Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni degli esercizi del 16 Maggio 2012

Esercizio 1.

1. *Creare un file* `liste.c` *che contenga il codice per la gestione delle liste visto nella lezione scorsa.*

```
// liste.c

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

typedef elemen_t* lista_t;
int is_empty(lista_t lista) {
  return(lista == NULL);
}

int length(lista_t lista) {
  int n = 0;
  while(lista != NULL) {
    n++;
    lista = lista->next;
  }
  return n;
}

lista_t insert(lista_t lista, int key) {
  elemen_t *paux;
  paux = (elemen_t *)malloc(sizeof(elemen_t));
  paux->key = key;
  paux->next = lista;
  return paux;
}

int head(lista_t lista) {
  if(lista != NULL) {
    return lista->key;
  }
  return 0;
}

void delete(lista_t *plista) {
  elemen_t *paux;
  if(*plista != NULL) {
    paux = *plista;
    *plista = (*plista)->next;
    free(paux);
  }
}

void print(lista_t lista) {
    printf("Lista: ");
    while(lista != NULL) {
      printf("%d\t", lista->key);
      lista = lista->next;
```

```
    }
    printf("\n");
}

int min(lista_t lista) {
  int min;
  if(lista == NULL) {
    fprintf(stderr, "ERRORE: la lista è vuota!\n");
    return 0;
  }
  min = lista->key;
  while(lista != NULL) {
    if(lista->key < min) {
      min = lista->key;
    }
    lista = lista->next;
  }
  return min;
}
```

2. *Scrivere un programma che includa il codice di gestione delle liste mediante la direttiva* `#include liste.c` *e che esegua le seguenti operazioni:*
   - *legga da tastiera una lista di* n *interi.* n *non è noto a priori e introdotto da tastiera;*
   - *cerchi il minimo valore presente nella lista lo stampi a schermo.*

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

#include "liste.c"

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

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

    for(i = 0; i < n; i++) {
      printf("Elemento %d: ", i+1);
      scanf("%d", &k);
      lista = insert(lista, k);
    }

    lmin = min(lista);
    printf("Il minimo è %d\n", lmin);
}
```

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

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

typedef struct elem {
```

```
  int key;
  struct elem *next;
} elemen_t;

typedef elemen_t* lista_t;

int is_empty(lista_t lista);

int length(lista_t lista);

lista_t insert(lista_t lista, int key);

int head(lista_t lista);

void delete(lista_t *plista);

void print(lista_t lista);

int min(lista_t lista);
```

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

int is_empty(lista_t lista) {
  return(lista == NULL);
}
...
```

4. *Riscrivere il programma del punto 2 includendo l'header file invece di* `liste.c`.

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

#include "liste.h"

main()
{
  ...
}
```

5. *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`

6. *Fare il link dei due file oggetto per creare l'eseguibile e verificarne il funzionamento.*

   `gcc -o main main.o liste.o`

   Con il comando `./main` il programma viene eseguito correttamente:

```
$ ./main
Inserire il numero di elementi: 3
Elemento 1: 2
Elemento 2: -3
Elemento 3: 4
Il minimo è -3
```

7. *Creare una libreria statica* `libliste.a` *che contenga le funzioni di gestione delle liste.*

   ```
   ar r libliste.a liste.o
   ```

8. *Fare il link del file oggetto del programma con la libreria statica* `libliste.a` *e verificare il funzionamento dell'eseguibile.*

   ```
   gcc -o main main.o -lliste -L.
   ```

   Con il comando `./main` il programma viene eseguito correttamente:

9. *Creare una libreria dinamica* `libliste.so` *che contenga le funzioni di gestione delle liste.*

   ```
   gcc -shared -o libliste.so liste.o
   ```

10. *Fare il link del file oggetto del programma con la libreria dinamica* `libliste.so` *e verificare il funzionamento dell'eseguibile.*

    ```
    gcc -o main main.o -lliste -L.
    ```

    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:.
    $ ./main
    Inserire il numero di elementi: 3
    Elemento 1: 2
    Elemento 2: -3
    Elemento 3: 4
    Il minimo è -3
    ```

11. *Verificare le dipendenze delle librerie condivise di ognuno degli eseguibili generati ai punti 2, 6, 8 e 10, usando il comando* `ldd`*.*

    ```
    $ ldd minlist-dir
    linux-gate.so.1 =>  (0x00922000)
    libc.so.6 => /lib/libc.so.6 (0x00a82000)
    /lib/ld-linux.so.2 (0x002da000)
    $ ldd minlist-sep
    linux-gate.so.1 =>  (0x0081b000)
    libc.so.6 => /lib/libc.so.6 (0x001e0000)
    /lib/ld-linux.so.2 (0x001ac000)
    $ ldd minlist-static
    linux-gate.so.1 =>  (0x0012f000)
    libc.so.6 => /lib/libc.so.6 (0x00820000)
    /lib/ld-linux.so.2 (0x00503000)
    $ ldd minlist-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.c`.

12. *Confrontare le dimensioni dei file eseguibili generati ai punti 2, 6, 8 e 10. Qual'è quello più grande? E qual'è quello più piccolo?*

```
$  ls -la  minlist-*
-rwxr-xr-x 1 davide davide 7483 2012-05-23 09:54 minlist-dir
-rwxr-xr-x 1 davide davide 7506 2012-05-23 09:55 minlist-sep
-rwxr-xr-x 1 davide davide 7192 2012-05-23 09:56 minlist-shared
-rwxr-xr-x 1 davide davide 7506 2012-05-23 09:55 minlist-static
```

Il file eseguibile più grossi sono quelli generati ai punti 6 e 8, mentre quello più piccolo è quello generato al punto 10 (libreria statica).

Esercizio 2.

1. *Modificare la libreria di gestione delle liste come segue:*
   - *aggiungere un puntatore* `prev` *all'elemento precedente nella lista nella struttura* `elemen_t`*;*
   - *scrivere 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 scrivono un messaggio d'errore su* `stderr` *e ritornano il valore* `0`*;*
   - *scrivere una funzione* `void delete_key(lista_t *plista, int key)` *che cerchi il primo elemento della lista con chiave* `key` *e, se esiste, lo elimini dalla lista.*

```
// listedoppie.h
#include <stdio.h>
#include <stdlib.h>

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

typedef elemen_t* lista_t;

int is_empty(lista_t lista);

int length(lista_t lista);

lista_t insert(lista_t lista, int key);

int head(lista_t lista);

void delete(lista_t *plista);

void delete_key(lista_t *plista, int key);

int min(lista_t lista);

int max(lista_t lista);

void print(lista_t lista);

float media(lista_t lista);


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

```
int is_empty(lista_t lista) {
  return(lista == NULL);
}

int length(lista_t lista) {
  int n = 0;
  while(lista != NULL) {
    n++;
    lista = lista->next;
  }
  return n;
}

int min(lista_t lista) {
  int min;
  if(lista == NULL) {
    fprintf(stderr, "ERRORE: la lista è vuota!\n");
    return 0;
  }
  min = lista->key;
  while(lista != NULL) {
    if(lista->key < min) {
      min = lista->key;
    }
    lista = lista->next;
  }
  return min;
}

int max(lista_t lista) {
  int max;
  if(lista == NULL) {
    fprintf(stderr, "ERRORE: la lista è vuota!\n");
    return 0;
  }
  max = lista->key;
  while(lista != NULL) {
    if(lista->key > max) {
      max = lista->key;
    }
    lista = lista->next;
  }
  return max;
}

lista_t insert(lista_t lista, int key) {
  elemen_t *paux;
  paux = (elemen_t *)malloc(sizeof(elemen_t));
  paux->key = key;
  paux->next = lista;
  paux->prev = NULL;
  if(lista != NULL) {
    lista->prev = paux;
  }
  return paux;
}

int head(lista_t lista) {
  if(lista != NULL) {
```

```
    return lista->key;
  }
  return 0;
}

void delete(lista_t *plista) {
  elemen_t *paux, *pprev;
  if(*plista != NULL) {
    paux = *plista;
    pprev = (*plista)->prev;
    *plista = (*plista)->next;
    if(*plista != NULL) {
      (*plista)->prev = pprev;
    }
    if(pprev != NULL) {
      pprev->next = *plista;
    }
    free(paux);
  }
}

elemen_t *search(lista_t lista, int key) {
  while(lista != NULL) {
    if(lista->key == key) {
      return lista;
    }
    lista = lista->next;
  }
  return lista;
}

void delete_key(lista_t *plista, int key) {
  elemen_t *paux;
  paux = search(*plista, key);
  if(paux == *plista) {
    delete(plista);
  } else {
    delete(&paux);
  }
}

void print(lista_t lista) {
    printf("Lista: ");
    while(lista != NULL) {
      printf("%d\t", lista->key);
      lista = lista->next;
    }
    printf("\n");
}

float media(lista_t lista) {
  float somma = 0.0;
  int n = 0;
  while(lista != NULL) {
    n++;
    somma += lista->key;
    lista = lista->next;
  }
  if(n > 0) {
    return somma/n;
```

```
  } else {
    return 0.0;
  }
}
```

2. *Creare una libreria statica* `liblistedoppie.a` *e una libreria dinamica* `liblistedoppie.so`

```
gcc -c listedoppie.c
ar r liblistedoppie.a listedoppie.o
gcc -shared -o liblistedoppie.so listedoppie.o
```

3. *Scrivere un programma che usi la libreria per fare le seguenti operazioni:*
   - *leggere da tastiera una lista di n interi. n non è noto a priori e introdotto da tastiera;*
   - *elimini dalla lista il valore minimo ed il valore massimo;*
   - *calcoli la media dei valori rimasti e la stampi a schermo.*

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

#include "listedoppie.h"

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

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

    for(i = 0; i < n; i++) {
      printf("Elemento %d: ", i+1);
      scanf("%d", &k);
      lista = insert(lista, k);
    }

    lmin=min(lista);
    lmax=max(lista);

    delete_key(&lista, lmin);
    delete_key(&lista, lmax);

    printf("La media dei valori è: %lf\n",media(lista));
}
```

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 main-shared main.o -L. -llistedoppie
gcc -static -o main-static main.o -L. -llistedoppie
```

Anche in questo caso, l'eseguibile linkato staticamente funziona senza necessità di installare le librerie, mentre quello linkato dinamicamente richiede la modifica di `LD_LIBRARY_PATH`.

```
$ ls -la main-s*
-rwxr-xr-x 1 davide davide   7228 2012-05-23 10:07 main-shared
-rwxr-xr-x 1 davide davide 616430 2012-05-23 10:07 main-static


$ ldd main-static
not a dynamic executable
$ ldd main-shared
linux-gate.so.1 =>  (0x00a8e000)
liblistedoppie.so (0x007d3000)
libc.so.6 => /lib/libc.so.6 (0x00c62000)
/lib/ld-linux.so.2 (0x006c9000)
```

In questo caso l'eseguibile linkato staticamente *non dipende da nessuna libreria dinamica*: l'opzione `-static` del `gcc` fa si che anche il codice delle librerie di sistema (come `libc.so.6`) venga copiato all'interno dell'eseguibile. Si noti come in questo caso `main-static` sia molto più grande di `main-shared`.