

# Cs60: C, C++ and Unix

Week 6

# Today

- Homework 6
- File I/O in C
- Review

# #include guards

- When we have a project with many source files, we may need to include one header file in several different source files.
- This causes a problem because the classes and or functions defined in the header are defined twice and we get a compiler error.
- We can avoid this by using #include guards in all our header files.

# #include guards

- `#ifndef <macro>`
  - Checks if the macro is defined.
  - `#endif` closes the if statement.
- `#define <macro>`
  - Defines a macro

# #include guards

```
#ifndef PEOPLETREENODE_HEADER
#define PEOPLETREENODE_HEADER
    Class PeopleTreeNode
    {
        //Lots of definitions here.
    };
    //Maybe some more stuff here.
#endif
```

# #include guards

- Make sure to pick coherent names for the macros defined in the #include guard.
- If you have two macros with the same name, one of the headers will never be included.

# File I/O

- All we need to open a file in c is a pointer of type FILE.

```
FILE *myfile;
```

- The type FILE is defined in stdio.h.
- To open a file, we use the fopen function.

```
myfile = fopen("file.txt", "r");
```

# File I/O

- Some useful functions:
  - fopen – open file.
  - fclose – close file.
  - fseek – seek to specified byte in file.
  - fgetc – read a character from the file.
  - fputc – write a character to the file.
  - fprintf – write to file.
  - fscanf – scan from a file.
  - feof – check if we reached the end-of-file.
  - ferror – check if an error has occurred.

# Opening a file

- C doesn't need try/catch exceptions to handle problems opening files.
- It's still a good idea to make sure that we were able to open the file before trying to read or write to it.

```
FILE *myfile;  
  
myfile = fopen("file.txt", "r");  
  
if (myfile == NULL) {  
    printf("Can't open the file\n");  
    exit(1);  
}
```

# Modes

- The second parameter in the `fopen` function determines the mode that we will use to handle the file.

`r` – open file for reading.

`w` – create file for writing.

`a` – append to file.

`r+` – open file for reading and writing.

`w+` – create file for reading and writing.

`a+` – append or create text file for appending.

# Reading a file

```
#include <stdio.h>

int main(void){
    FILE *myfile;
    char c;
    myfile = fopen("data.txt", "r");
    if (myfile == NULL){
        printf("Can't open the file\n");
        return(1);
    }
    do{
        c=(char)getc(myfile);
        //do stuff here
    }while (c!=EOF);
    fclose(myfile);
    return(0);
}
```

# Reading and writing

- `fprintf` and `fscanf` are the same as `printf` and `scanf`, except that you also pass the file pointer as an argument.

```
printf("Number = %d\n", num);
```

becomes:

```
fprintf(myfile, "Number = %d\n", num);
```