Lezione 14: Librerie e compilazione separata

Laboratorio di Elementi di Architettura e Sistemi Operativi

8 Maggio 2013

Soluzione dell'esercizio della lezione scorsa

```
#include <stdio.h>
#include <stdlib.h>
#include "liste.c"

void printlist(lista_t lista) {
  lista_t curr;
  for(curr = lista; curr != NULL; curr = curr->next) {
    printf("%d\t",curr->key);
  }
  printf("\n");
}

int main() {
  lista_t lista = NULL;
  lista_t curr;
  int i,n;

  printf("Inserire gli elementi della lista.\n");
  printf("Inserire un numero negativo per terminare.\n");
  scanf("%d", &n);
  while(n >= 0) {
    lista = insert(lista, n);
    scanf("%d", &n);
  }

  printf("La lista e' lunga %d elementi.\n",
         length(lista));
  printlist(lista);

  printf("Inserire il numero di elementi da eliminare: ");
  scanf("%d", &n);

  for(i = 0; i < n; i++) {
    lista = delete(lista);
  }

  printf("La lista e' lunga %d elementi.\n",
         length(lista));
  printlist(lista);

  return 0;
}
```

Librerie e compilazione separata

Suddivisione del codice in più file

- La maggior parte dei programmi che abbiamo visto finora erano composti da un unico file `.c`
- Nel caso di programmi più grandi è conveniente *suddividere il codice in più file*:
  - migliore suddivisione e organizzazione del codice
  - facilità di manutenzione e correzione degli errori
  - possibilità di riutilizzare il codice per progetti diversi
- Il linguaggio C e l'ambiente Unix forniscono *tre strade* per suddividere il codice:
  1. inclusione diretta di file sorgente e *compilazione unica*;
  2. inclusione di header file e *compilazione separata*;
  3. *creazione di librerie* statiche o dinamiche.

Inclusione diretta di file sorgente

- Effettuata dal preprocessore mediante la direttiva

```
#include "mylib.c"
```

- Prima della compilazione il preprocessore inserisce il codice contenuto in `mylib.c` nel programma
- La compilazione si fa normalmente richiamando il compilatore sul file contenente il main:

```
gcc main.c -o main
```

- È il metodo più semplice ma meno efficiente:
  - la modifica di uno solo dei file richiede la ricompilazione completa di tutto il programma.

```c
// mylib.c

void aggiungi(int *somma, int n) {
  *somma += n;
}
```

```c
// main.c
#include <stdio.h>
#include "mylib.c"

int main() {
  int i, n, somma = 0;

  for(i=1; i <= 3; i++) {
    scanf("%d", &n);
    aggiungi(&somma, n);
  }
  printf("TOTALE: %d\n", somma);
}
```

```
davide@ubuntu-virtual: ~/Sources/le
File  Edit  View  Search  Terminal  Help
davide@ubuntu-virtual:~/Sources/leso$ gcc -o main main.c
davide@ubuntu-virtual:~/Sources/leso$ ./main
1
2
3
TOTALE: 6
davide@ubuntu-virtual:~/Sources/leso$
```

Header file e compilazione separata

- Anziché includere tutto il sorgente si può includere un *header file*:
  - è un file con estensione `.h`
  - contiene solamente:
    - inclusioni di altri header file
    - definizione di strutture e di tipi (`typedef`)
    - prototipi di funzioni
  - si usa la direttiva `#include "mylib.h"`
- Il codice delle funzioni va scritto in un file `.c` separato:
  - che deve includere l'header!
- Compilazione e linking devono essere fatti separatamente:

```
gcc -c mylib.c
gcc -c main.c
gcc -o main main.o mylib.o
```

- La modifica di uno dei file richiede di ricompilare il file oggetto corrispondente e di rifare il linking.

```
// mylib.h

void aggiungi(int *somma, int n);
```

```
// mylib.c
#include "mylib.h"

void aggiungi(int *somma, int n) {
  *somma += n;
}
```

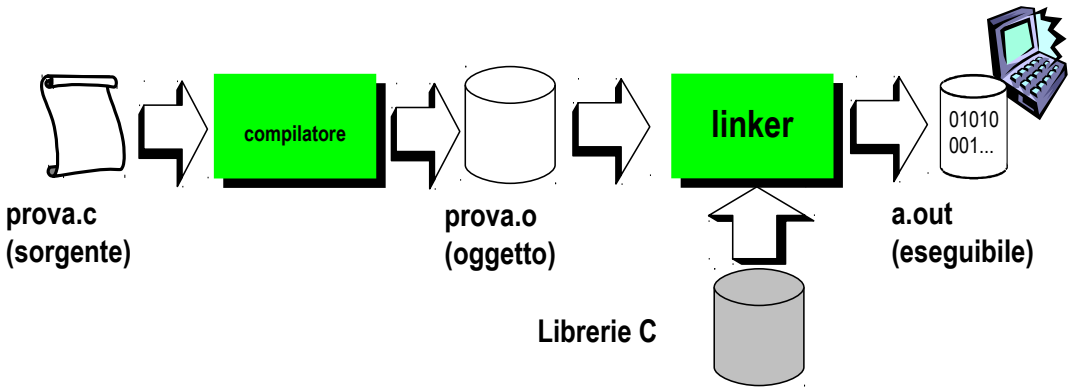
```
// main.c
#include <stdio.h>
#include "mylib.h"

int main() {
  int i, n, somma = 0;

  for(i=1; i <= 3; i++) {
    scanf("%d", &n);
    aggiungi(&somma, n);
  }
  printf("TOTALE: %d\n", somma);
}
```

```
File  Edit  View  Search  Terminal  Help
davide@ubuntu-virtual:~/Sources/leso$ gcc -c mylib.c
davide@ubuntu-virtual:~/Sources/leso$ gcc -c main.c
davide@ubuntu-virtual:~/Sources/leso$ gcc -o main main.o mylib.o
davide@ubuntu-virtual:~/Sources/leso$ ./main
1
2
3
TOTALE: 6
davide@ubuntu-virtual:~/Sources/leso$
```

Le librerie

Le librerie

- La fase di Linking è l'ultimo passaggio nella creazione di un eseguibile:
  - aggiunge ai file oggetto il codice delle librerie esterne necessarie per il programma



- Il linking può essere di due tipi:
  - **Linking statico:** l'eseguibile contiene una copia del codice delle librerie;
  - **Linking dinamico:** l'eseguibile non contiene il codice, ma solo un riferimento al file della libreria.
- Se l'eseguibile è stato "linkato dinamicamente", le librerie richieste devono essere presenti ed "installate" nel sistema per poterlo eseguire.
- Se l'eseguibile è "linkato staticamente", la presenza delle librerie non è necessaria.

Linking statico vs. linking dinamico

Linking statico

- **Vantaggi:**
  - non necessita della presenza delle librerie
  - è più veloce nel caricamento del programma
- **Svantaggi:**
  - maggior consumo di memoria: se più programmi usano la stessa libreria, essa viene caricata in memoria più volte

Linking dinamico

- **Vantaggi:**
  - minor consumo di memoria: se più programmi usano la stessa libreria, essa viene caricata in memoria una volta sola.
- **Svantaggi:**
  - necessita della presenza delle librerie nel sistema
  - caricamento del programma più lento

Creare una libreria statica

- Una libreria statica ha un nome che inizia con `lib` e termina con l'estensione `.a`
- Il sorgente va suddiviso in un header ed un file con il codice
- Dopo aver creato il file oggetto del codice si usa il comando `ar` per creare la libreria:

  ```
  ar r libmylib.a mylib.o
  ```

- Per fare il linking si usano le opzioni `-l` e `-L` del `gcc`:

  `-lmylib` fa il link con la libreria `libmylib.a`

  `-Ldir` cerca le librerie in `dir` oltre che tra quelle di sistema

  ```
  gcc -o main main.o -L. -lmylib
  ```

```
// mylib.h

void aggiungi(int *somma, int n);
```

```
// mylib.c
#include "mylib.h"

void aggiungi(int *somma, int n) {
  *somma += n;
}
```

```
// main.c
#include <stdio.h>
#include "mylib.h"

int main() {
  int i, n, somma = 0;

  for(i=1; i <= 3; i++) {
    scanf("%d", &n);
    aggiungi(&somma, n);
  }
  printf("TOTALE: %d\n", somma);
}
```

```
File  Edit  View  Search  Terminal  Help
davide@ubuntu-virtual:~/Sources/leso$ gcc -c mylib.c
davide@ubuntu-virtual:~/Sources/leso$ ar r libmylib.a mylib.o
ar: creating libmylib.a
davide@ubuntu-virtual:~/Sources/leso$ gcc -c main.c
davide@ubuntu-virtual:~/Sources/leso$ gcc -o main main.o -L. -lmylib
davide@ubuntu-virtual:~/Sources/leso$ ./main
1
2
3
TOTALE: 6
davide@ubuntu-virtual:~/Sources/leso$
```

Creare una libreria dinamica

- Una libreria statica ha un nome che inizia con `lib` e termina con l'estensione `.so`
- Il sorgente va suddiviso in un header ed un file con il codice
- Dopo aver creato il file oggetto del codice si usa l'opzione `-shared` di `gcc` per creare la libreria:

  ```
  gcc -shared -o libmylib.so mylib.o
  ```

- Per fare il linking si usano le opzioni `-l` e `-L` del `gcc`:

  ```
  gcc -o main main.o -L. -lmylib
  ```

- Se una libreria è presente sia in versione statica che diamica, il compliatore *sceglie la versione dinamica*
  - per forzarlo ad usare la versione statica si usa l'opzione `-static`:

    ```
    gcc -static -o main main.o -L. -lmylib
    ```

```c
// mylib.h

void aggiungi(int *somma, int n);
```

```c
// mylib.c
#include "mylib.h"

void aggiungi(int *somma, int n) {
  *somma += n;
}
```

```c
// main.c
#include <stdio.h>
#include "mylib.h"

int main() {
  int i, n, somma = 0;

  for(i=1; i <= 3; i++) {
    scanf("%d", &n);
    aggiungi(&somma, n);
  }
  printf("TOTALE: %d\n", somma);
}
```

```
File  Edit  View  Search  Terminal  Help
davide@ubuntu-virtual:~/Sources/leso$ gcc -c mylib.c
davide@ubuntu-virtual:~/Sources/leso$ gcc -shared -o libmylib.so mylib.o
davide@ubuntu-virtual:~/Sources/leso$ gcc -c main.c
davide@ubuntu-virtual:~/Sources/leso$ gcc -o main main.o -L. -lmylib
davide@ubuntu-virtual:~/Sources/leso$ ./main
./main: error while loading shared libraries: libmylib.so: cannot open sha
red object file: No such file or directory
davide@ubuntu-virtual:~/Sources/leso$
```

Installare le librerie dinamiche

- Il comando `ldd` elenca le librerie dinamiche richieste da un eseguibile:

  `ldd main`

- Per specificare dove si trovano le librerie esistono due sistemi di configurazione:
  - Modifcare il file `/etc/ld.so.conf`
    - specifica dove si trovano le librerie di sistema
    - modificabile solamente dall'amministratore (`root`)
  - Usare la variabile d'ambiente `LD_LIBRARY_PATH`
    - specifica dove si trovano le librerie dell'utente
    - modificabile da tutti usando `export`

```
davide@ubuntu-virtual: ~/Sources/leso
File  Edit  View  Search  Terminal  Help
davide@ubuntu-virtual:~/Sources/leso$ ldd main
        linux-gate.so.1 =>  (0x008ab000)
        libmylib.so => not found
        libc.so.6 => /lib/libc.so.6 (0x005c7000)
        /lib/ld-linux.so.2 (0x0032b000)
davide@ubuntu-virtual:~/Sources/leso$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
davide@ubuntu-virtual:~/Sources/leso$ ldd main
        linux-gate.so.1 =>  (0x00783000)
        libmylib.so (0x00909000)
        libc.so.6 => /lib/libc.so.6 (0x00e33000)
        /lib/ld-linux.so.2 (0x0063c000)
davide@ubuntu-virtual:~/Sources/leso$ ./main
1
2
3
TOTALE: 6
davide@ubuntu-virtual:~/Sources/leso$
```