Text and Binary Files Processing

COEN 11

Text Files

Text Files

- Collections of characters saved in a secondary storage (e.g., on a disk)
- Have no fixed size
- End marked with a special character <eof>
- End of lines are marked by a newline ('\n') character

Text Files

- Example
  - Text file:

```
This is a text file!<newline>
It has two lines.<newline><eof>
```

  - Actual disk file:

```
This is a text file!<newline> It has two lines.<newline><eof>
```

Text Files

- All textual input and output data are actually a continuous stream of character codes
  - We refer to a data source or destinations as an input stream or an output stream
  - These general terms can be applied
    - to files
    - to the keyboard,
    - to the screen, and
    - to any other sources of input data or destinations of output data

5

Text Files

- Keyboard and Screen
  - stdin
    - keyboard's input stream
  - stdout
    - "normal" output stream associated with the screen
  - stderr
    - "error" output stream associated with the screen
- Streams are treated like text files
  - Their individual components are characters

6

Text Files

- To read from stdin

  r = scanf ("%d", &num);

  or

  ch = getchar ()
- The characters are read sequentially
  - Consecutive calls to scanf and/or getchar will read consecutive elements from the keyboard
  - ^D usually represents <eof>

7

Text Files

- scanf
  - Returns how many elements were read
  - Placeholders define the type
    - Examples
      - ✓ %c, %d, %f, %s
      - ✓ For limiting the size of a string
        %Ns – reads up to N characters
        (array size should be N+1)
      - ✓ For lines

        r = scanf("%[^'\n']", string_name);

8

Text Files

- scanf
  - Only reads the characters as specified by the placeholder
    - Spacers (\n, \t, or spaces) may be left in the buffer from previous calls
    - To clear the buffer
      - ✓ fpurge() -- erases any input or output buffered in the given stream
      - ✓ Example

```
fpurge (stdin);
r = scanf("%[^'\n']", string_name);
```

9

Text Files

- Other functions
  - To read lines
    - fgets (array, SIZE, stdin);
      - ✓ Includes the \n at the end of the line
      - ✓ If used after scanf, may need to clear the buffer

10

Text Files

- To write to stdout

```
printf ("%d", num);
or
putchar (ch)
```

- The file is written sequentially
  - Consecutive calls to printf and/or putchar will write consecutive elements to the file

11

Text Files

- printf - common escape sequence
  - '\n' → new line
  - '\t' → tab
  - '\f' → form feed (new page)
  - '\r' → return (back to column 1)
  - '\b' → backspace

12

Text Files

- printf – placeholders for format string
  - %c → character
  - %s → string
  - %d → integer (base 10)
  - %o → integer (base 8)
  - %x → integer (base 16)
  - %f → floating point
  - %e → floating point in scientific notation
  - %E → floating point in scientific notation
  - %% → a single % sign

13

Text Files

- printf – field width, justification, and precision in format strings
  - printf ("%5d%4d\n", 100, 2);
  - printf ("%2d with label\n", 5210);
  - printf ("%-16s%d\n", "Joe Smith", 28);
  - printf ("%15f\n", 981.48);
  - printf ("%10.3f\n", 981.48)
  - printf ("%7.1f\n", 981.48)
  - printf ("%12.3e\n", 981.48)
  - printf ("%.5E\n", 0.098148);

14

Text Files

- printf
  - Data is buffered until
    - a newline or
    - the buffer is full
  - To empty the buffer
    - fpurge- erases the buffer
    - fflush – outputs the buffer

15

Text Files

- To open a text file for reading

```
FILE *infp;

if ((infp = fopen ("data.txt", "r")) == NULL)
   printf ("cannot open the file data.txt\n");
```

- The file is always open for reading from the beginning

16

Text Files

- To open a text file for reading when the name is stored in a string

```
char  file_name[50];
FILE *infp;

if ((infp = fopen (file_name, "r")) == NULL)
   printf ("cannot open the file %s\n", file_name);
```

17

Text Files

- To read from the text file

```
ret = fscanf (infp, "%d", &num);
or
ch = getc (infp)
```

- The file is read sequentially
- Returned value
  - fscanf -- number of values converted
  - getc -- the character read
- At the end of the file
  - getc return EOF
  - scanf returns EOF if it was detected before any conversions

18

Text Files

- To open a text file for writing from the beginning

```
FILE *outfp;

if ((outfp = fopen ("data.txt", "w")) == NULL)
   printf ("cannot open the file data.txt\n");
```

19

Text Files

- To open a text file for writing from the beginning when the name is stored in a string

```
char file_name[50];
FILE *outfp;

if ((outfp = fopen (file_name, "w")) == NULL)
   printf ("cannot open the file %s\n", file_name);
```

20

Text Files

- To open a text file for writing from the end (append)

```
FILE *outfp;

if ((outfp = fopen ("data.txt", "a")) == NULL)
  printf ("cannot open the file data.txt\n");
```

21

Text Files

- To open a text file for writing from the end (append) when the name is stored in a string

```
Char  file_name[50];

FILE *outfp;
if ((outfp = fopen (file_name, "a")) == NULL)
  printf ("cannot open the file %s\n", file_name);
```

22

Text Files

- To write to the text file

  fprintf (outfp, "%d", num);

  or

  putc (ch, outfp)

- The file is written sequentially
  - Consecutive calls to fprintf and/or putc will write consecutive elements to the file

23

Text Files

- To close a text file

  fclose (infp);

  fclose (outfp);

24

Binary Files

Binary Files

- When text files are used
  - Internal data needs to be converted to and from characters
  - These conversions are done by scanf and printf or by the program itself

Binary Files

- When a program produces output files which are used as input files for other programs
  - If there is no need for a human to read the file
    - Converting information into a stream of characters and back into internal format is a waste of computational cycles and time.
- To avoid these unnecessary conversions
  - Use binary files

Binary Files

- Files containing binary numbers that are the computer's internal representation of each file component
- Created by executing a program that stores directly in the computer's internal representation of each file component

Binary Files

- Actually just a stream of zeros and ones and cannot be read with a text editor

29

Binary Files

- Example
  - If a char x = 2 is written to a file, the file will have the following data:

    00000010
  - If an short x = 2 is written to a file, the file will have the following data:

    0000000000000010
  - If an int x = 2 is written to a file, the file will have the following data:

    00000000000000000000000000000010

30

Binary Files

- To open a binary file for reading

```
FILE *infp;
if ((infp = fopen ("data.txt", "rb")) == NULL)
   printf ("cannot open the file data.txt\n");
```

- The file is always open for reading from the beginning
- The file name can also be stored in a string.

31

Binary Files

- To read from a binary file

  ret = fread (&x, sizeof (int), 1, infp);

32

Binary Files

- To read from a binary file

  ret = fread (&x, sizeof (int), 1, infp);
- The file is read sequentially
  - Consecutive calls to fread will read consecutive elements from the file
- The next integer is placed into the variable x
- Function fread returns the number of elements read.
- At the end of the file, fread returns -1

33

Binary Files

- To open a binary file for writing from the beginning

```
FILE *outfp;
if ((outfp = fopen ("data.txt", "wb")) == NULL)
   printf ("cannot open the file data.txt\n");
```

- The file name can also be stored in a string

34

Binary Files

- To open a binary file for writing from the end (append)

```
FILE *outfp;
if ((outfp = fopen ("data.txt", "ab")) == NULL)
   printf ("cannot open the file data.txt\n");
```

- The file name can also be stored in a string

35

Binary Files

- To write to a binary file

  ret = fwrite (&x, sizeof (int), 1, outfp);

36

Binary Files

- To write to a binary file

  ret = fwrite (&x, sizeof (int), 1, outfp);
- The contents of variable x is written to the file
- The file is written sequentially
  - Consecutive calls to fwrite will write consecutive elements to the file
- Function fwrite returns the number of elements written.

37

Binary Files

- To close a binary file

  fclose (infp);

  fclose (outfp);

38

Files -- Functions

- fopen, fclose
- fprintf, fscanf
- fgets
- fread, fwrite
- fflush
- fseek

39