Day14 A

Young W. Lim

2017-12-26 Tue

Outline

Based on

"C How to Program",
Paul Deitel and Harvey Deitel

Characters

- a character constant is an `int` value
- a character is represented by single quotes
- the value of a character constant is
  the character's integer value in the machines' character set
    - ASCII

Strings

- a series of characters
- treated as a single unit
- may include
  - letters, digits
  - various special characters
    such as +, -, *, / and $.
- $string literal$s or $string constant$s are in double quotes
- a string is accessed via a pointer to its first character
- the value of a string is the the address of its first character

Intialization

- a character array can be initialized with a string constant.
  `char a[ 20 ] = "Hello, world!";`
  each element of the array a can be changed
- a charatter pointer can be intialized with a string constant.
  `char *p = "Hello, world!";`
  no element of the string constant can be modified

Character Pointer Initialization (1)

- `char *s = "Hello, World!";`
- a character <u>pointer</u> s is declared with an initialization
- the value of s is an address
  of a memory location where a character resides
- "Hello, World!" is a constant character string
  stored in the read-only memory region (defined by a compiler)
- "Hello, World!" returns the address
  of the 1st character in the string (the address of 'H')
- s points to this address of the 1st character

Character Pointer Initialization (2)

- `char *s = "Hello, World!";`
- `s[ 5 ]=0` causes a run-time error (Segmentation Faults)
- though this string is a string constant,
  it is not explicitly declared with `const`,
- therefore, no error message will be shown during compilation
- but during execution, the "Segmentation fault" error occurs
- because s[5]=0 attempts to change its element
  in the read-only memory location.
- we can compile but cannot execute normally.

Types of Unformatted IO

	stdio		file	
character	getc getchar	putc putchar	fgetc	fputc
string	gets	puts	fgets	fputs

- c : character
- s : string
- f : file
- get : read, input
- put : write, output

- getchar() = getc(stdin)
- putchar(c) = putc(c, stdout)
- gets(s) = fgets(s, stdin)~
- do not use gets
  - insecure (no bound check)

Standard I/O Library Functions

- `fgets` reads characters until
  - a newline character or
  - the end-of-file character

  is encountered
  - arguemnt :
    - an array of type `char`
    - the maximum number of characters that can be read
    - the stream from which to read
  - a null character is appened to the array after finishing
- `char *fgets(char *s, int size, FILE *stream);`

getc/putc vs fgetc/fputc

	stdio		file	
character	getc	putc	fgetc	fputc

- getc can be implemented as a macro
- fgetc cannot be implemented as a macro
  - the argument to getc should not be an expression with side effects
  - since fgetc is guaranteed to be a function,
    pointer to fgetc can be used
  - calls to fgetc probably take longer than calls to getc

    – Advanced Programming in Unix Environment
- practically, no significant differences
  - getc(stream) = fgetc(stream)
  - putc(c, stream) = fputc(c, stream)

Stream

- a common, logical interface to the various devices
- a stream is be a logical interface to a file
    - a disk file
    - a tape file
    - a port
    - the screen (stdout)
    - the keyboard (stdin)
- Although files differ in form and capabilities,
  all streams are the same. (a uniform interface)

`https://www.le.ac.uk/users/rjm1/cotter/page_74.htm`

fopen and fclose

- FILE * fopen (const char * filename, const char * mode);
- int fclose (FILE * stream);

"r"	read
"w"	write
"a"	append
"r+"	read/update
"w+"	write/update
"a+"	append/update

```
#include <stdio.h>
int main ()
{
  FILE * pfile;  // pfile stream

  pfile = fopen ("test.txt","w");

  if (pfile!=NULL) {
    fputs ("fopen example",pfile);
    fclose (pfile);
  }
}
```

getchar() and a buffer

- there is an underlying buffer/stream
- when you enter text, the text is stored in a buffer somewhere
- the enter key must be pressed before `getchar()` gets anything to read
- `getchar()` can stream through the buffer one character at a time
- each read returns a character
    - until it reaches the end of the buffer (EOF)
    - until you press CTRL+D (end of file)

https://stackoverflow.com/questions/3676796/how-does-getchar-work

EOF

- EOF isn't a character that exists in the stream, but a sentinel value
- to indicate when the end of the input has been reached.

`https://stackoverflow.com/questions/3676796/how-does-getchar-work`