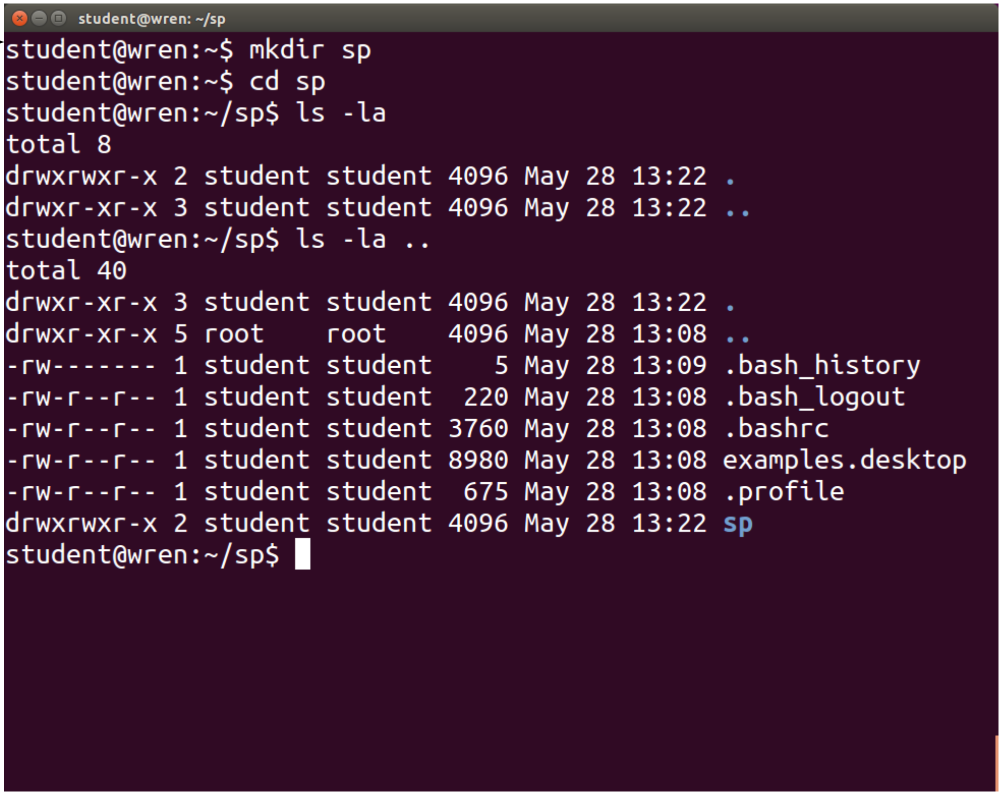
Week 4 Lecture 3

Setting Up

Create Directory for Class

- Create “sp” dir
- Enter sp
- Look at parent
  - Parent contains “sp”



student@wren: ~/sp

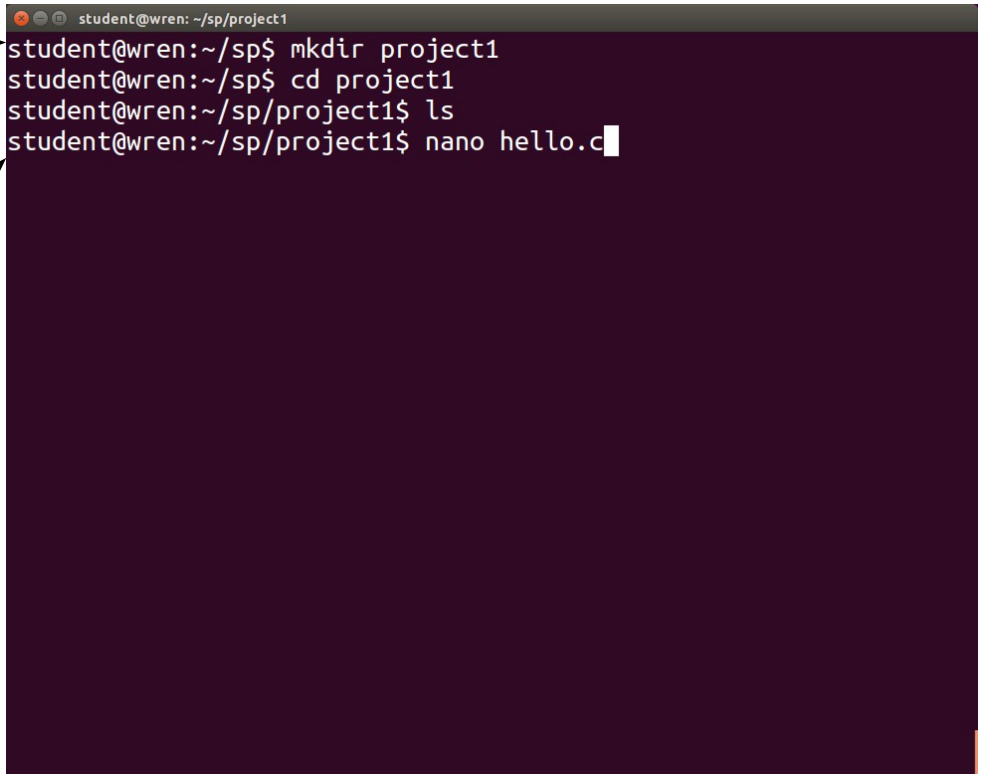
```
student@wren:~$ mkdir sp
student@wren:~$ cd sp
student@wren:~/sp$ ls -la
total 8
drwxrwxr-x 2 student student 4096 May 28 13:22 .
drwxr-xr-x 3 student student 4096 May 28 13:22 ..
student@wren:~/sp$ ls -la ..
total 40
drwxr-xr-x 3 student student 4096 May 28 13:22 .
drwxr-xr-x 5 root    root    4096 May 28 13:08 ..
-rw------- 1 student student    5 May 28 13:09 .bash_history
-rw-r--r-- 1 student student  220 May 28 13:08 .bash_logout
-rw-r--r-- 1 student student 3760 May 28 13:08 .bashrc
-rw-r--r-- 1 student student 8980 May 28 13:08 examples.desktop
-rw-r--r-- 1 student student  675 May 28 13:08 .profile
drwxrwxr-x 2 student student 4096 May 28 13:22 sp
student@wren:~/sp$ 
```

First Program

- Write a program that will print “Hello world” on the screen.
- How will we know it worked?
  - We will see “Hello world” on the screen.
  - Like Scratch, every element is visible
  - Unlike Scratch, the work the computer does to make your work visible must be specified.
    - Luckily someone else has done the hard part.

Create a Project Dir

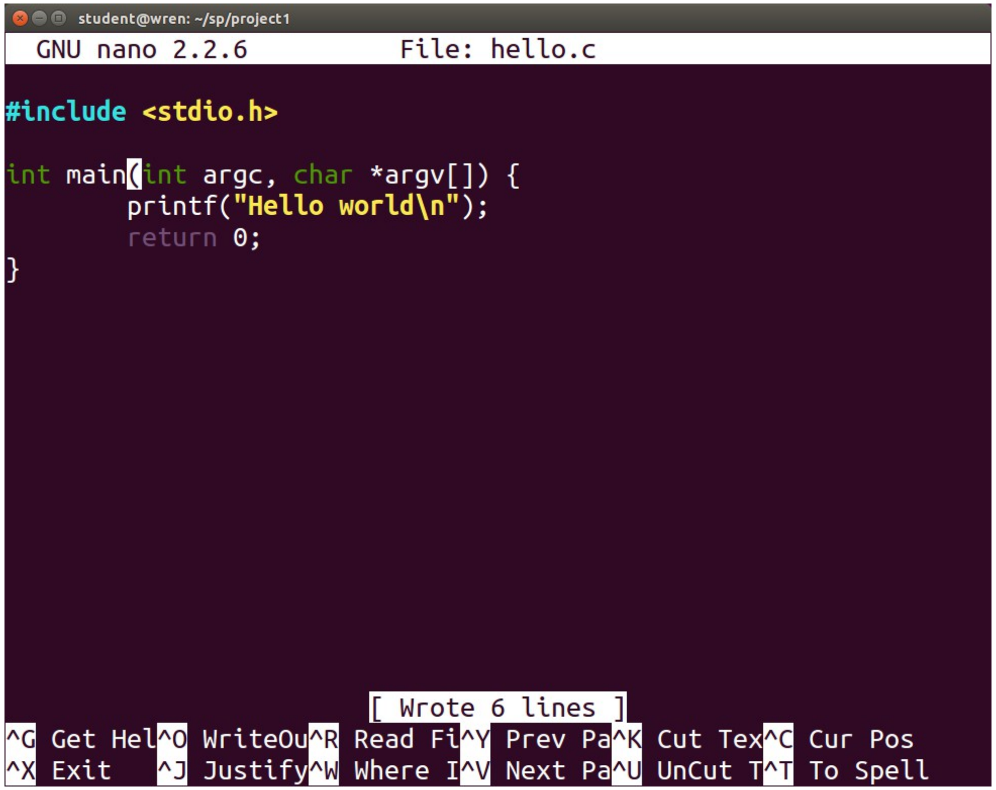
- Create a project
- Move to the project
- Project is empty
- Edit file “hello.c”



```
student@wren:~/sp$ mkdir project1
student@wren:~/sp$ cd project1
student@wren:~/sp/project1$ ls
student@wren:~/sp/project1$ nano hello.c
```

Write your first program

- Nano is a very simple editor
  - But it will help you by coloring text
- Hit Ctrl-o to save the text.
- Hit Ctrl-x to exit the editor



```
student@wren: ~/sp/project1
  GNU nano 2.2.6               File: hello.c

#include <stdio.h>

int main(int argc, char *argv[]) {
        printf("Hello world\n");
        return 0;
}

                        [ Wrote 6 lines ]
^G Get Hel^O WriteOu^R Read Fi^Y Prev Pa^K Cut Tex^C Cur Pos
^X Exit    ^J Justify^W Where I^V Next Pa^U UnCut T^T To Spell
```

The Program

- Define I/O
- Define main function
- Body of main
  - Print out “Hello world”
  - Exit with success code

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello world\n");
    return 0;
}
```

#include <stdio.h>

- Adds the definitions of standard input/output functions
- The function printf is a function; not part of the C language.
- The function is linked to your program when it is compiled
- The definitions allow the compiler to check that the function you are using is one that can be linked in.

int main(int argc, char *argv[])

- This line defined the main function.
  - The main function is special; it is where computer starts executing the program
  - It return an int (i.e., … -2, -1, 0, 1, 2 …)
    - 0 mean success, anything else means failure
  - It takes two parameters (supplied by the shell)
    - int argc: a positive int that tells how man items are in argv
    - char *argv[]: a list of strings that are the program can use

Example int argc, char *argv[]

- ls -la: argc = 1; argv = “-la”
- ls -la sp: argc = 2; argv = “-la”, “sp”

printf(“Hello world/n”);

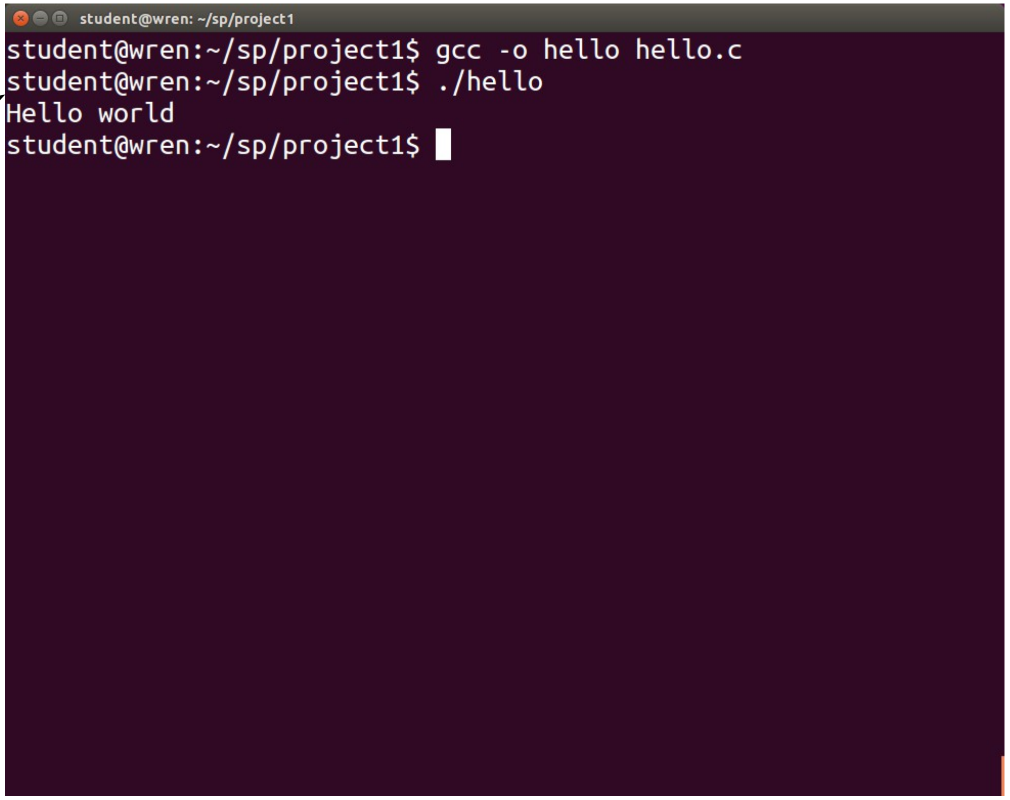
- Argument is the string “Hello world/n”
  - A sequence of characters
  - '/n' is a single character called “newline”
    - '/' is an escape character that changes the meaning of the following character
- The semicolon ';' terminates a statement
- A statement is something the computer can do
  - Here the statement is a function

return 0;

- The main function declares that it will return and int, this statement does it.
- 0 represents success

Compile and Run hello.c

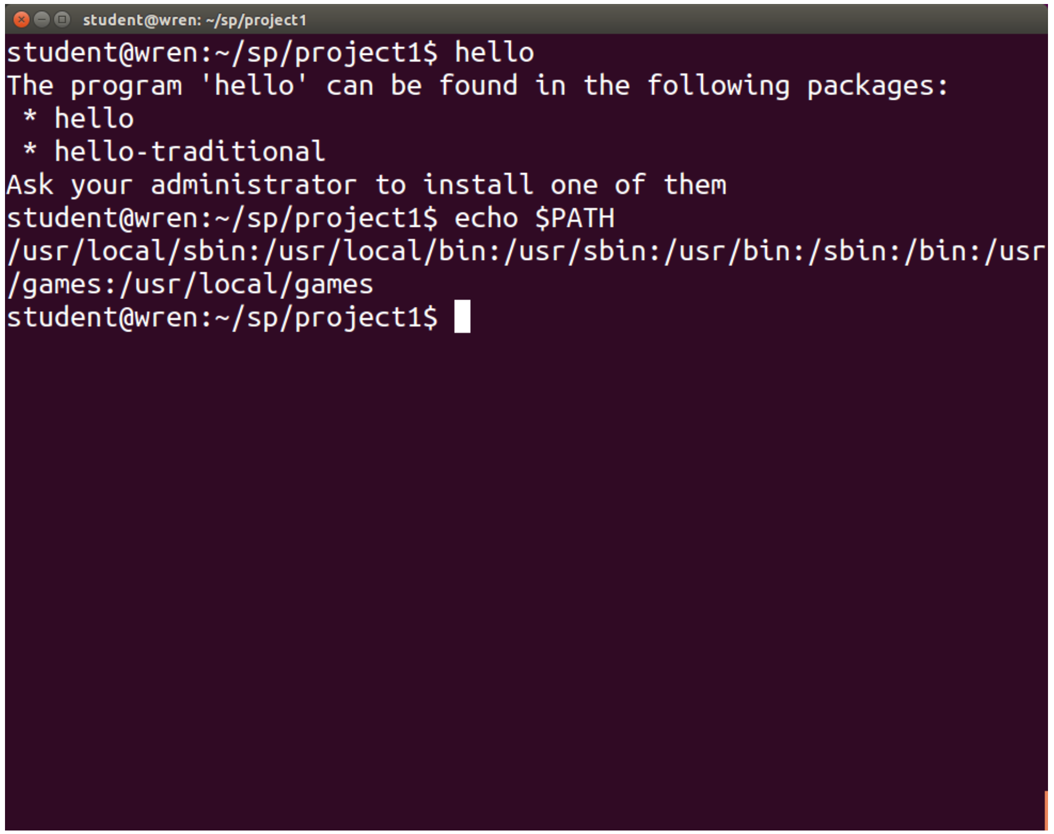
- Compile
  - "-o hello" means executable file is "hello"
- Run
  - ./hello means run from current directory
  - Otherwise it would not be found



```
student@wren:~/sp/project1$ gcc -o hello hello.c
student@wren:~/sp/project1$ ./hello
Hello world
student@wren:~/sp/project1$ 
```

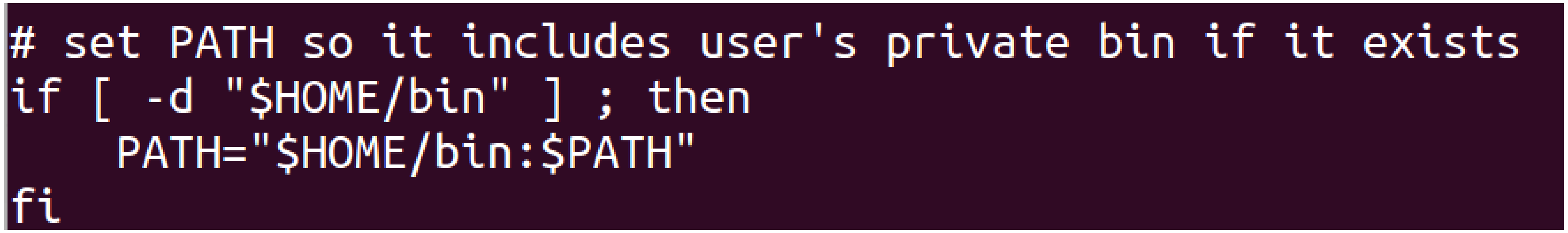
Naive execution

- Plain “hello” fails
- “$PATH” contains a list of directories can have commands in them



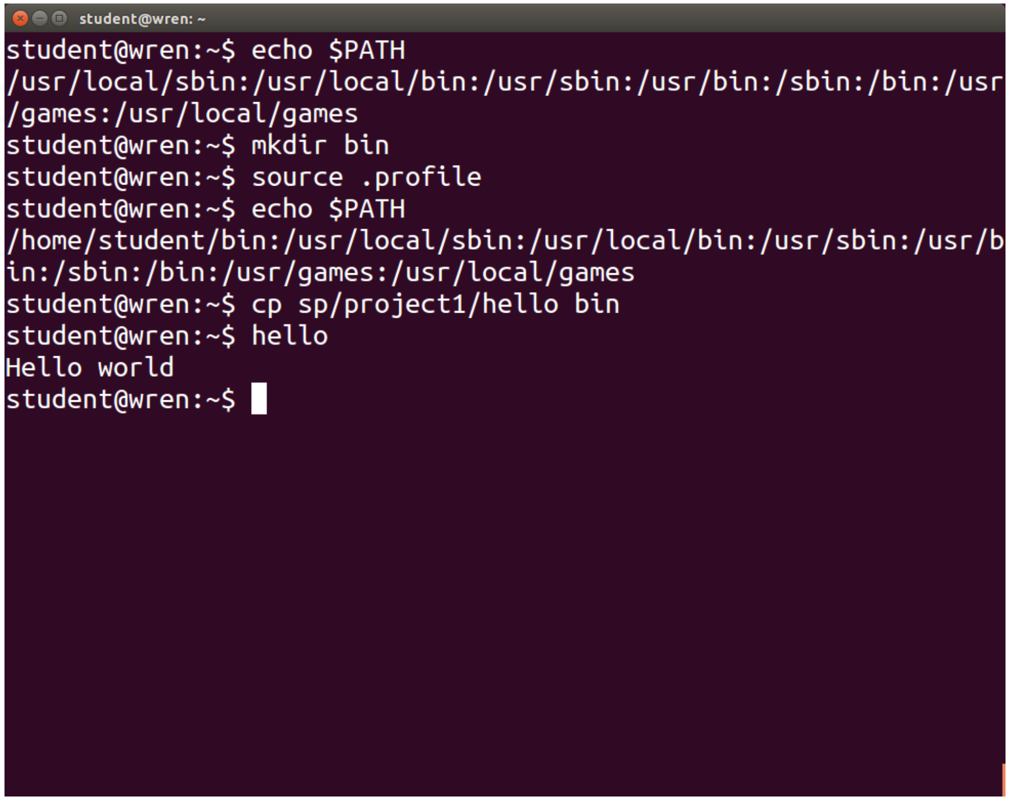
```
student@wren:~/sp/project1$ hello
The program 'hello' can be found in the following packages:
 * hello
 * hello-traditional
Ask your administrator to install one of them
student@wren:~/sp/project1$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games
student@wren:~/sp/project1$ 
```

Add new directory to $PATH



```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

- .profile adds “~/bin” to path
- Create ~/bin
- Source .profile to set path
  - Now “/home/student/bin” is on path
  - Now “hello” works anywhere



```
student@wren:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/games:/usr/local/games
student@wren:~$ mkdir bin
student@wren:~$ source .profile
student@wren:~$ echo $PATH
/home/student/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/sbin:/bin:/usr/games:/usr/local/games
student@wren:~$ cp sp/project1/hello bin
student@wren:~$ hello
Hello world
student@wren:~$ █
```