**214 Midterm 1 Study Guide**

shell commands gone over in lecture

* More:
* Less:
* Pipe: (|)
* Cat:
* Redirect: (>)
* Apend: (>>)
* Grep:

pointers:

* size
  + 4 bytes on the iLab machines
* Void
  + What is a void pointer?
    - A non-typed (type-neutral) pointer that can be assigned to any type
* to structured types
* with const
* & and \* operators
* incrementing/addition
* to functions
* local vs global scope (i.e. stack vs heap pointers)

Enums:

* an alias for a series of ints (enumerated types, where every value is listed in the type)
* size of an int

Structs:

* collection of types under 1 name
* size of a struct >= sum of the size of the data types in the struct

Unions:

* Single name assigned to a collection of types that assign to the same memory
* Size of a union == the size of the largest data type in the union
  + Because there is only 1 block of memory allocated for it, changing one thing changes all of it

Malloc() operation & use

* Ask system for a certain number of bytes of space to be used, then returned with free()
* Be careful to see if it returned NULL
  + If it did return NULL
* What can you not do with free?
  + Free ptrs that are not dynamically allocated
    - Int a = 0; ptr = &a;
  + Free local/static variables (i.e. not dynamically allocated, like int or char)

size of data types (on iLab machines)

* char: 1 byte
* int: 4 bytes
* ptr: 4 bytes
* double: 8 bytes

pre/post increment

* a = x++
  + a will be the old value of x and x will be incremented
* a = ++x
  + a will be the next x incremented value and x will the new incremented value

C strings

(hexadecimal)

* String in c is a character array with a null terminator at the end (‘\0’)
* Strings can be assigned statically as char[] or char\*, and can also be dynamically allocated for with char\*
* \*\*\* Need hex part

non-blocking IO:

* Calls
* Operation
* read/write loops

i-nodes:

* as a file abstraction
* as an extensible data type

FDE loop:

* Fetch: next instruction is fetched from main memory at the address held in the program counter (PC)
* Decode: Do things necessary to make the loaded instruction runnable
* Execute: ALU does operation to operands…and…program counter (PC) is updated

Stages of Compilation

* Preprocessing: replaces preprocessor macros (#include, #define, etc.)
* Compiler: translates code to assembly code
* Assembler: translates assembly code to machine code
* Linking: redirects and points to library code