ECE 282 Lab 4

# Command practice (Include the following questions/tasks in the report)

Answer to the following questions are required in your lab report:

What do the following commands do?

1. date - returns the current date as a character vector in the format dd-MMM-yyyy.
2. touch - used to update the access date and/or modification date of a computer file or directory.
3. wc - prints newline, word and byte counts for files.
4. What is the meaning of each output of the command ‘wc’?  
   **Hint: Use manpage if you are unsure.**

The first column shows number of lines present in a file specified, the second column shows number of words present in the file, the third column shows the number of characters present in the file, and the fourth column itself is the file name which are given as argument.

Write down the commands that can do the following tasks:

1. Find out how many lines are there in the output of ‘man –k read’  
   **Hint: ‘wc’ and pipeline (|)**man -k read | wc -l
2. Find out how many words are there in the file ‘dirent.h’  
   **Hint 1: The file is located at /usr/include/dirent.h  
   Hint 2: Use ‘cat’ (or ‘more’, ‘less’) and pipeline to ‘wc’**cat/usr/include/dirent.h | wc -l
3. Change the mode (the permission of a file)  
   **Hint: Search the manual for “change mode”**

chmod

# Make file

You know how to compile many sources file into a single executable. When you change the code of any source file, you will need to recompile it into an object file, then link all object files again. Can we create some utility that can compile the whole thing, so we don’t have to type a bunch of **gcc** again and again? Yes, it is **make file**.

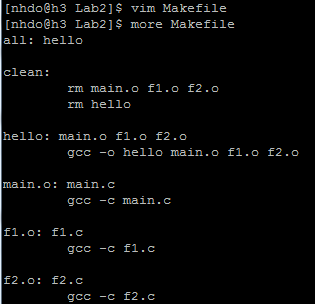
Here is a tutorial for you: <https://www.gnu.org/software/make/manual/html_node/Introduction.html>

Let us use the source files from last lab **(Lab3)**

Before, we need to do all the following to compile all the sources into **hello.** Pay attention to how these commands transfer to the makefile.

|  |  |
| --- | --- |
| **gcc –c main.c** | Create main.o (main.o depends on main.c) |
| **gcc –c f1.c** | Create f1.o (f1.o depends on f1.c) |
| **gcc –c f2.c** | Create f2.o (f2.o depends on f2.c) |
| **gcc –o hello main.o f1.o f2.o** | Create hello (hello depends on main.o, f1.o, and f2.o) |

Now with make file, we can just type “make”. Here is my makefile. You can create it using any text editor and name it ‘**Makefile’**. Put your makefile in the same folder with the source code files.



# Play with masking (include the following questions/tasks in the report)

Answer to the following questions are required in your lab report:

1. In **struct stat**, the variable that contains the file size is **st\_size.** What is the variable that contains the file permission (protection)?  
   mode\_t st\_mode;
2. The flag defined for the **st\_mode** field for a **‘directory’** is S\_IFDIR. What is its bitmask value?  
   In Binary: 100000000000000  
   In Octal: 0040000
3. What is the bit mask to check for “owner has read permission”?  
   In Binary: 100000000  
   In Octal: 00400
4. What is the bit mask to check if a file has all the following permissions  
   *owner can read, write; group can read only; others cannot read/write/execute.*  
   In Binary: 100000000 <- owner can read, 10000000 <- owner can write, 100000 <- group can read, 1000000 <- owner can execute  
   In Octal: 00400 <- owned can read, 00200 <- owner can write, 00040 <- group can read, 00100 <- owner can execute

**Hint:  
Remember st\_mode is 16 bits…  
Search the manual: man –k file  
Opps, too many results, search again: man –k file | grep status  
See that ‘stat’ is the function we need.  
Read the manual: man 2 stat, you can find all answers here.**

# Can you LS?

From Chapter 3 in the text:

3.11 The ls2 bug Modify ls2. c so it works correctly when the name of a directory is given as a

command-line argument.

The new file name is **ls3.c**. The program must be modular, i.e. use separated functions to carry out the required functionalities. Furthermore, each function must be written in a separate .c file. Header files must be used and included in the header file to define the functions. Finally, write a makefile to compile the whole project.

Include your ls3.c and all other .c and .h files as well as the word doc or pdf report file in a .zip file names lab04.zip. Then submit the file to Canvas. Also, **leave a copy** of the files (except .zip) on your home directory under folder **lab04**

In the lab report, tell me why software engineers prefer using macro and flag (like S\_IXUSR), instead of the octal bit mask directly? It is easier and quicker to comprehend what macro and flag means instead of the octal bit mask directly.