Math 4610 Fundamentals of Computational Mathematics - Topic 5.

This topic covers information on how to work once a terminal is up and running on your computer. A Linux/Unix operating system should be running in the terminal. In most real computational settings, it is important to be able to work using a command line to create files, modify these files, compile code, and a number of other tasks. You can invoke multiple terminals and work on multiple files at the same time. Your instructor has three or four terminals open at any given time. One to edit files, one to compile and execute code, and the third to display results. The homework and projects assigned in the class will typically require the use of a terminal to complete the work.
We will eventually use High Performance Computing (HPC) resources at the Center for High Performance Computing (CHPC) at the University of Utah to work on a project or two. The access we will have will be through the terminals or terminal emulators running Linux/Unix operating systems. So, it is important to be familiar with at least a few basic Linux/Unix commands. In this section of the notes we will work through a few of the more important commands needed. We will also introduce more commands with options as we work through the semester.

> > go there (pdf)

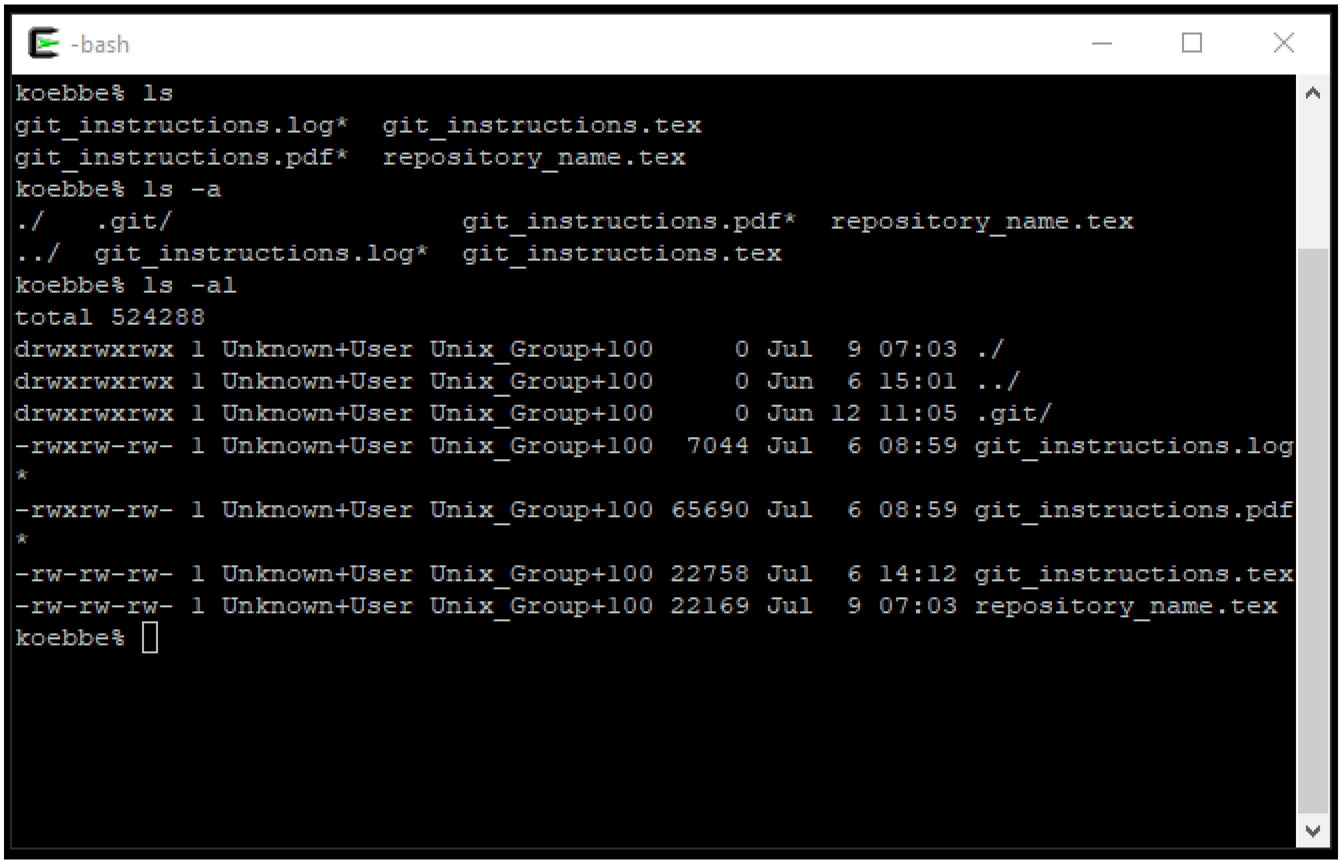
The content for each lecture in Math 4610 will be presented in a lecture format that will follow an outline presented at the beginning of each class period. For the first lecture, the following list of items will be covered:

Linux/Unix Primer for Math 4610 at USU: Just Some Basic Commands

Cygwin Primer for Math 4610 at USU: List the Contents of the Home Folder

The first thing to look at is what is in a directory. It is important to know where you are at in a directory and the like. This also serves as a first linux or unix command. In the screenshot below there are a couple of versions of a command that will list files and folders with more or less information. Note that most linux commands look like the following:

```
% command [options] [input parameters]
```



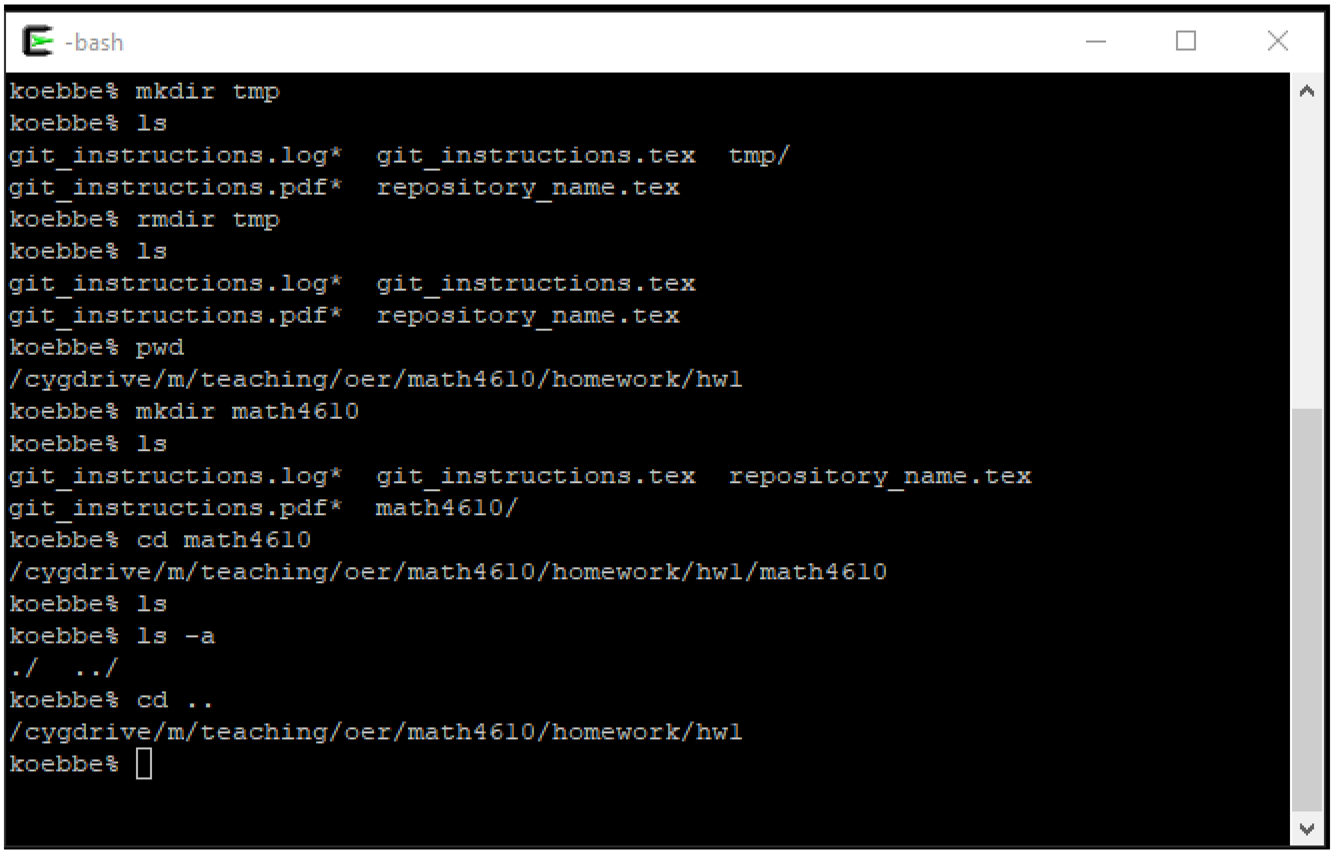
```
koebbe% ls
git_instructions.log*  git_instructions.tex
git_instructions.pdf*  repository_name.tex
koebbe% ls -a
./   .git/                  git_instructions.pdf*  repository_name.tex
../  git_instructions.log*  git_instructions.tex
koebbe% ls -al
total 524288
drwxrwxrwx 1 Unknown+User Unix_Group+100     0 Jul  9 07:03 ./
drwxrwxrwx 1 Unknown+User Unix_Group+100     0 Jun  6 15:01 ../
drwxrwxrwx 1 Unknown+User Unix_Group+100     0 Jun 12 11:05 .git/
-rwxrw-rw- 1 Unknown+User Unix_Group+100  7044 Jul  6 08:59 git_instructions.log
*
-rwxrw-rw- 1 Unknown+User Unix_Group+100 65690 Jul  6 08:59 git_instructions.pdf
*
-rw-rw-rw- 1 Unknown+User Unix_Group+100 22758 Jul  6 14:12 git_instructions.tex
-rw-rw-rw- 1 Unknown+User Unix_Group+100 22169 Jul  9 07:03 repository_name.tex
koebbe% 
```

Figure 1: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

Cygwin Primer for Math 4610 at USU: Directory Commands

You will need to create, move, remove, and other things to directories to keep work organized. The **mkdir** command allows a persion to create a new directory in the current working directory. This is the same thing tha Windows Explorer allows you to do with a popup menu. There will be many places where a directory structure will be required. You can remove a directory with the **rmdir** command. The **cd** command can be used to navigate through a directory structure. Finally, on this screen capture, the **pwd** command is used to determine the current working directory. This can be used to figure out where you are in a directory structure.

```
% pwd                   current working directory
% cd                    change working directory
% mkdir                 make a new directory
% rmdir                 remove an existing directory
```



```
koebbe% mkdir tmp
koebbe% ls
git_instructions.log*  git_instructions.tex  tmp/
git_instructions.pdf*  repository_name.tex
koebbe% rmdir tmp
koebbe% ls
git_instructions.log*  git_instructions.tex
git_instructions.pdf*  repository_name.tex
koebbe% pwd
/cygdrive/m/teaching/oer/math4610/homework/hw1
koebbe% mkdir math4610
koebbe% ls
git_instructions.log*  git_instructions.tex  repository_name.tex
git_instructions.pdf*  math4610/
koebbe% cd math4610
/cygdrive/m/teaching/oer/math4610/homework/hw1/math4610
koebbe% ls
koebbe% ls -a
./  ../
koebbe% cd ..
/cygdrive/m/teaching/oer/math4610/homework/hw1
koebbe% 
```

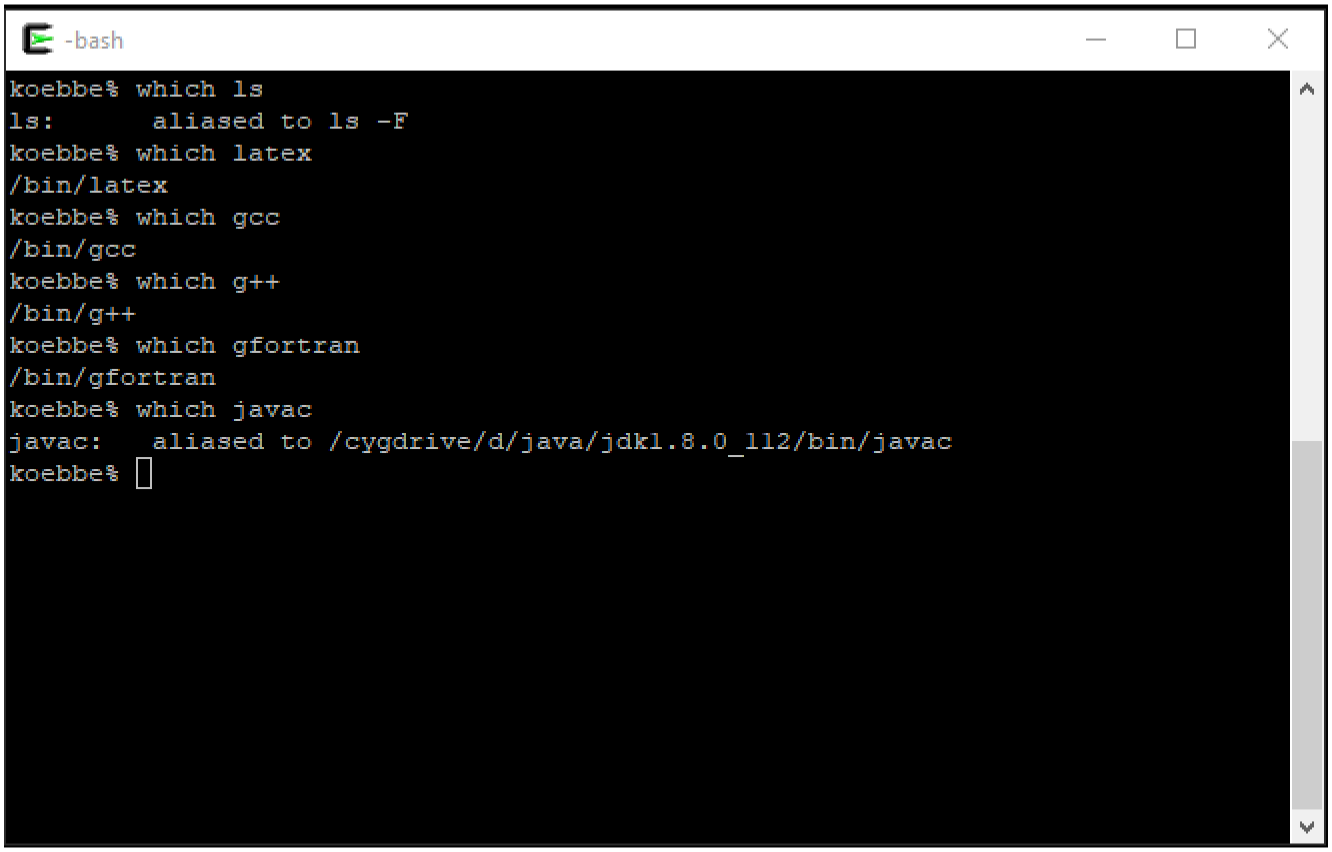
Figure 2: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

---

Cygwin Primer for Math 4610 at USU: Which Command

You will want to know what is available for doing work within Cygwin or any other platform. The which command will let you know if apps or other executables are available on your version of Cygwin. In particular, it is important to know if certain compilers (e.g, javac, gcc, f77) are available. A significant number of tasks you will be asked to complete will require the use of a compiler and Cygwin has a number of (good) standard compilers for C, C++, and fortran. The syntax for the command is the following.

```
% which command
```



```
koebbe% which ls
ls:      aliased to ls -F
koebbe% which latex
/bin/latex
koebbe% which gcc
/bin/gcc
koebbe% which g++
/bin/g++
koebbe% which gfortran
/bin/gfortran
koebbe% which javac
javac:   aliased to /cygdrive/d/java/jdk1.8.0_112/bin/javac
koebbe% 
```

Figure 3: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

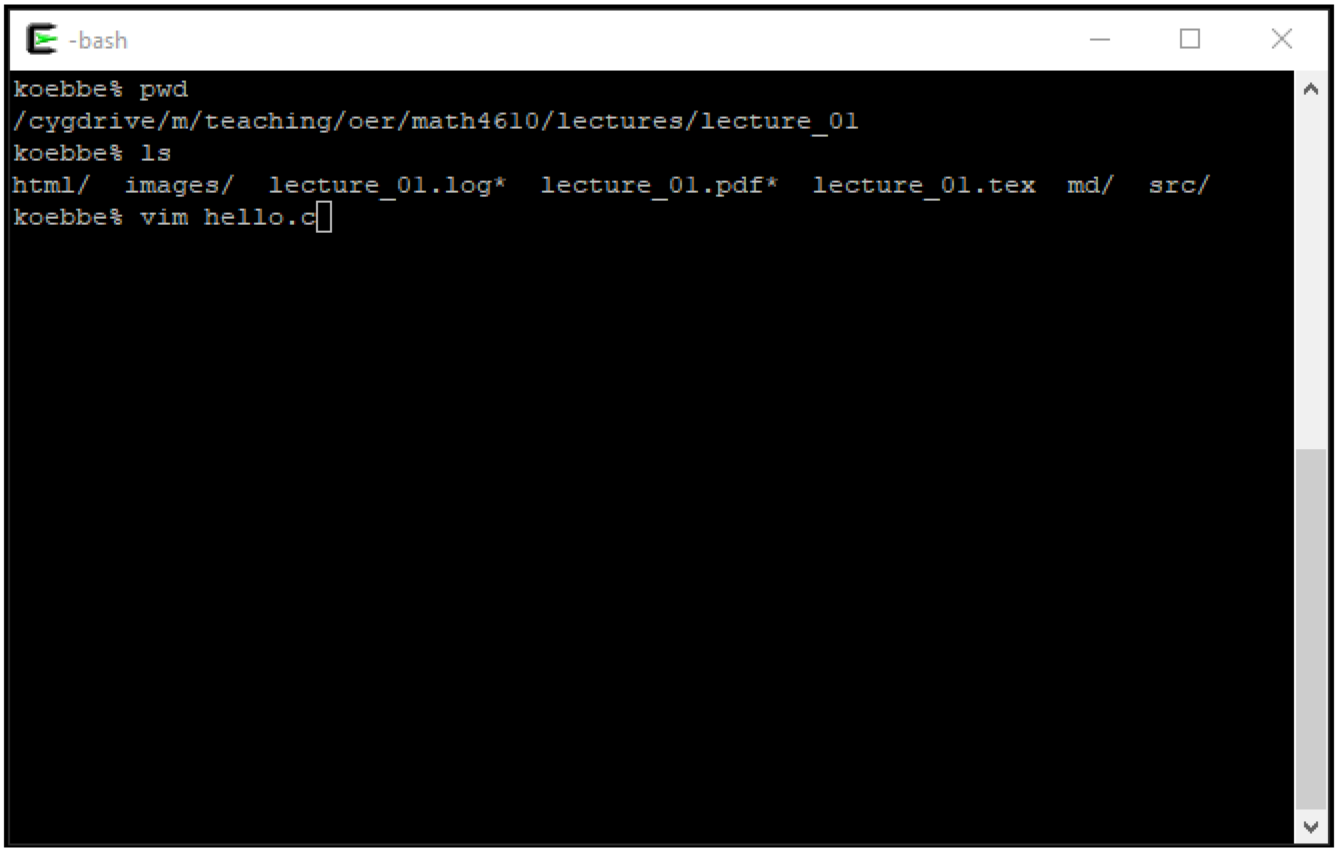
---

Cygwin Primer for Math 4610 at USU: A Simple Editing Program

You will need an editor to create text files. There are a number of editors that can be downloaded and used in any Cygwin installation. The standard editor that is always available for linux and unix boxes is 'vi'Ṫhis editor is a bit rudimentary, but works. Another editor which will be used in by the instructor in the course is 'vim'. The syntax for starting the editor in a window is the following.

```
% vim filename
```

There are a lot of escape sequences it insert text, write a file and so on. If you are new to vim, you will need to learn at least a few of these editing commands.



```
koebbe% pwd
/cygdrive/m/teaching/oer/math4610/lectures/lecture_01
koebbe% ls
html/  images/  lecture_01.log*  lecture_01.pdf*  lecture_01.tex  md/  src/
koebbe% vim hello.c
```

Figure 4: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

---

Cygwin Primer for Math 4610 at USU: First View of the vim Editor

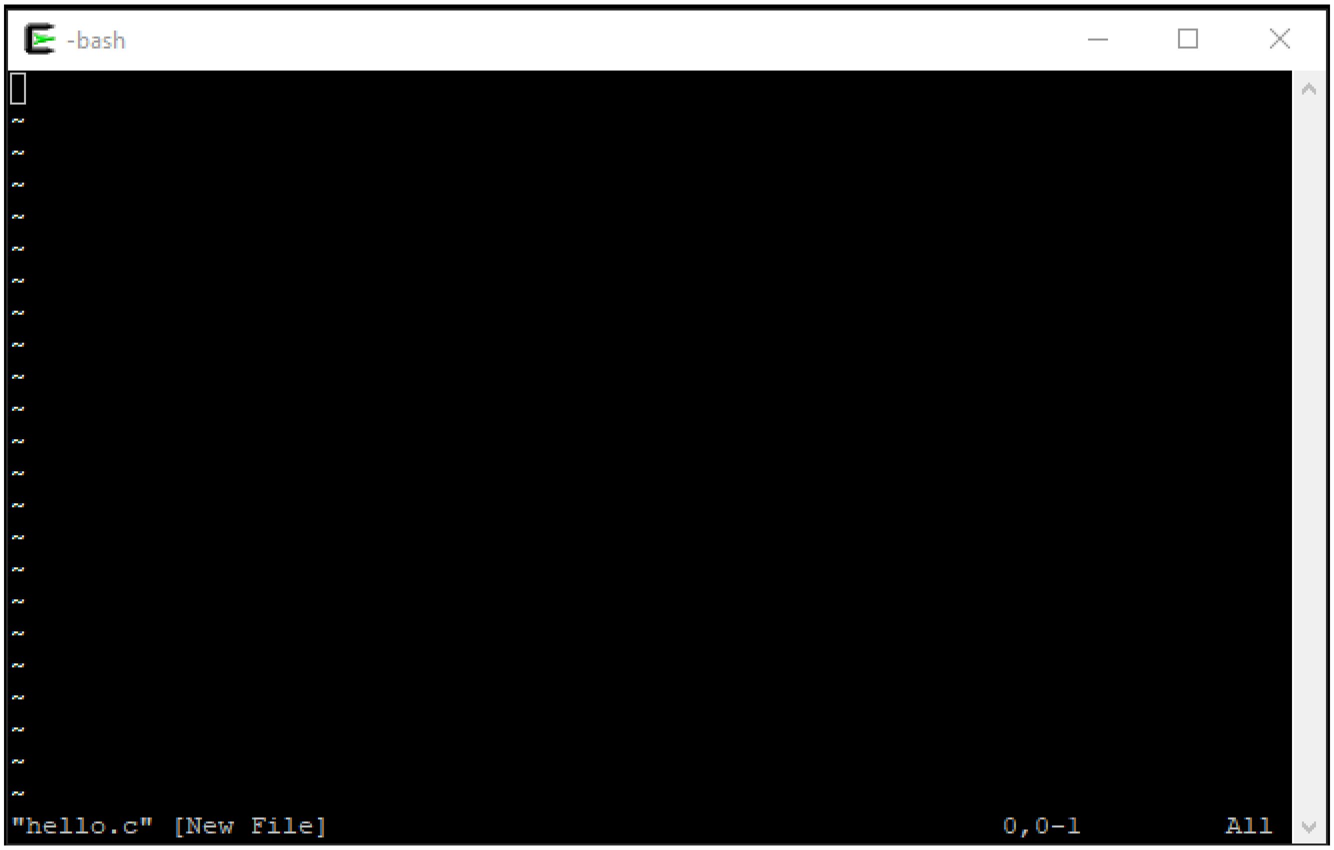
---

Below is what the terminal will turn into when you start up the editor on a new file. To get out of the editor, you can use any of the following commands inside the editor.

```
    :x                  write and exit the editor - saves changes in the file
    :q                  exit the editor if no changes have been made
    :x!                 force a write and exit the editor - saves the file
    :q!                 force an exit of the editor - no changes are saved
```

Note that there are a few other commands that can be used to save changes. For example

```
    :w                  write and stay in the editor
    :w!                 force a write and stay in the editor
```



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"hello.c" [New File]                                   0,0-1          All
```

Figure 5: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

---

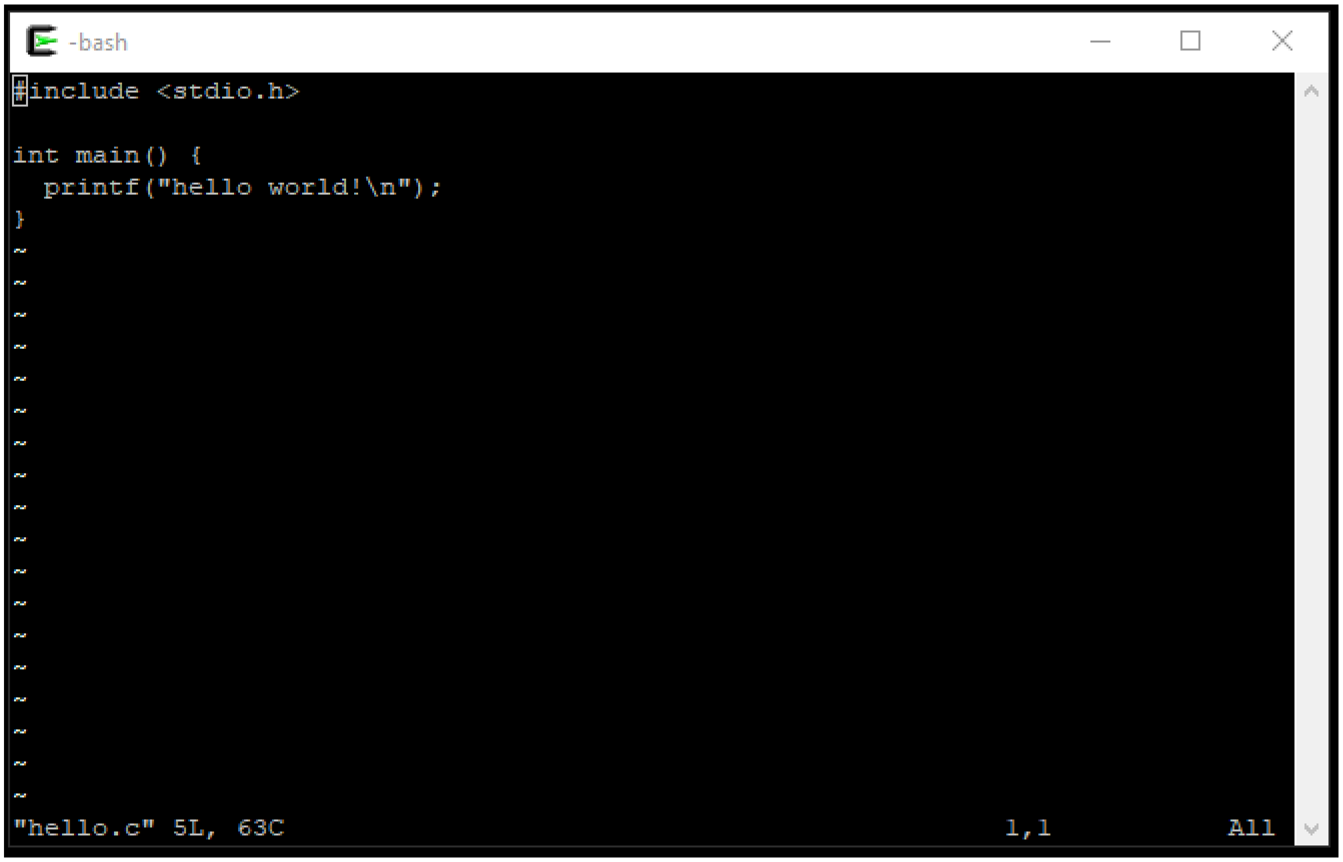
Cygwin Primer for Math 4610 at USU: An Example of a Text File/Program

---

The following screenshot shows a few lines that have been typed into vim that deinfes a standard hello world example for C. To insert/append characters in the text file, you can use the following commands to do this. Note that the commands below do not show up on the screen and the chnages are made where the cursor is currently located.

```
a                      append text at this point in the file
o                      open a line after the current line
O                      open a line before the current line
```

To end adding or inserting text, use the escape character. Again, the commands will not show up on screen. Learning everything about vi or vim is a time consuming process. It is one of those things that you figure out as you go.



```
#include <stdio.h>

int main() {
  printf("hello world!\n");
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"hello.c" 5L, 63C                                            1,1           All
```

Figure 6: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

---

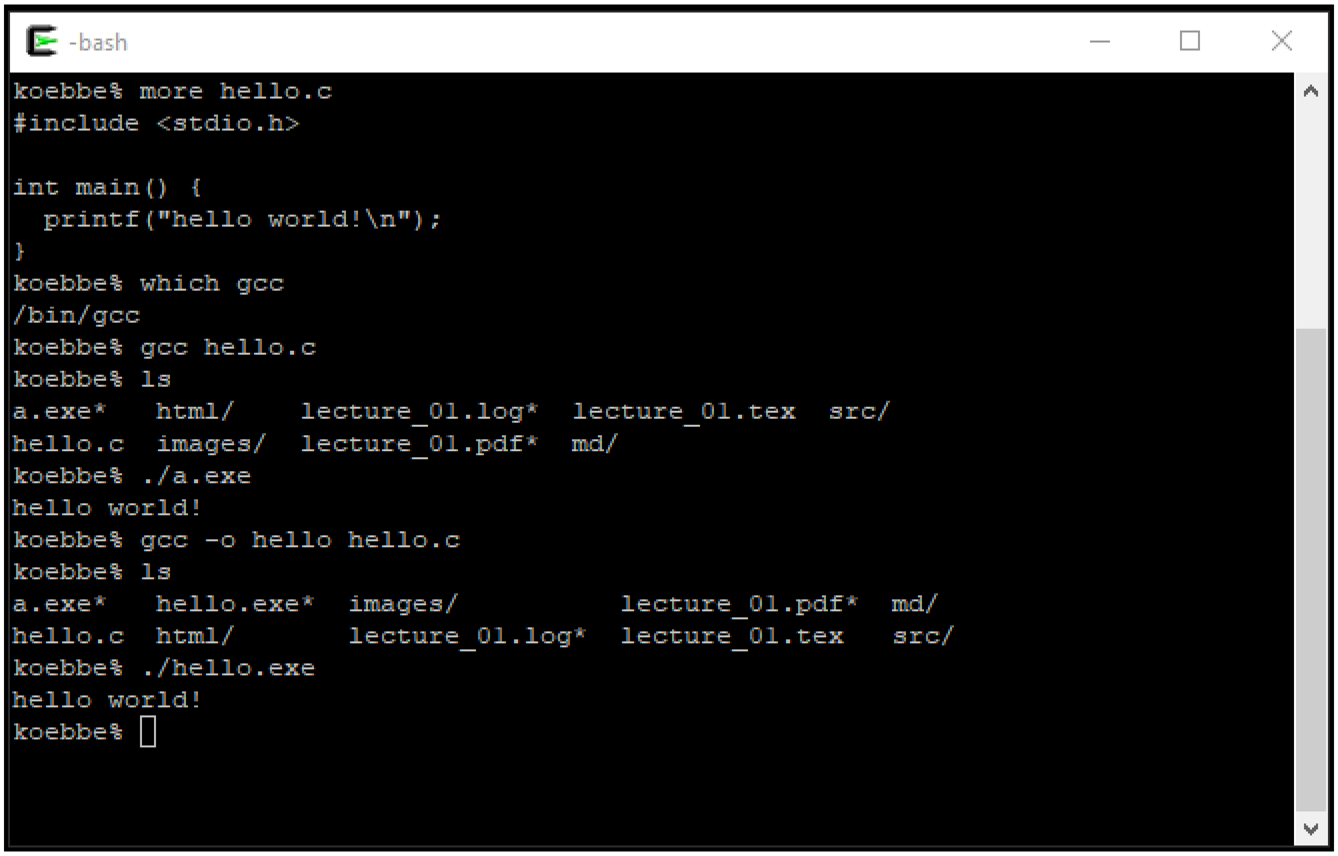
Cygwin Primer for Math 4610 at USU: Compiling a Program

Compiling a program is relatively easy at this point in time. To compile the program on the previous page you should type in

```
% gcc hello.c
```

The result is an exeutable as seen below. If you want to name the executable something besides ‘a’ then type the following.

```
% gcc -o hello hello.c
```



```
koebbe% more hello.c
#include <stdio.h>

int main() {
  printf("hello world!\n");
}
koebbe% which gcc
/bin/gcc
koebbe% gcc hello.c
koebbe% ls
a.exe*   html/    lecture_01.log*  lecture_01.tex  src/
hello.c  images/  lecture_01.pdf*  md/
koebbe% ./a.exe
hello world!
koebbe% gcc -o hello hello.c
koebbe% ls
a.exe*   hello.exe*  images/          lecture_01.pdf*  md/
hello.c  html/       lecture_01.log*  lecture_01.tex   src/
koebbe% ./hello.exe
hello world!
koebbe% 
```

Figure 7: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box

---

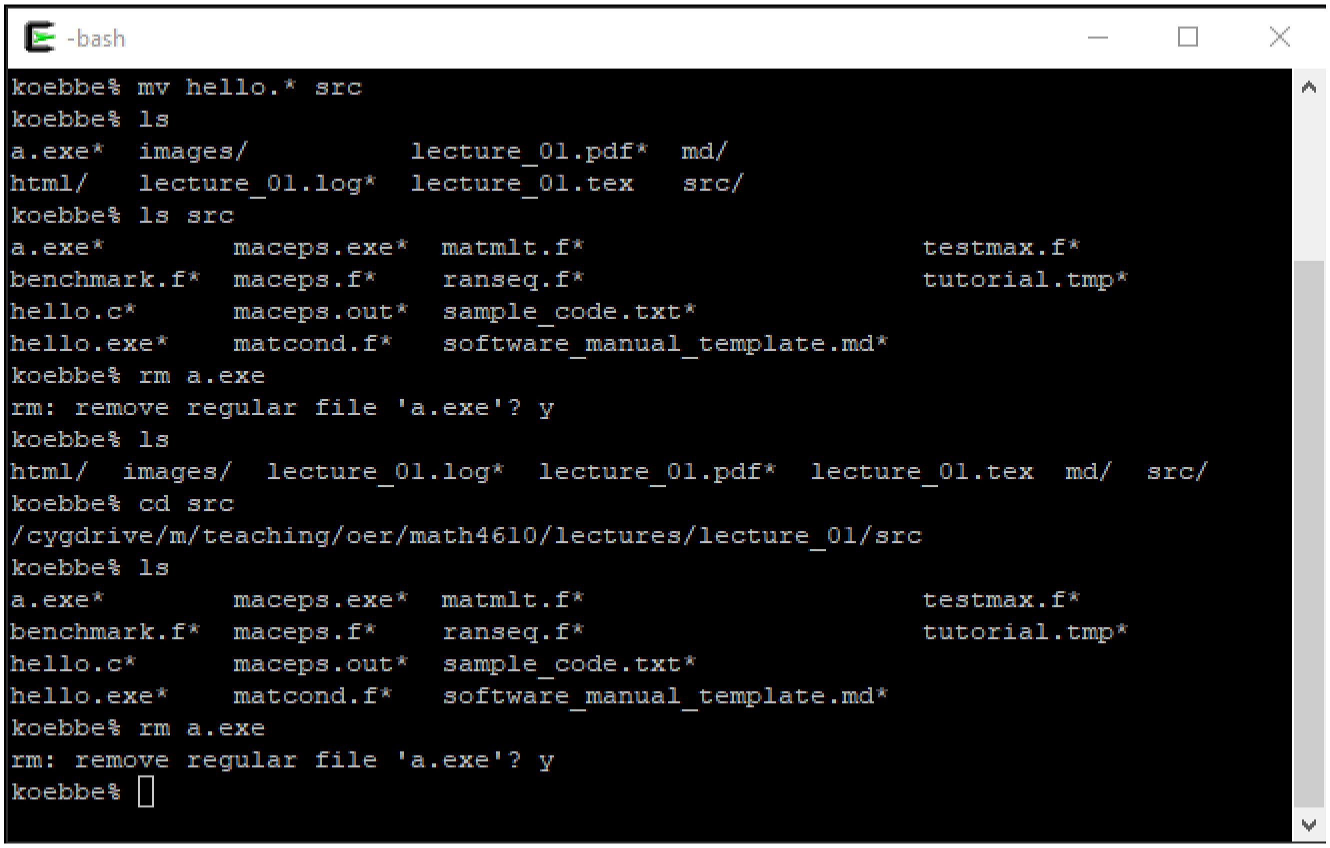
Cygwin Primer for Math 4610 at USU: Keeping Track of Working Code

---

It is a good idea to organize your work within assignments and projects. There is a standard set of folders/directories in linux and unix that most have adopted. Your instructor follows this idea and usually creates a list of folders including /src, /data, /bin, and /doc. When computer literate folks see these folders, they know what is stored in the folders. As an example,

```
% mkdir src
% mkdir bin
```

can be used and then the executable the text file can be put into /src and the binary can be copied into /bin.



```
koebbe% mv hello.* src
koebbe% ls
a.exe*  images/          lecture_01.pdf*  md/
html/   lecture_01.log*  lecture_01.tex   src/
koebbe% ls src
a.exe*         maceps.exe*  matmlt.f*                      testmax.f*
benchmark.f*   maceps.f*    ranseq.f*                      tutorial.tmp*
hello.c*       maceps.out*  sample_code.txt*
hello.exe*     matcond.f*   software_manual_template.md*
koebbe% rm a.exe
rm: remove regular file 'a.exe'? y
koebbe% ls
html/  images/  lecture_01.log*  lecture_01.pdf*  lecture_01.tex  md/  src/
koebbe% cd src
/cygdrive/m/teaching/oer/math4610/lectures/lecture_01/src
koebbe% ls
a.exe*         maceps.exe*  matmlt.f*                      testmax.f*
benchmark.f*   maceps.f*    ranseq.f*                      tutorial.tmp*
hello.c*       maceps.out*  sample_code.txt*
hello.exe*     matcond.f*   software_manual_template.md*
koebbe% rm a.exe
rm: remove regular file 'a.exe'? y
koebbe% 
```

Figure 8: Screenshot taken using **Snip & Sketch**. This is an app on my Windows 10 box