**Functions on Files**

**getChar()**

* Reads 1 char from standard input and then eventually returns EOF and stops

**putChar()**

* Writes 1 char to the standard output stream

**fgets(Buf,Size,File)**

* Reads from a name file, puts into a buffer, reads until it says newline char and fill the buffer up. If it reaches the end of the buffer before it reaches a newline character, then it will stop.
* It won’t overflow the buffer as **gets()** would.

**Fputs(Buf,File)**

* Writes a string into a file, not including the null character.

**Fscanf(File,Fmt,Ptrs)**

* Reads formatted input from a file
* **Scanf(Fmt,Ptrs)** is exactly the same, except reads from standard input rather than a file

**Fprintf(File,Fmt,Expr)**

* Writes to a specified file
* **Printf(Fmt,Expr)** is exactly the same, except writes to standard output rather than to a file

UNIX / LINUX has two different kind of representations of files / byte streams

Two kinds of file access data structures:

* **FILE \*fp** (An abstract data type. A file pointer or stream)
  + Example: For fprintf, it will read a big chunk of chars from the file, store them in a buffer and then it will scan / read through the buffer until it gets to the end. When that is complete, it will go back to the file and take in another big chunk of data and keep going.
  + Defined in <stdio.h> in C
* **Int fd** (file descriptor)
  + A simple integer value that gives you un-buffered access to a byte-stream.
  + Defined in <unistd.h> system call
  + System calls include: **open()**, **close()**, **read()**, **write()**, **lseek()**, **stat()**
  + These are defined in section 2 of the manual e.g. **man 2 open()**