Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni del Compitino dell'8 Maggio 2013

Esercizio 1.

1. *Suddividere il codice per la gestione delle liste visto nella lezione scorsa in un file di header* `liste.h` *che contenga le dichiarazioni ed un file* `liste.c` *con il codice.*

   Il file `liste.h` contiene:

```
// Definizione dei tipi

typedef struct elem {
  int key;
  struct elem *next;
} elemen_t;

typedef elemen_t* lista_t;

// Prototipi delle Funzioni

int is_empty(lista_t lista);
int length(lista_t lista);
lista_t insert(lista_t lista, int key);
int head(lista_t lista);
lista_t delete(lista_t lista);
```

   Il file `liste.c` include `liste.h` e contiene solamente il codice delle funzioni.

2. *Riscrivere l'esercizio della lezione scorsa in modo che che includa l'header di gestione delle liste mediante la direttiva* `#include "liste.h"`.

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

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;
  int i,n;

  printf("Inserire gli elementi della lista.\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;
}
```

3. *Compilare separatamente il file `liste.c` ed il file con il codice del programma, generando due file oggetto.* `gcc -c liste.c; gcc -c main.c`
4. *Fare il link dei due file oggetto per creare l'eseguibile e verificarne il funzionamento.*
   `gcc -o exlist-sep main.o liste.o`
5. *Creare una libreria statica `libliste.a` che contenga le funzioni di gestione delle liste.*
   `ar r libliste.a liste.o`
6. *Fare il link del file oggetto del programma con la libreria statica `libliste.a` e verificare il funzionamento dell'eseguibile.*
   `gcc -o exlist-static main.o -L. -lliste`
7. *Creare una libreria dinamica `libliste.so` che contenga le funzioni di gestione delle liste.*
   `gcc -shared -o libliste.so liste.o`
8. *Fare il link del file oggetto del programma con la libreria dinamica `libliste.so` e verificare il funzionamento dell'eseguibile.*
   `gcc -o exlist-shared main.o -L. -lliste`
   Il comando `./main` genera il seguente errore:
   `error while loading shared libraries: libliste.so: cannot open shared object`
   Per eseguire correttamente il programma è necessario installare la libreria `libliste.so` modificando il valore della variabile d'ambiente `LD_LIBRARY_PATH`:
   ```
   $ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
   $ ./exlist-shared
   ```
9. *Verificare le dipendenze delle librerie condivise di ognuno degli eseguibili generati ai punti 4, 6, e 8, usando il comando `ldd`.*
   ```
   $ ldd exlist-sep
   linux-gate.so.1 =>  (0x0081b000)
   libc.so.6 => /lib/libc.so.6 (0x001e0000)
   /lib/ld-linux.so.2 (0x001ac000)
   $ ldd exlist-static
   linux-gate.so.1 =>  (0x0012f000)
   libc.so.6 => /lib/libc.so.6 (0x00820000)
   /lib/ld-linux.so.2 (0x00503000)
   $ ldd exlist-shared
   linux-gate.so.1 =>  (0x0046d000)
   libliste.so (0x00edd000)
   libc.so.6 => /lib/libc.so.6 (0x006cf000)
   /lib/ld-linux.so.2 (0x0062a000)
   ```
   Tutti gli eseguibili dipendono dalle stesse librerie di sistema. L'eseguibile linkato dinamicamente dipende anche dalla libreria `libliste.so`.
10. *Confrontare le dimensioni dei file eseguibili generati ai punti 4, 6, e 8. Qual'è quello più grande? E qual'è quello più piccolo?*

```
$  ls -la  exlist-*
-rwxr-xr-x 1 davide davide 7506 2012-05-23 09:55 exlist-sep
-rwxr-xr-x 1 davide davide 7192 2012-05-23 09:56 exlist-shared
-rwxr-xr-x 1 davide davide 7506 2012-05-23 09:55 exlist-static
```

Il file eseguibile più grossi sono quelli generati ai punti 4 e 6, mentre quello più piccolo è quello generato al punto 8 (libreria dinamica).

Esercizio 2.

1. *Modificare la libreria di gestione delle liste come segue:*
   - *aggiungere le funzioni* `int max(lista_t lista)` *e* `int min(lista_t lista)` *che restituiscono il valore minimo e massimo contenuto nella lista. Se la lista è vuota le funzioni ritornano il valore* `-1`*;*

```
int min(lista_t lista) {
  int res;
  if(lista == NULL) {
    return -1;
  }
  res = lista->key;
  lista = lista->next;
  while(lista != NULL) {
    if(lista->key < res) {
      res = lista->key;
    }
    lista = lista->next;
  }
  return res;
}

int max(lista_t lista) {
  int res;
  if(lista == NULL) {
    return -1;
  }
  res = lista->key;
  lista = lista->next;
  while(lista != NULL) {
    if(lista->key > res) {
      res = lista->key;
    }
    lista = lista->next;
  }
  return res;
}
```

   - *aggiungere la funzione* `lista_t delete_key(lista_t lista, int key)` *che cerchi il primo elemento della lista con chiave* `key` *e, se esiste, lo elimini dalla lista. La funzione ritorna la lista con l'elemento cancellato.*

```
lista_t delete_key(lista_t lista, int key) {
  lista_t curr, prec;
  prec = NULL;
  for(curr = lista; curr != NULL; curr = curr->next) {
    if(curr->key == key) {
      curr = delete(curr);
      if(prec == NULL) { // ho cancellato il primo elemento
        return curr;
      } else { // ho cancellato un elemento interno
```

```
            prec->next = curr;
            return lista;
          }
        }
        prec = curr;
      }
      return lista;
    }
```

2. *Ricreare la libreria statica* `libliste.a` *e la libreria dinamica* `libliste.so`

```
gcc -c liste.c
ar r libliste.a liste.o
gcc -shared -o libliste.so liste.o
```

3. *Scrivere un programma che usi la libreria per fare le seguenti operazioni:*
   - *leggere da tastiera una lista di interi non negativi;*
   - *eliminare dalla lista il valore minimo ed il valore massimo;*
   - *stampare la lista senza il minimo ed il massimo;*
   - *calcolare la media dei valori rimasti e stamparla a schermo.*

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

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

double media(lista_t lista) {
  lista_t curr;
  double sum = 0.0;
  int n = 0;
  if(lista == NULL) return 0.0;
  for(curr = lista; curr != NULL; curr = curr->next) {
    sum += curr->key;
    n++;
  }
  return sum/n;
}

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

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

  lista = delete_key(lista,min(lista));
  lista = delete_key(lista,max(lista));
```

```
  printlist(lista);
  printf("Il valor medio della lista e': %lf\n", media(lista));

  return 0;
}
```

4. *Generare un eseguibile linkato staticamente ed uno linkato dinamicamente del programma, verificarne il funzionamento e confrontarne le dimensioni e le dipendenze dalle librerie condivise.*

   `gcc -c main.c`
   `gcc -o minliste-shared main.o -L. -lliste`
   `gcc -static -o minliste-static main.o -L. -lliste`

   L'eseguibile di dimensione maggiore è quello linkato staticamente. In questo caso particolare, poiché si è usata l'opzione `-static` per forzare il link statico nella generazione di `minliste-static`, anche le librerie di sistema sono state linkate staticamente. Quindi l'eseguibile `minliste-static` non ha dipendenze da librerie dinamiche, e l'esecuzione del comando `ldd minliste-static` ritorna il messaggio `minliste-static non e' un eseguibile dinamico`.