# F02a - Organisation av C-kod

## 5DV149 Datastrukturer och algoritmer

Niclas Börlin
niclas.borlin@cs.umu.se

2020-01-22 Wed

# Funktionsdeklarationer och definitioner (1)

- En funktions-*deklaration* berättar för kompilator att en funktion *finns*.
- Den berättar vad funktionen har för:
  - Namn.
  - Typ på returvärde.
  - Parametrar (antal och typ men ej namn).
- Exempel:
  - `list_next` är en funktion som tar två parametrar:
    - den första parametern är av typen `const list *`.
    - den andra parametern är av typen `const list_pos`.
  - `list_next` returnerar ett värde av typen `list_pos`.
  - Notera det **avslutande semikolonet!**

code/list.h

```
80  list_pos list_next(const list *l, const list_pos p);
```

# Funktionsdeklarationer och definitioner (2)

- En funktions-*definition* berättar vad funktionen *gör*.
- Den innehåller också funktionens kropp.

code/list.c

```
114 list_pos list_next(const list * l, const list_pos p)
115 {
116     if ( p == list_end(l) ) {
117         // This should really throw an error.
118         fprintf(stderr,"list_next: Warning: Trying to navigate "
119             "past end of list!");
120     }
121     return p->next;
122 }
123
```

# Funktionsdeklarationer och definitioner (3)

- Funktionsdeklarationer får upprepas.
  - Måste komma före användande av (anrop till) funktionen.
- Funktionsdefinitionen får bara förekomma en gång.

# Organisation av C-kod (1)

- I C-kod är det vanligt att definitioner av funktioner som hör ihop samlas i en kodfil, t.ex. `list.c`. Vi kan kalla det för en modul.
- Dessutom skapas en s.k. *header*-fil (.h-fil) där varje publik funktion deklareras, t.ex. `list.h`.
- .h-filerna kan sägas innehåller funktionernas *gränssnitt*.

# Organisation av C-kod (2)

- Ett program som ska använda modulens funktioner gör `#include "list.h"` eller `#include <list.h>`.
- När kompilatorn hittar anrop till funktionerna som deklarerats i `list.h` så kan den kontrollera att parameternamn och -antal samt returtyp är korrekt.
- Dessutom genererar kompilatorn anropskod till funktionen.
- Filen som innehåller funktionsdefinitionen bör också innehålla `#include "list.h"` för att ha bl.a. kunna upptäcka om definitionen eller deklarationen förändrats.
- I ett senare skede av kompileringen så ”länkas” den kompilerade koden av funktionen in (den exakta adressen i minnet på funktionen läggs till).

# Header-filer (.h-filer) (1)

- Välskrivna headerfiler innehåller förutom *deklarationerna* av funktioner också en hjälptext som förklarar den användaren behöver veta för att kunna använda funktionen på rätt sätt.

code/list.h

```
72 /**
73  * list_next() - Return the next position in a list.
74  * @l: List to inspect.
75  * @p: Any valid position except the last in the list.
76  *
77  * Returns: The position in the list after the given position.
78  *          NOTE: The return value is undefined for the last position.
79  */
80 list_pos list_next(const list *l, const list_pos p);
```

- Dessutom innehåller .h-filerna vanligen definitioner av publika typer och konstanter som hör ihop med modulen.

code/list.h

```
27 // ==========PUBLIC DATA TYPES============
28
29 // List type.
30 typedef struct list list;
31
32 // List position type.
33 typedef struct cell *list_pos;
```

# Header-filer (.h-filer) (2)

- För att undvika att samma definitioner inkluderas flera gånger innehåller .h-filer vanligen `#ifndef`-direktiv (*if not defined*):

code/list.h

```
1  #ifndef __LIST_H
2  #define __LIST_H
```

code/list.h

```
153 #endif
```

# Kod-filer (.c-filer)

- C-filer som ska använda funktionerna inleder filen med ett antal include-direktiv som gör funktionernas deklarationer och andra definitioner tillgängliga när resten av filen kompileras.

code/list_mwe1.c

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include "list.h"
```

- Ett include-direktiv
  - på formen `#include <list.h>` söker efter .h-filen på standardställen.
  - på formen `#include "list.h"` söker dessutom i aktuell katalog.
- Vanligen använder man den senare versionen bara på kod man skrivit själv och som ligger i samma katalog som C-filen.

# Mappstruktur

- Det är vanligt att strukturera koden i mappar på följande sätt:
  - `/include` innehåller .h-filerna.
  - `/lib` innehåller biblioteksfiler.
  - `/bin` kompilerade filer läggs hit.
  - `/src` toppkatalog för källkoden
    - `/list` kod för list-modulen
    - `/dlist` kod för dlist-modulen
    - `/stack` kod för stack-modulen
    - ...
- För små projekt kan all kod samlas i en katalog `/src`.

# Kompilering

- Vid kompilering med `gcc` så måste alla C-filer som ska kompileras anges.
- För header-filerna ska en sökväg anges. Då används flaggan `-I`.

```
gcc ... -I../path/to/include/files c-file1.c c-file2.c ...
```

- Exempel:

```
cd .../src/list
gcc -Wall -o list_mwe1 -I../../include/ list_mwe1.c list.c
```