Basic IO in C

Ion Mandoiu
Laurent Michel

Basic I/O

- On the menu [Chapter 5, C for Java Programmers]
  - getchar / putchar
  - getc / putc
  - Formatted
    - printf
    - scanf

getchar / putchar

- IO done one character (8-bits) at a time.
  - Simple API
    - int getchar(void)
      - Reads a character from STDIN
      - Returns the character just read in
    - int putchar(int)
      - Write the character received as argument on STDOUT
      - Returns the character that was just written out.
- Both assume the existence of two “default” Streams
  - STDIN Standard Input [an integer really]
  - STDOUT Standard Output [ditto]

getc / putc

- Same idea as getchar / putchar
- But…
  - You get to provide the stream!

```
int fputc(int c, FILE *stream);

int putc(int c, FILE *stream);
```

- All you need is an IO stream
  - Which you get with fopen
  - Which you close with fclose

Streams

- Simple API

```
FILE* fopen( const char *restrict filename,
             const char *restrict mode);
```

  - Give the filename
  - Give a string to indicate
    - “r” : Reading mode
    - “r+” : Read and write
    - “w” : Writing mode
    - “w+”: Read and write, but the file is created or truncated
    - “a” : Write, the file is created if it does not exist.
    - “a+” : Write, the file is created if it does not exist, we start by adding at the end.

Stream

- When done with a stream…
  - Close it!

```
int  fclose(FILE *stream);
```

  - Returns
    - 0 if it worked
    - EOF if there was a problem (and errno is set)

errno is a global variable (type int)
Beware:
this is not thread friendly!

End-Of-File?

- Easy to do
  - Simply use the following API

```
int  feof(FILE *stream);
```

  - Returns TRUE (value non-zero) if we reached the end-of-file
  - Returns FALSE (0) if we still have some data

Rewinding / Seeking

- Key Idea
  - Every open file maintain a seek pointer
    - This is a position into the file from the beginning.
    - Reading (or writing) modifies the seek pointer (advances it!)
  - You can move the seek pointer arbitrarily
  - You can tell where the seek pointer is

```
int fseek(FILE *stream, long offset, int whence);
int fseeko(FILE *stream, off_t offset, int whence);
long  ftell(FILE *stream);
off_t ftello(FILE *stream);
void  rewind(FILE *stream);
```

Formatted I/O

- When you wish to do some more complex I/O
- Use the printf / scanf family of functions
  - Java's format method is inspired by those.
  - Very simple idea
  - Provide a "format" string
  - Provide all the arguments to format
  - Format string governs how many things to format and how
  - Great for strings, booleans, integers, floats, pointers,…..

Formatted I/O Example

- Check the printf man page for formatting details
  - man -S3 printf
- Simple example formatting with a string

```c
#include <stdio.h>

int main()
{
   char name[] = "world";
   printf("Hello %s\n",name);
   return 0;
}
```

Formatted I/O Example

- Check the printf man page for formatting details
  - man -S3 printf
- Simple example formatting with a string and an integer

```c
#include <stdio.h>

int main()
{
   char name[] = "Jake";
   int age = 22;
   printf("%s is %d years old.\n",name,age);
   return 0;
}
```

Formatted I/O Example

- Check the scanf man page for formatting details
  - man -S3 scanf
- Simple example reading a string and two integers

NOTE

Some & symbols?…

Pointers!

```
#include <stdio.h>

int main()
{
   char name[128];
   int pears  = 0;
   int apples = 0;
   scanf("%s %d %d",name,&pears,&apples);
   printf("%s ate %d apples and %d pears.\n",name,apples,pears);
   return 0;
}
```

Stream oriented versions

- Going together with printf/scanf there is….
  - fprintf
  - fscanf
- Same idea but work on a specified stream (rather than stdout/stdin)

```
int printf(const char * restrict format, ...);

int fprintf(FILE * restrict stream,
            const char * restrict format, ...);
```