CS 214 / 2021-02-22
===================

```
Notes on assignment 2:

An array list is a list (meaning a sequence of values)
that is stored in memory using an array.

- we want to be able to add elements to the list
        - arrays are fixed size; lists can grow

- we don't want to have to reallocate the array every time we
        add to the list
        - reallocating is O(n) time in the length of the list
        -> we allow the array to be bigger than the list
        -> when we reallocate, we give ourselves extra space to
           store future items that might be added later

-> we need to store two numbers about the list
        - the length of the underlying array
        - the amount of the array that is currently in use
                - i.e., the length of the list

        - in arraylist_t, these are "length" and "used", respectively

append adds an item to the end of the list
        this increases "used"
        this may also require reallocation, which will increase "length"
                we use realloc, but could also use malloc/memcpy/free

remove pops the last item from the list
        this decreases used
        (this can also fail if used == 0)
        the item that is removed is optionally written to the provided address

insert adds an item between existing items
        - if we insert at index I, then we should see that item at index I
                al_insert(&list, 5, 10);
                // should have list.data[5] == 10
        - we should not overwrite items currently in the list
                the items at indices 5 or more should move one space down

        - but what if we pick an index that is too large?
                - we could signal an error and refuse
                - instead, we pad the list with uninitialized items until we have
                  enough

        - note that we only care about items in the list
                - data in the unused portion of the array does not matter

For strbuf, we also need to track the terminator
        - this makes append/remove/insert a little more complicated

------

File permissions (Unix/Linux specific)

We control access to a file by setting permissions bits

Three kinds of access
        r - read access
        w - write access
        x - execute access (for files) or content listing (for directories)

There are three groups of people we can give permissions to
        u - "user", or user account that "owns" the file
        g - "group", a group of accounts (excluding the owner)
        o - "other", all user accounts that are not the owner or group

3 * 3 = 9 permission bits, in groups of three

ls -l prints the permissions in the order

        rwxrwxrwx
        ---        user permissions (owner)
       ---     group permissions
          ---  other permissions (global)

Because these are in groups of 3, it is common to use octal to specify

        4 -> 100 -> r--  read-only
        6 -> 110 -> rw-  read/write
        2 -> 010 -> -w-  write-only?
        7 -> 111 -> rwx  read/write/execute

        600  -> rw-------  user can read/write, no one else can read or write

Note: these are permissions for a file on disk
They are not directly related to O_RDONLY or O_WRONLY
        -> those specify what operations the file descriptor supports

The permissions that a file has on disk restrict what modes you can open
        a file in


Shell command: chmod
        changes the permissions for a file

        chmod 644 some_file
                change permissions to rw-r--r--

        chmod u+rw some_file
                set the user read and user write bits to 1

Shell command: chown
        change the owner (or group) for a file
        (may be root-only)


When creating a file, we must specify the permissions to give it
        this is the third argument to open
                -> you must provide a mode when opening a file with O_CREAT
                -> the compiler might not catch mistakes here, so be careful

Note that we can create a file in write-mode without giving the user
        write permission!
But we cannot open an existing file in write-mode if the user does not
        have write permission


O_TRUNC  -> if the file exists, truncate its length to 0
O_APPEND -> starts writing from the end of the file

-> if we don't use O_TRUNC or O_APPEND, then we will start writing
        from the beginning of the file and overwrite the existing data, but
        leave data we don't overwrite in place


A common combination O_WRONLY|O_TRUNC|O_CREAT
        open the file in write mode
        delete its contents if it already exists
        create it if it does not exist

Another O_WRONLY|O_CREAT|O_EXCL
        open the file in write mode
        create it if it does not exist
        fail if it does exist (ie., return -1 and set errno)

Class business
--------------
        2 extra days for assignment 1
        project 1 is coming soon
```