I2010 - Langage C

Chap. 8:
La modularisation

La découpe en module

Module =

- une boîte à outil comme une interface java

et

- une classe qui implémente cette interface

Définir un module

- Un fichier entête (extension .h)

  ≈ une interface en Java

- Un fichier source (extension .c)

  ≈ une classe qui implémente cette interface

Les fichiers d'entête

Un **header** comprend toutes les informations **publiques** du module

- définitions de constantes
- définitions de types
- prototypes de fonctions
- spécifications des fonctions
- Attention: pas de code!

Les fichiers d'entête


```
#ifndef _PILE_H_
#define _PILE_H_

/* definition de constantes */
#define OK  1

/* definition de types */
typedef Pile

/* déclaration de fonctions*/
Pile init ();
int pop (Pile*);
void push (Pile*, int);

#endif   // _PILE_H_
```

Eviter la double inclusion


```
#ifndef _PILE_H_
#define _PILE_H_

...

#endif   // _PILE_H_
```

Les fichiers sources: inclusions


```
#include <stdio.h>
#include <stdlib.h>
#include "pile.h"
```

Les fichiers sources: implémentation


```
Pile init () {
...
}

int pop (Pile*) {
...
}

void push (Pile*, int) {
...
}
```

Application

La génération d'une application se fait en deux étapes

1) **compilation**
   produit des fichiers objets .o en compilant les différentes sources
2) **édition des liens**
   produit un exécutable en assemblant les fichiers objets .o

Makefile


```
CFLAGS = -std=c11 -pedantic -Wvla

npi : npi.o pile.o
   cc $(CFLAGS) -o npi npi.o pile.o

npi.o : npi.c pile.h
   cc $(CFLAGS) -c npi.c

pile.o : pile.c pile.h
   cc $(CFLAGS) -c pile.c

clean :
   rm *.o
   rm npi
```

Makefile


```
CFLAGS = -std=c11 -pedantic -Wvla

npi : npi.o pile.o
   cc $(CFLAGS) -o npi npi.o pile.o

npi.o : npi.c pile.h
   cc $(CFLAGS) -c npi.c

pile.o : pile.c pile.h
   cc $(CFLAGS) -c pile.c

clean :
   rm *.o
   rm npi
```

options de compilation

Makefile


```
CFLAGS = -std=c11 -pedantic -Wvla

npi : npi.o pile.o
   cc $(CFLAGS) -o npi npi.o pile.o

npi.o : npi.c pile.h
   cc $(CFLAGS) -c npi.c

pile.o : pile.c pile.h
   cc $(CFLAGS) -c pile.c

clean :
   rm *.o
   rm npi
```

règle d'édition de liens

Makefile


```
CFLAGS = -std=c11 -pedantic -Wvla

npi : npi.o pile.o
   cc $(CFLAGS) -o npi npi.o pile.o

npi.o : npi.c pile.h
   cc $(CFLAGS) -c npi.c

pile.o : pile.c pile.h
   cc $(CFLAGS) -c pile.c

clean :
   rm *.o
   rm npi
```

règles de compilation

Makefile


```
CFLAGS = -std=c11 -pedantic -Wvla

npi : npi.o pile.o
   cc $(CFLAGS) -o npi npi.o pile.o

npi.o : npi.c pile.h
   cc $(CFLAGS) -c npi.c

pile.o : pile.c pile.h
   cc $(CFLAGS) -c pile.c
```

```
clean :
   rm *.o
   rm npi
```

règle utilitaire

Makefile

Liste de règles qui ont

- une condition de dépendance

```
npi : npi.o pile.o
```

- une action

```
cc $(CFLAGS) -o npi npi.o pile.o
```

make

- **`make`**

  MÀJ de **`npi`**

- **`make npi.o`**

  MÀJ de **`npi.o`**

- **`make clean`**

  provoque l'effacement des modules objets et de l'exécutable