**L4 - Working with 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. Benefit = It won’t overflow the buffer like **gets()**

**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)** = Same as above except reads from standard input rather than a file

**Fprintf(File,Fmt,Expr)** = Writes to a specified file

**Printf(Fmt,Expr** = Same as above except writes to standard output rather than to a file

UNIX / LINUX has more operations on files / byte streams

Two kinds of file access data structures:

* **FILE \*fp** = Keeps track of a file being accessed via. a FILE ptr.
  + It is an abstract data type. A file pointer or stream. Defined in <stdio.h> in C
* **Int fd** (file descriptor) = A simple integer value that gives you un-buffered access to a byte-stream
  + In simple terms = An integer that uniquely represents an opened file in your operating system
  + If your process opens 10 files, then your Process table will have 10 entries for file descriptors.
  + Defined in <unistd.h> system call
  + System calls include: **open()**, **close()**, **read()**, **write()**, **lseek()**, **stat()**

Establishing access to a file:

* **FILE \*fopen(char \*FileName, char \*Mode)**  
  Attempts to open a named file with modes:
  + **“r”** = Opens file for reading. File must exist
  + **“w”** = Creates empty file for writing. If file name exists, content is erased and file is considered new.
  + **“a”** = Appends to a file. Append data at end of the file. File is created if it doesn’t exist.
  + **“r+”** = Opens file to update both ready and writing. File must exist.

Returns **NULL** if non-existent or non-readable file

* **int open(char \*FileName, int Flags, int Mode)**

Attempts to open named file

**Abstract Data Types (ADTs)**

Separates interface from implementation.

* Similar to using printf(), scanf() etc.
* Users don’t care how it works, just how to use it.
* Our way of implementing ADTs:
  + ptr to a representation of the data type, user doesn’t know what the implementation looks like, they just pass the ptr around and do things that changes the underlying representation.
* **ADT Interface** = user view of the data structure
* **ADT Implementation** = concrete definition of the data structures + definition of functions for all operations

Many of the ADTs consists of a **collection of items**

* Each item may be a simple type or an ADT
* Items often have a **key**
* Collections may be categorised by
  + **Structure**: Linear (list), Branching (tree), cyclic (graph) 🡪 **3 levels of data structures in this course**
  + **Usage**: Set, matrix, stack, queue, search-tree, dictionary **🡪 What operations the ADTs provide**

**Set ADTs**

Set operations:

* **Set SetUnion(Set,Set)** = Take two sets and combine, return new set.
* **Set SetIntersection(Set,Set)** = Find items that exist in both sets, return new set.

**REMEMBER, for complexity analysis, we only care about the highest order cost.**



