ECS 36A, May 5, 2023

Announcements

1. On Wednesday, May 10, we will resume in-person classes
2. I will also hold office hours in person beginning then
    - Until then, same Zoom link for both until then:

      https://ucdavis.zoom.us/j/95840281592?pwd=a1NhNmpLNFp2VVVrYkpGY3pDcWdlQT09
3. The midterm will be *next* Friday, May 12

Formatted Write

These write to a file or the standard output as indicated by the format string.

- printf (*format*, *variables*)
  - Same as fprintf(stdout, *format*, *variables*)
- sprintf(*string*, *format*, *variables*)
  - Writes its output into *string*
- fprintf(*fp*, *format*, *variables*)
  - Writes to the file with file pointer *fp*

The Formats

A string with the following format indicators:

- Anything except the conversion specifiers is copied
- Conversion specification: begins with %; “%%” outputs a single “%”
  - ”%d”: print next argument as a decimal integer
  - “%o”: print next argument as an octal integer
  - “%x”: print next argument as a hexadecimal integer
  - “%c”: print next argument as a single character
  - “%s”: print next argument as a string (sequence of characters)
  - “%f”: print next argument as a floating point number, like 12.345678
  - “%e”: print next argument as a floating point number, line 1.2345678e1

Format Modifiers

- %#x, %#o: put “0x” in front of hexadecimal output, 0 in front of octal output
- %+d: always print a sign, either “+” or “-”
- %*n*d: print integer in a field of *n* characters
- %0*n*d: as above, but pad with leading 0s
- %-*n*d: left align the number in the field

If the number is bigger than the field, the field size is ignored

More Format Modifiers

For floating-point numbers:

- %*n*.*m*f: print float in a field of size *n*, with *m* decimal digits
  - If *m* is 0, then no decimal point or decimal fraction is printed
  - Printed value is in normal floating point form, like 123.4567
- %*n*.*m*e: print float in a field of size *n*, with *m* decimal digits
  - If *m* is 0, then no decimal point or decimal fraction is printed
  - Printed value is in scientific notation, like 1.234567e2

Still More Format Modifiers

For characters:

- %12.4c: this ignores the .4 and prints the character in a field of 12 wide
- %012c: as above, but ignores the 0

For strings:

- %12.4s: prints first 4 characters of string in a field 12 wide
- %-12.4s: left justifies the above
- %012s: this ignores the 0

Useful File Functions

```
int feof(FILE *fp)
```

- Nonzero if file pointer fp is at EOF; 0 if not

```
int ferror(FILE * fp)
```

- Nonzero if error indicator for fp is set; 0 if not

```
void clearerr(FILE * fp)
```

- Clear EOF and error indicators for fp

Reading Binary Files

- Binary files contain non-ASCII values; for example, integers are written into the file as integers, not printed representations of integers

```
fread(ptr, sz, num, fp)
```

  - Reads num items of size sz from file fp and stores them beginning at address ptr
  - ptr can point to any type (it’s declared as void *ptr)
  - Returns number of items actually read; on error or EOF, returns the number of items actually read (or 0 if none were)
    - Use feof() and ferror() to determine if it’s an EOF or an error when it returns 0

Writing Binary Files

- Writes out the data in the form it is stored in in memory.

```
fwrite(ptr, sz, num, fp)
```

  - Writes num items of size sz beginning from address ptr to file fp
  - ptr can point to any type (it’s declared as void *ptr)
  - Returns number of items actually written; on error, returns the number of items actually written (or 0 if none were)