Reading data - `scanf()` etc.

Alon Efrat
Computer Science Department
University of Arizona

The `\`function\`' getchar()

- `getchar()` reads and returns a single character from the standard input.
- If reaches the end of the file returns EOF (end of file)
  - char c; int cnt = 0;
  - while( ( c = getchar() ) != EOF)
    - If (c == '\n' ) cnt++;
    - print(\" The number of \"s in the file is %d\" , cnt) ;
  /* When typing from the keyboard, ^D is usually EOF */

The function `putchar()`

- `putchar(x)` prints `x` to the standard input (i.e., without redirection, terminal).
- `x` is a single character
- Returns the value that was printed, or EOF
  - `putchar\"H\" ; putchar\"e\" ; putchar\"l\" ; putchar\"o\" ; putchar\"\n\"`
  - Prints Hello\n
`scanf()` - reading a few variables

- Analogous to (surprise) printf
- Used to read data in a formatted way, and assign it to variables.
- Returns the number of successful variable assignments.
  - `scanf(\"%d%lf%lf\" , &c1, &i1, &i2, &x) ;`
  /* input: 512 47 192 23.193 */
  Assignment: c=5, i1=12, i2=192, x=23.193, n=4 * /

Reading a string

- `char buf1[20];`
- `char buf2[] = \"hello world\" ;`
- `printf(\"%s\" , buf2 ) ;`
- `scanf(\"%s\" , buf1 ); /* Note – no & */`
- /* Careful – no checks that the input string is of length < 20 */

Chars and integers - some facts

- A character is stored in memory as an 8-bits integer. The value which is stored is the ascii value of the char
- `char c=\'A\'` has the same effect as `char c=96;`
- Only the format of the printing determines whether a number (between 0 and 255) would be printed as a character, or as an integer.
  - `printf(\"%d %d, %c%c\", 96 , \'A\' , 96, \'A\' );`
  - Output: 96 96, AA
- `char c; c=getchar();`
- If `(c== \'a\' & & c<=\'z\' ) print("%c is a small letter\n\",c );`