

Capitolo 14 – Argomenti avanzati

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Utilizzare gli argomenti a linea di comando

- Passare gli argomenti al main da DOS o UNIX
 - Definire il main come

```
int main( int argc, char *argv[] )
```
 - int argc
 - Numero di argomenti passati
 - char *argv[]
 - Array di stringhe
 - Ha i nomi degli argomenti in ordine
 - argv[0] è il primo argomento
 - Esempio: \$ mycopy input output
 - argc: 3
 - argv[0]: "mycopy"
 - argv[1]: "input"
 - argv[2]: "output"

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
1 /* Fig. 14.3: fig14_03.c
2 Using command-line arguments */
3 #include <stdio.h>
4
5 int main( int argc, char *argv[] )
6 {
7     FILE *inFilePtr; /* input file pointer */
8     FILE *outFilePtr; /* output file pointer */
9     int c; /* define c to hold characters input by user */
10
11     /* check number of command-line arguments */
12     if ( argc != 3 ) {
13         printf( "Usage: copy infile outfile\n" );
14     } /* end if */
15     else {
16
17         /* if input file can be opened */
18         if ( ( inFilePtr = fopen( argv[ 1 ], "r" ) ) != NULL ) {
19
20             /* if output file can be opened */
21             if ( ( outFilePtr = fopen( argv[ 2 ], "w" ) ) != NULL ) {
22
23                 /* read and output characters */
24                 while ( ( c = fgetc( inFilePtr ) ) != EOF ) {
25                     fputc( c, outFilePtr );
26                 } /* end while */
27
28             } /* end if */

```



Outline

fig14_03.c (Part 1 of 2)

argv[1] is the second argument, and is being read.

argv[2] is the third argument, and is being written to.

Loop until End Of File. fgetc a character from inFilePtr and fputc it into outFilePtr.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
29     else { /* output file could not be opened */
30         printf( "File \"%s\" could not be opened\n", argv[ 2 ] );
31     } /* end else */
32
33 } /* end if */
34 else { /* input file could not be opened */
35     printf( "File \"%s\" could not be opened\n", argv[ 1 ] );
36 } /* end else */
37
38 } /* end else */
39
40 return 0; /* indicates successful termination */
41
42 } /* end main */
```



Outline

fig14_03.c (Part 2 of 2)

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Compilazione di più file sorgenti

- Programmi con più file sorgenti
 - La definizione delle funzioni deve essere presente in un solo file
 - Le variabili globali sono accessibili alle funzioni dello stesso file
 - Esempio:
 - se l'intero `flag` è definito in un file
 - il suo utilizzo in un altro file deve includere l'istruzione
`extern int flag;`
 - `extern`
 - la variabile è definita in un altro file
 - i prototipi di funzione possono essere utilizzati in altri file senza l'istruzione `extern`
 - un prototipo in ogni file che usa la funzione

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Compilazione di più file sorgenti

- `static`
 - Specifica che le variabili possono essere usate solo nel file in cui sono definite
- Programmi con più file sorgenti
 - Tedioso compilare ogni cosa anche in casi di piccoli cambiamenti ad un solo file
 - è possibile compilare solo il file modificato
 - Le procedure variano da sistema a sistema
 - UNIX: `make utility`

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.