Summary Page: Low-Level I/O (I/O System Calls)
CISC 220, fall 2012

Note: the text advises using `creat` instead of `open` to create a new file, a practice which is a bit outdated. Use `open` instead.

Representing Files: In low-level I/O, a file is represented by a non-negative integer.

Headers To Include For Low-Level I/O: `<fcntl.h>` and `<unistd.h>`.

Opening Or Creating a File:

```
int open(char *filename, int flags, [mode_t mode]);
```

Possible values for the flags:

- `O_RDONLY`: open for reading only
- `O_WRONLY`: open for writing only
- `O_RDWR`: open for both reading and writing
- `O_CREAT`: for writing, creates the file if it doesn't exist
- `O_TRUNC`: for writing, erase the old contents of the file if it exists
- `O_APPEND`: for writing, append to the old contents of the file if it exists
- `O_EXCL`: for writing, if the file already exists don't write to it (return a negative integer)

You can combine these values with the bitwise or operator "`|`" – for example, `O_WRONLY|O_CREAT`.
`mode` is an optional parameter, needed only if we're creating a new file. It gives the octal permissions for the new file. A common permission is 0644, which gives the owner of the file permission to read and write, while other users can only read it. Another is 0600, which gives the owner read and write permission while other users can't read or write it.

A return value less than zero signifies an error.

Closing a File:

```
int close(int filedes)
```

where `<filedes>` is a number obtained from a call to `open` or `creat`.
Non-zero return value signifies an error.

Reading Bytes From a File:

```
ssize_t read(int filedes, void *buffer, size_t n);
```

`filedes` is a file number obtained from a call to `open`.
`buffer` is the address we want to read into.
`n` is the maximum number of bytes to read.
`ssize_t` & `size_t` are integer types.
The return value is the number of bytes actually read – usually the parameter n, but less if we hit the end of the file before we could read n bytes. If `read` returns -1, this signifies an error.

Writing Bytes To a File:

```
ssize_t write(int filedes, void *buffer, size_t n);
```

`filedes` is a file number obtained from a call to `open`.
`buffer` is the address of the start of the data we want to write
`n` is the number of bytes to write.
The return value is the number of bytes actually written, or -1 in case of an error.

Standard Input, Output and Error Files

The standard input, output and error files are files 0, 1, and 2. For better readability, use the macros `STDIN_FILENO, STDOUT_FILENO and STDERR_FILENO, defined in <unistd.h>.`