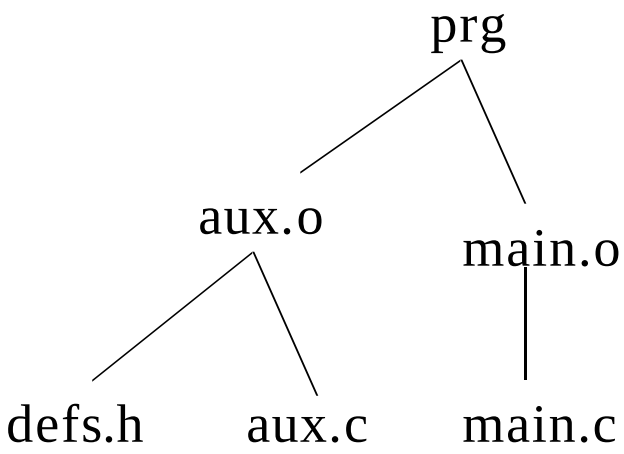
A ferramenta make

Exemplo 1: seja `prg` gerado a partir dos ficheiros `aux.o` e `main.o`, `aux.o` compilado a partir de `aux.c` (que contém directiva de inclusão de `defs.h`) e `main.o` compilado a partir de `main.c`

```
prg: aux.o main.o
   gcc -o prg aux.o main.o

aux.o: aux.c defs.h
   gcc -c aux.c

main.o: main.c
    gcc -c main.c

clean:
    rm *.o
```

A ferramenta make

A representação gráficas das dependências entre ficheiros é dada na seguinte árvore



Nota: Apesar de `defs.h` ser importado em `aux.c`, através da directiva `#include`, a dependência é estabelecida em `aux.o` Tal acontece porque `defs.h` é consultado apenas na geração do ficheiro objecto, durante a compilação.

A ferramenta make

O make é executado às 19:00 e os ficheiros têm as seguintes datas:

```
main.c@10:05    aux.c@10:20    defs.h@18:00
main.o@10:40    aux.o@10:45
prg@10:50
```

1 verificação da dependência temporária entre `prg` e `aux.o` é suspensa
   - `defs.h` é posterior a `aux.o`: logo, `aux.c` é compilado e `aux.o` passa a ter como última modificação as 19:00

2 dependência temporária entre `prg` e `main.o` é suspensa
   - `main.o` é posterior a `main.c`: logo, não há compilação

3 `prg` é posterior a `aux.o`: logo, é executada a edição de ligações e a última modificação de `prg` passa para as 19:01. A aplicação informática volta a ser coerente.

4 Para eliminar ficheiros desnecessários, basta executar `make clean`

A ferramenta make

D - Macros

Uma **macro**, é uma directiva de substituição, indicada na 1ª coluna no início do makefile

- Sintaxe:
  - Id = Substituição

A macro é referida por $(Id): os parêntesis podem ser eliminados, se Id for apenas 1 caractere.

Exemplo 2: o ficheiro makefile do exemplo 1 alterado para

```
CC = gcc  #  para alterar o compilador basta actualizar macro
prg: aux.o main.o
   $(CC)-o prg aux.o main.o
```

A ferramenta make

- As macros **dinâmicas** permitem alterar identificadores em cada regra.
- O ficheiro *makefile* torna-se menor (embora menos legível)

- Exemplos das macros dinâmicas mais usadas

$@	Nome do ficheiro alvo
$?	Lista das dependências de data posterior à do alvo
$<	Nome da dependência
$*	Nome do ficheiro alvo retirado o sufixo

A ferramenta make

- As macros dinâmicas aproveitam o facto da extensão do ficheiro indicar o seu tipo.

Exemplo 3: os ficheiros objecto e fonte diferem apenas na extensão. O make usa a seguinte regra implícita

```
CC=gcc
CFLAGS=
.c.o:
   $(CC) $(CFLAGS)-c $<  # ficheiro fonte tem o
                         # mesmo nome do alvo
```

A ferramenta make

- As regras implícitas são listadas num ficheiro de omissão (por exemplo, /usr/include/make/default.mk)
- As macros definidas no ficheiro `makefile` sobrepõem-se às do ficheiro de omissão.
- O make usa o seguinte algoritmo para reconstruir um ficheiro:
  1. Usa regra explícita, se existir.
  2. Se não existir, usa regra implícita, definida no makefile.
  3. Se não existir, usa regra implícita, definida no ficheiro de omissão.
  4. Se falhar, pára com mensagem de erro

A ferramenta make

Exemplo 4: o ficheiro makefile do exemplo 1 alterado para

```
CFLAGS=-Wall -pedantic -ansi
prg: aux.o main.o
   gcc -o prg aux.o main.o

aux.o: aux.c defs.h
    # reconstrução aux.o definida na macro
    # dinâmica

# dependências e reconstruição main.o definidas
# na macro dinâmica
```

Nota: na reconstrução de *aux.o* é necessário indicar dependências, porque os ficheiros de inclusão não são determinados na regra dinâmica.