Linux command:

- Man: show the manual of a command
- ls: list files inside a folder
- cat: print the content of a file

Andrew file system command:

- Cd: change directory
- Mkdir: create a new directory
- Rm: remove files
- Rmdir: remove empty directory
- Cp: copy
- Scp: copy files between machines

GCC command: gcc -Wall -m32 -std=gnu99 -o output output.c

- Wall: print all warning to terminal
- M32: 32-bit machine
- std=gnu99: compiler version
- -o output: name of the executable file
- Output.c: c file

Compiling stages:

- Preprocessing: remove comments, expand macros (-E flag and .i file)
- Compiling: Translate code into assembly language (-S flag and .s file)
- Assembling: Convert assembly code into machine code (-C flag and .o file)
- Linking: Link .o file with other predefined object file

Data Representation:

- Little endian: lowest bit in the lowest address
- Big endian: highest bit in the lowest address

Pointers and Arrays:

- They are similar but not the same
- Arrays are located in stack. Syntax: a[]
- Array store address of the first element but cannot be used as a variable
- Pointers store the address to pointees. Syntax: int *a or char *a
- Pointer can point to array

Address arithmetic:

- Let's consider int a[5], *b = a;
- Then, b[0 = *b = *(b + 0) = *(a + 0) = *a = a[0]
- Index i > 0: b[i] = *(b + i) = *(a + i) = a[i]
- If b = a + x then b - a = x

Heap allocation:

- malloc(size_t n): allocate n bytes in heap
- calloc(size_t n, size_t size): allocated block with size n * size and initialize them to 0
- realloc(void *ptr, size_t n): reallocate the block of ptr to size n
- free(void *ptr): remove all of the memory in heap. Do nothing if ptr == NULL

Memory of a process: From bottom to top:

- CODE: stores program code in binary-machine code
    - .text: machine code

    - .rodata: string literal
    - Only store read-only data
    - Lifetime: entire program's execution time
    - Initialized by Loader from executable object file
- DATA: stores global and static variables
    - .data: variables initialized to non-zero values
    - .bss: uninitialized or initialized to zero variables
    - Read/write segment
    - Lifetime: entire program's execution time
    - Initialized by Loader from executable object file
- HEAP: Free Store
    - Contains memory allocated and freed by programmer
    - Read/write segment
    - Lifetime: Managed by programmer
    - Initialization: None by default
- STACK: Auto Store
    - Memory is allocated and freed automatically
    - Read/write segment
    - Lifetime: from declaration to end of scope
    - Initialization: None by default

File IO:

- fgets(char *, size, stream)
- fputs(char *, stream)
- fscanf(stream formatted string, variables…)
- fprintf(stream, formatted string. variables…)
- FILE *fopen(const char *filename, const char *mode)
- Int fclose(const char *filename)