

# CSE 220 – C Programming

Writing Large Programs Part 2

# Code Sharing

- Share function prototypes
  - (already covered)
- Share macro definitions
- Share variable definitions

# Sharing Macro Definitions

physconstants.h

```
#define speedLightKms 299792  
#define kWhToKJoules 3600  
#define calToJoules 4184
```



Prog1.c

```
#include  
"physconstants.h"  
...  
...  
int x = calToJoules*10;
```

Prog2.c

```
#include "physconstants.h"  
...  
  
float y =  
speedLightKms/1000.0;
```

# Sharing external variables

config.h

```
extern int delay;
```

config.c

```
#include "config.h"  
int delay = 10;  
//define
```

“extern” tells the compiler that the variable delay is declared elsewhere, so there’s no need to allocate space for it

MainProg.c

```
#include "config.h"  
  
int main(void) {  
    printf("%d", delay);  
}
```

config.c actually  
initializes the variable

Make sure the types  
match in the definition  
and the declaration

# Makefiles

- Putting all source files on command line:
  - Tedious
  - Wastes time building program
- Makefile: a file containing information to build the program
  - Lists the files that are part of the program
  - Dependencies: files needed by the program
  - Normally stored in a file named Makefile

# Example Makefile contents

```
myprog: myprog.o arrayDisplay.o arrayControl.o
```

```
    gcc myprog.o arrayDisplay.o arrayControl.o -o myprog
```

```
myprog.o: myprog.c arrayDisplay.h arrayControl.h
```

```
    gcc -c myprog.c
```

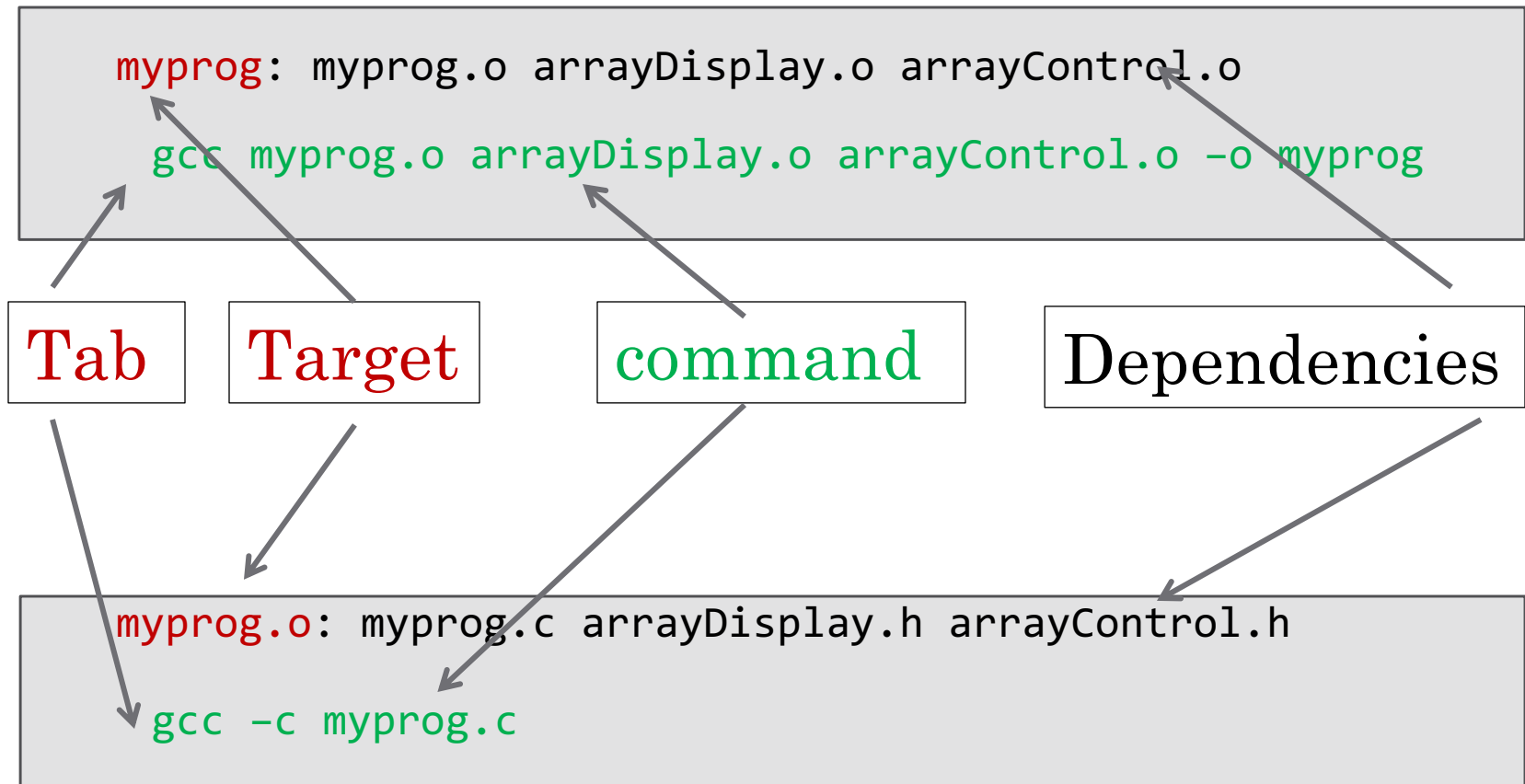
```
arrayDisplay.o: arrayDisplay.c arrayDisplay.h
```

```
    gcc -c arrayDisplay.c
```

```
arrayControl.o: arrayControl.c arrayControl.h
```

```
    gcc -c arrayControl.c
```

# Example



# Makefiles

- The *make* utility is used to invoke a makefile

`make target`

`make myprog`

- Calling `make` without a target will build the target of the first rule
- When rebuilding, `make` utility checks the timestamp of the files, and decides what needs to be rebuilt