Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni del Compitino del 10 Aprile 2013

Esercizio 1. Scrivere un programma C utilizzi le funzioni `fgetc` e `fputc` per eseguire le seguenti operazioni:

- prende come *parametri della linea di comando* due nomi di file;
- copia il contenuto del primo file nel secondo (riscrivendolo se esiste già), sostituendo tutti le sequenze di caratteri di tabulazione `'\t'` e di spazi con uno spazio singolo.

Non è permesso l'utilizzo di `fgets`, `fputs`, `fprintf` o `fscanf`.

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

void printerror(char *str, int n) {
  fprintf(stderr, "Errore %d: %s\n", n, str);
  exit(n);
}

int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  int c, res, flag;

  if (argc != 3)
    printerror("Specificare due file come argomenti", 1);
  fin = fopen(argv[1],"r");
  if (fin == NULL)
    printerror("Errore nell'apertura del primo file.", 2);
  fout = fopen(argv[2],"w");
  if (fout == NULL)
    printerror("Errore nell'apertura del secondo file.", 2);

  flag = 0;
  c = fgetc(fin);
  while (c != EOF) {
    if(c == ' ' || c == '\t') {
      c = ' ';
      flag++;
    } else {
      flag = 0;
    }
    if(flag <= 1) {
      res = fputc(c, fout);
      if(res == EOF) {
        printerror("Errore nella scrittura del secondo file.", 3);
      }
    }
    c = fgetc(fin);
  }

  if ( fclose(fin) != 0 ) {
    printerror("Errore nella chiusura del primo file.", 4);
  }
  if ( fclose(fout) != 0 ) {
```

```
    printerror("Errore nella chiusura del primo file.", 4);
  }
  return 0;
}
```

Esercizio 2. Scrivere un programma C che utilizzi le funzioni `fgets` e `fputs` per simulare il comando `grep`. Il programma deve:

- prendere come *parametri della linea di comando* un pattern e due nomi di file;
- scorrere il contenuto del primo file, e scrivere sul secondo file solamente le righe che contengono il pattern.

Per trovare un pattern all'interno di una stringa, si può usare la funzione

```
char* strstr(char* str, char* pattern)
```

presente in `string.h`, e che ritorna un valore diverso da `NULL` se `pattern` si trova all'interno di `str`. Non è permesso l'utilizzo di `getc`, `putc`, `fgetc`, `fputc`, `fprintf` o `fscanf`.

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

int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  char s[100];
  char *res;
  int n = 1;

  if (argc != 4) {
    printf("Uso: %s pattern inputfile outputfile\n", argv[0]);
    return 1;
  }

  fin = fopen(argv[2],"r");
  if (fin == NULL) {
    printf("Errore nell'apertura del file %s.\n", argv[2]);
    return 1;
  }
  fout = fopen(argv[3],"w");
  if (fin == NULL) {
    printf("Errore nell'apertura del file %s.\n", argv[3]);
    return 1;
  }

  res = fgets(s, 100, fin);
  while (res != NULL) {
    if(strstr(s, argv[1]) != NULL) {
    if(fputs(s, fout) == EOF) {
      printf("Errore nella scrittura del secondo file.");
      return 3;
    }
}
    res = fgets(s, 100, fin);
    n++;
  }

  if ( fclose(fin) != 0 ) {
    printf("Errore nella chiusura del file %s.\n", argv[1]);
    return 1;
  }
```

```
  return 0;
}
```

Esercizio 3. Scrivere un programma C che utilizzi le funzioni `fprintf` e `fscanf` per fare le seguenti operazioni:

- prendere come *parametri della linea di comando* due nomi di file;
- leggere una sequenza di numeri *razionali* dal primo file;
- calcolare la media aritmetica della sequenza e scrivere il risultato nel secondo file.

Non è permesso l'utilizzo di `getc`, `putc`, `fgetc`, `fputc`, `fgets` o `fputs`.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  FILE *fin, *fout;
  int res, n;
  double d, sum;

  if (argc != 3) {
    printf("Uso: %s inputfile outputfile.\n", argv[0]);
    return 1;
  }

  fin = fopen(argv[1],"r");
  if (fin == NULL) {
    printf("Errore nell'apertura del file %s.\n", argv[1]);
    return 1;
  }
  fout = fopen(argv[2],"w");
  if (fout == NULL) {
    printf("Errore nell'apertura del file %s.\n", argv[2]);
    return 1;
  }

  n = 1;
  sum = 0.0;
  res = fscanf(fin, "%lf", &d);
  while (res != EOF) {
    if(res != 1) {
      printf("Errore : file %s in formato non corretto.\n", argv[1]);
      return 1;
    }
    sum += d;
    n++;
  res = fscanf(fin, "%lf", &d);
  }

  if(fprintf(fout, "%lf\n", sum/n) == EOF) {
    printf("Errore nella scrittura del file %s.\n", argv[2]);
    return 1;
}

  if ( fclose(fin) != 0 ) {
    printf("Errore nella chiusura del file %s.\n", argv[1]);
    return 1;
```

```
  }
  if ( fclose(fout) != 0 ) {
    printf("Errore nella chiusura del file %s.\n", argv[2]);
    return 1;
  }
  return 0;
}
```