Politecnico di Milano - Dipartimento di Elettronica e informazione

Prof. Mauro Negri

Fondamenti di Informatica **15 luglio 2013**
II appello

Matricola_____ Cognome _____ Nome_____

Durata prova: 2 ore

Istruzioni

LE RISPOSTE DEVONO ESSERE SCRITTE SU QUESTO PLICO (anche sul retro dei fogli)
Non separare questi fogli. Si può utilizzare la matita.
L'uso di cellulari, libri, eserciziari, appunti o calcolatrici durante lo svolgimento della prova comporta l'annullamento della prova.
Non è possibile uscire dall'aula durante la prova tranne in casi eccezionali.

Esercizio 1 (12 punti) _____

Esercizio 2 (6 punti) _____

Esercizio 3 (6 punti) _____

Esercizio 4 (6 punti) _____

Punteggio totale (30 punti) _____

Esercizio 1.
Scrivere un programma C che esegue le seguenti operazioni in ordine:

1. Legge un file binario chiamato programmi.ttt record per record, dove ogni record ha la seguente struttura:
   struct el {char nome[20]; char p1[10]; char p2[10]; char p3[10]}
   Si supponga che ogni stringa sia correttamente memorizzata ("\0" incluso nella dimensione prevista) e che non si conosca la dimensione del file.
   Nome rappresenta il pathname completo di un programma eseguibile, p1, p2 e p3 i parametri del programma; si noti che tutti i programmi al massimo hanno due parametri, ma possono averne di meno (anche non averli). La stringa utilizzata per indicare la fine dei parametri utili è la stringa "null" e sarà memorizzata nel parametro successivo a quello utile, pertanto se un programma ha un parametro il null sarà memorizzato nel parametro p2.
   Il programma legge al massimo 10 record dal file e carica i record letti in un vettore da definire.
2. Per ogni record caricato nel vettore il programma crea un processo figlio, il quale provvede ad accedere al proprio record (primo figlio, primo record, secondo figlio secondo record,…) e lanciare in exec il programma previsto nel campo nome del proprio record, passando come parametri i valori utili contenuti nelle stringhe p1, p2 e p3. Si consideri che ogni programma lanciato in esecuzione termini con una exit(1).
3. Il programma del processo padre dopo aver attivato i processi si mette in attesa di al massimo i primi 3 processi che terminano e quando l'attesa è completata provvede a scrivere al terminale la stringa "completata l'esecuzione di N processi figli", dove N rappresenta il numero di processi realmente aspettati. Infine termina la propria esecuzione.

Soluzione: il programma contiene anche parti non richieste, ma inserite se si vuole estrarre e lavorare sul programma

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>


#define F_ESEGUIBILI "programmi.ttt"
#define NULL_VALUE "NULL"
#define MAX_ES 1
#define MAX_GEN 16

struct el { char nome [20]; char p1 [10]; char p2 [10];char p3 [10];};

int generaFile ();

int main ()
{
      FILE * fp;
      pid_t pid;
      struct el es [10];
      int num_processi=0, j;
      int fileOK = 0, status;

      fileOK = generaFile();

      /* apertura file programnmi */
      if (fileOK=0 || (fp = fopen(F_ESEGUIBILI, "rb")) == NULL) {printf("\nError open file %s.\n",
F_ESEGUIBILI);exit(0); }

       /* scansione del file degli eseguibili  fino alla fine del file o per un massimo di MAX_ES */
      fread(&es[num_processi], sizeof(struct el), 1, fp);
      while (num_processi < MAX_ES && !feof(fp)) { num_processi++;fread(&es[num_processi], sizeof(struct
el), 1, fp); }

      //lettura del vettore e generazione dei processi figli
      printf("\nScansione del vettore acquisito");
      for (j=0; j<num_processi; j++)
```

```
          {printf ("\n P%d Nome: %s \t\t p1: %s \t p2: %s \t p3: %s ", j+1, es[j].nome, es[j].p1,
es[j].p2, es[j].p3);
           pid = fork();
           if (pid == -1) {   printf("\nErrore! Anomalia generazione figlio");     exit(0);}
           if (pid == 0) {//figlio
                         printf ("\n\nF%d Mio pid: %d\n", j+1, (int) getpid ());
                         if (strcmp(es[j].p1, NULL_VALUE) == 0)
                             {//printf ("\n\nF%d Eseguo %s \n", j+1, es[j].nome);
                              execl(es[j].nome, NULL);
                            }
                         else if (strcmp(es[j].p2, NULL_VALUE) == 0) {
                                 printf ("\n\nF%d Eseguo %s %s  \n", j+1, es[j].nome, es[j].p1);
                                 execl(es[j].nome, es[j].p1,  NULL);
                                  }
                              else {printf ("\n\nF%d Eseguo %s %s %s \n", j+1, es[j].nome, es[j].p1,
es[j].p2);
                                    execl(es[j].nome, es[j].p1, es[j].p2,NULL);
                                 }
                       //qui non dovrebbearrivare a meno di malfunzionamenti
                       printf("\nErrore! Execl fallita");exit(-1);
                      }
            }

      //il padre attende solo al massimo i primi 3 figli
      j=0;
      while (j< 3 && j <num_processi) {wait(&status); j++;printf ("\nUn figlio ha terminato");}
      printf ("\nCompletata esecuzione di %d processi figli\n\n\n", j);

      fclose(fp);

      return 0;
}

int generaFile () {
      FILE * FP_TEMP;
      struct el es_temp;
      int numero, i;

      FP_TEMP = fopen(F_ESEGUIBILI, "wb");
      if (FP_TEMP==NULL) {
            printf("\nErrore di apertura del file %s.\n", F_ESEGUIBILI);
            return 0;
      }
      else {
            for (i=0;i<MAX_GEN;i++) {
                  strcpy(es_temp.p1, NULL_VALUE);
                  strcpy(es_temp.p2, NULL_VALUE);
                  strcpy(es_temp.p3, NULL_VALUE);

                  numero= rand() % 3;
                  switch (numero) {
                        case 2:                       strcpy(es_temp.p2, "-G\0");
                        case 1:strcpy(es_temp.p1, "-l\0");strcpy(es_temp.nome, "/bin/ls\0");break;
                        case 0:strcpy(es_temp.nome, "/bin/pwd\0");break;

                  /*    default:
                              strcpy(es_temp.p1, NULL_VALUE);
                              strcpy(es_temp.p2, NULL_VALUE);
                              strcpy(es_temp.p3, NULL_VALUE);
                              break;
                  */
                  }

                  printf ("\n %d Nome: %s, num_par: %d", i+1, es_temp.nome, numero);
                  printf ("\n \t\t p1: %s, p2: %s, p3: %s", es_temp.p1, es_temp.p2, es_temp.p3);
                  fwrite(&es_temp,sizeof(struct el), 1, FP_TEMP);
            }
      }
      fclose(FP_TEMP);
      printf ("\n File generato!");
      return 1;
}
```

Esercizio 2.
Scrivere una funzione Carica che riceve come parametro la lista SORG così definita

struct el {char str[20]; struct el *next;}; struct el *SORG=NULL;

e procede a leggere da terminale un testo che viene digitato dall'utente: il testo è composto da parole che usano le sole lettere dell'alfabeto e che sono separate da un solo spazio bianco; l'intero testo è terminato quando l'utente inserisce 1 dopo l'ultima parola inserita.
Per ogni parola individuata la funzione crea una variabile dinamica di tipo struct el dove inserisce la parola nel campo str; nel caso in cui si legga una parola che non sia memorizzabile nel campo str per la sua lunghezza, la funzione provvede a troncarla in modo da inserirla nel suddetto campo e i caratteri in eccesso sono eliminati. La lista alla fine della lettura deve contenere nel primo elemento la prima parola, nel secondo la seconda parola e così via). La funzione infine restituisce la lista generata come parametro.

Esercizio 3.
Sempre con riferimento all'esercizio precedente si scriva la funzione inverti che riceve come parametro la suddetta lista SORG e restituisce sempre come parametro un'altra lista (che chiamo DEST per semplicità) che contiene gli stessi elementi di SOG, ma invertiti, ossia il primo elemento di SORG è l'ultimo di DEST, il secondo di SORG è il penultimo di DEST e così via.

Soluzioni:

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

#define MAX_C 20

struct el{    char str [MAX_C]; struct el* next;};
typedef struct el * TipoLista;


void carica (TipoLista * SORG)
{    char str[MAX_C]; char c;
     TipoLista lista = *SORG;
     TipoLista temp;

     printf ("\nInserisci il testo di caratteri terminato da 1: \n");
     fflush(stdin);
       c=getchar();
       while (c!='1')
       { int i=0;
         while((i<19)&&(c!=' '))  {
               str[i]=c;
               printf("\nlettera=%c",str[i]);
               i++;
               c = getchar();
         }
         str[i]='\0';
         if (c!=' ')
               do c=getchar(); while (c!=' ');
         c = getchar(); //idem
         printf("\n parola=\t\t %s ",str);
         temp =crea(str);
         if (temp != NULL) {
               if (lista == NULL) //Modifico la testa
                { *SORG = temp;  lista = *SORG;}
               else { //inserisco in coda
```

```
                        lista->next = temp;    lista = lista->next;}
        }
    }//end while
}


void inverti (TipoLista SORG, TipoLista * DEST) {
    TipoLista temp;

    printf ("\n Inversione della lista");
    while (SORG !=NULL) {
        temp = crea(SORG->str);

        //inserisco sempre in testa per avere l'invertita
        temp->next = *DEST;
        *DEST = temp;

        SORG = SORG->next;
    }
}


TipoLista crea (char str[20]) {
    TipoLista temp;
    //allocazione memoria
    temp = (TipoLista) malloc (sizeof(struct el));
    if (temp == NULL)
        printf("\nErrore di allocazione!");
    else {
        //inizializzo elemento
        strcpy(temp->str, str) ;
        temp->next = NULL;
    }
    //restituisce indirizzo della memoria allocata
    return (temp);
}
```

Esercizio 4. Rispondere alle seguenti domande, motivando brevemente le risposte in modo convincente.

1. Siano date le seguenti definizioni

   struct comp {int A; struct comp *prec;}; struct comp *lista=NULL;
   e la seguente funzione
   void F()
   { struct comp * var;
   … codice che crea variabili dinamiche collegate tra loro (ad esempio, come lista, come albero,…) e che sono referenziate dalla sola variabile var.
   return;
   }
   Che accade ai dati nell'area di Stack e nell'area di Heap quando la funzione F esegue l'istruzione return?

   STACK al termine dell'esecuzione della funzione sono cancellate tutte le variabili locali della funzione e quindi "var".

   HEAP il termine dell'esecuzione della funzione non produce alcun effetto in quest'area, pertanto deallocando "var" si perde ogni riferimento alle eventuali variabili dinamiche generate che diventano quindi garbage.

2. Cosa significa generare una dangling reference e mostrare una porzione di codice che generi tale situazione.

   Una variabile puntatore che contiene un indirizzo al quale non esiste una variabile utile ai fini del programma.

   … *a, *b;
   a= malloc(sizeof(…));
   b=a;
   free(a);
   b contiene indirizzo non valido.

3. Quali delle seguenti sequenze di 8 bit (codificate in esadecimale rappresentano un numero negativo nella rappresentazione in complemento a 2 e perché.

   A. 7F B. 55 C. A6 D. 08

   La C. perché il primo degli otto bit contiene un 1

4. In quali delle seguenti somme (usando il complemento a 2 nella codifica dei numeri) si genera un overflow e perché

A.		B.		C.	
0011		0100		1100	
+ 1010		+ 0100		+ 1100	
1101		1000		(1)1000	

   La B perché due operandi di segno concorde (primo bit) producono un risultato di segno discorde. Come controprova tradurre in decimale le somme.

5. rappresentare la seguente sequenza di bit 1111.0100.1001.1111 in esadecimale e in decimale sapendo che è stata codificata in complemento a 2

decimale = - 2913 esadecimale (dal binario): F49F esadecimale (dal decimale): - 0B61

6. Data la definizione della variabile double v= - 33,2 descrivere la sua rappresentazione in virgola mobile in base allo standard IEEE754 semplice precisione FP64 descritto a lezione;

   a) Definire prima il numero nel formato normalizzato intermedio (TRONCARE parte decimale alla terza cifra decimale):

   $$- 1{,}037 * 2^5$$

   b) Codificare poi il numero normalizzato usando il formato FP32:
   S (1 bit di segno): 1
   M (23 bit per la mantissa – fermarsi ai primi 6 bit): 000010
   E (8 bit per l'esponente): 10000100

7. Che succede agli eventuali figli generati nell'esercizio 1 che sono in esecuzione quando il processo padre termina la propria esecuzione

   Rimangono in esecuzione, ma sono ereditati dal processo INIT

8. (con riferimento alla domanda precedente) In che stato passano tali figli all'atto della loro terminazione e indicare se tale stato sia definitivo e perché.

   Passano in stato ZOMBIE fino a quando il processo INIT li elimina definitivamente (kill).