C introduction part 5

header and make files

https://sieprog.ch/#h
https://sieprog.ch/#make
https://sieprog.ch/#creer-bibliotheque
https://sieprog.ch/#inclure-bibliotheque

Note: make files may be used for the project, but is not covered in final exam

Objectives

- Long code files are difficult to read. Need to break up into multiple files.

- → learn to organize larger programs into a collection of code files (grouped by theme or common tasks)

Increasing complexity

1. #include .c files
2. #include .h and .c files
3. Makefiles

```
.
├── driver.c
└── helloworld.c
```

Contents of "driver.c"

```c
#include "helloworld.c"

int main() {

  myPrintHelloWorld();

  return(0);
}
```

Contents of "helloworld.c"

```c
#include <stdio.h>

void myPrintHelloWorld(void) {

  printf("Hello World\n");

  return;
}
```

Compile and run with

```
gcc -Wall -o driver driver.c
./driver
```

Header files

.h files:

- Declaration of structures
- Function prototypes (declaration of name and type signature)

.c files:

- #include statement to libraries
- #include statement to corresponding .h file
- Function definitions

```
.
├── driver.c
├── helloworld.c
└── helloworld.h
```

Contents of "driver.c"

```c
#include "helloworld.h"

int main() {

  myPrintHelloWorld();

  return(0);
}
```

Contents of "helloworld.h"

```c
void myPrintHelloWorld(void);
```

Contents of "helloworld.c"

```c
#include <stdio.h>
#include "helloworld.h"

void myPrintHelloWorld(void) {

  printf("Hello World\n");

  return;
}
```

```
gcc -Wall -o driver driver.c helloworld.c
./driver
```

or, compile each module separately

```
gcc -Wall -c driver.c                   # -> creates driver.o
gcc -Wall -c helloworld.c                       # -> creates helloworld.o
gcc -Wall -o driver driver.o helloworld.o       # -> creates driver
./driver
```

make

Motivation.

```
gcc -Wall -c driver.c                    # -> creates driver.o
gcc -Wall -c helloworld.c                      # -> creates helloworld.o
gcc -Wall -o driver driver.o helloworld.o      # -> creates driver
./driver
```

What happens when there are many .c files?
Do we need to recompile files that have not changed?

→ use program called *make*

Setting up and running Makefile with *make*

"Makefile"

```
CC=gcc
CFLAGS=-Wall
DEPS = helloworld.h
OBJ = driver.o helloworld.o

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

driver: $(OBJ)
	$(CC) -o $@ $^ $(CFLAGS)

.PHONY: clean

clean:
	rm -f *.o *~
```

%: wildcard character
$@: name of the target (what is on the left slide of :)
$<: name of prerequisite file (source file - i.e., the .c file)
$^: name of all prerequisite files written with space in between them

Execution:

```
make
```

looks for "Makefile" in same directory

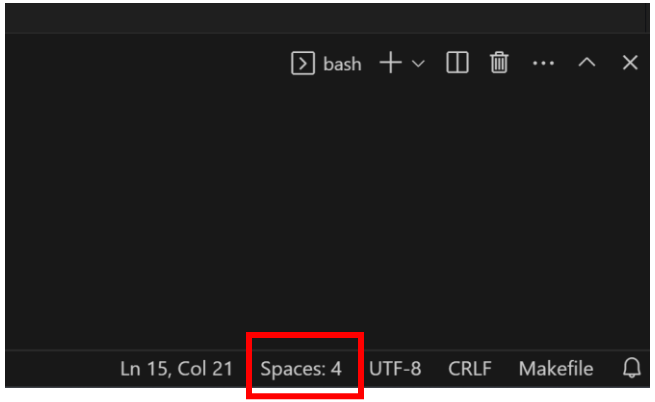
This translates to running

```
gcc -c -o driver.o driver.c -Wall          # from $(CC) -c -o $@ $< $(CFLAGS)
gcc -c -o helloworld.o helloworld.c -Wall # from $(CC) -c -o $@ $< $(CFLAGS)
gcc -o driver driver.o helloworld.o -Wall # from $(CC) -o $@ $^ $(CFLAGS)
```

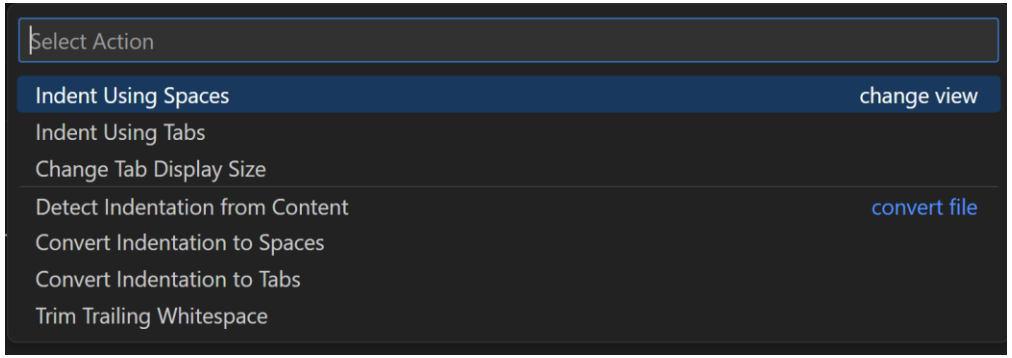
Editing Makefile in VS Code

1. Create Makefile (no extension – e.g., not “Makefile.mak” – just “Makefile”)

2. lower left corner, click on Spaces



3. Select Indent Using Tabs, pick 4 spaces



4. Use TABS for indentation rather than spaces

5. Run *make* in terminal

Or set for all Makefiles:
https://stackoverflow.com/a/56060185/