

## C Tutorial 2

CIT 593  
Diana Palsetia

1

## Example: nullpointer.c

```
int * p;  
p = NULL;  
printf("%d", *p);
```

- Dereferencing a null pointer results in program crash
- Because Address 0 is not a legal address for most programs on most platforms
- Runtime Error: Segmentation fault

2

## Ascii to Integer Example: asciitoint.c

```
#include<stdio.h>  
#include<stdlib.h>  
int main(){  
    char mychar = '1';  
    char *string = "123";  
    int myint = 0;  
  
    printf("ascii (in Dec) value of \"1\" = %d\n",mychar);  
    myint = atoi(&mychar);  
    printf("decimal value of \"1\" =%d\n",myint);  
  
    myint = atoi(string);  
    printf("decimal value of \"123\"=%d\n",myint);  
  
    return 0;  
}
```

`int atoi ( const char * string );`



3

## File I/O

- A **file** is a sequence of ASCII characters stored on some device.
  - Allows us to process large amounts of data without having to type it in each time or read it all on the screen as it scrolls by.
- We use “**FILE**” structure to do file I/O
- **FILE** structure has information such as:
  - Location of the file
  - Whether the file is being read or written
  - Current character position in the file
  - Whether errors or end of file has occurred
- Hence we declare file handler or pointer to do any operations  
`FILE *infile;`
- Note: The **FILE** type is defined in <stdio.h>.

4

## fopen

- The `fopen()` function opens the file in a particular mode  
`FILE *fopen(char* name, char* mode);`
- **First argument: name**
  - The name of the physical file, or how to locate it on the storage device. This may be dependent on the underlying operating system
- **Second argument: mode**
  - How the file will be used:
    - "r" -- read from the file
    - "w" -- write, starting at the beginning of the file
    - "a" -- write, starting at the end of the file (append)
  - Returns a pointer of type File
    - If pointer is NULL (0) then error has occurred in reading the file
    - Else a non-zero value is returned

5

## fprintf & fscanf

- Once a file is opened, it can be read or written using `fscanf()`, `fprintf()`, `fgetc()` respectively.
- These are just like `scanf()` and `printf()`, except an additional argument specifies a file pointer.

```
FILE *infile;
FILE *outfile;

fprintf(outfile, "The answer is %d\n", x);

fscanf(infile, "%d", &number);
```

6

## fgetc

```
int fgetc (FILE * stream);
```

- Get the next character from file stream.
- Returns the next character of the file stream and increases the file pointer to point to the following one.
  - **Return Value:** The character read is returned as an `int` value.
  - If the End Of File has been reached or there has been an error reading, the function returns EOF

See next slide on how to use `fgetc()`

7

## Example of File I/O: textcopy.c

```
#include <stdio.h>

/* Library function prototypes */
FILE * fopen(const char * filename, const char * mode);
int fclose(FILE * stream);
int fgetc (FILE * stream); */

int main(int argc, char * argv[] )
{
    char * inputname;
    char * outputname;

    FILE * infile;
    FILE * outfile;

    int charread = 0;
    if (argc != 3)
    {
        printf("Usage: textcopy inputfile\noutputfile.\n");
        return 1;
    }
    inputname = argv[1];
    outputname = argv[2];

    /* Open/ create files. */
    infile = fopen(inputname, "r"); /* mode "Read" */
    outfile = fopen(outputname, "w"); /* mode "Write" */

    if (infile == NULL || outfile == NULL) /* files did not open */
    {
        printf("Files could not be opened\n");
        return 1; /* quit now */
    }

    while ((charread = fgetc(infile)) != EOF)
    {
        printf("%c", charread);
        fprintf(outfile, "%c", charread);
    }

    /* close the file */
    if (fclose(infile) != 0 || fclose(outfile) != 0)
    {
        printf("Close file error.");
    }
    return 0;
}
```

Reads file & gets a character from file till EOF is encountered

8

### Struct Example: WeatherDay

#### .h file usually

- Contains the declaration of data structure
- You can typedef the struct here
- Contains function prototypes that use struct

#### .c file contains (example: weatherday.c)

- Implementation of the function defined in .h file
- Must remember to include the .h file
- Can also contain function main()

9

### Using separate file for main()

- In order to instantiate a object of new type
  - must remember to include .h file (i.e. where the data structure is declared and typedefed)
    - e.g. WeatherDay day1
  - You can also declare pointer of new type
    - WeatherDay \* dayptr

10

### Struct pointers: main.c (part 1)

```
#include <stdio.h>
#include "weatherday.h"
int main() {
    WeatherDay day1;
    WeatherDay *dayptr =
        &day1;
    day1.precip = 1.2;
    dayptr->highTemp = 65;
    dayptr->lowTemp = 30;

    PrintWeatherData(&day1);
    printf("-----\n");

    avg = avgTemp(dayptr);
    printf("Avg temp of the day
    = %f\n",avg);

    //printf("Enter low temp\n");
    //scanf("%d", dayptr-
    >lowTemp);
    //PrintWeatherData(dayptr);

    PrintWeatherData(dayptr);
    printf("-----\n");
    return 0;
}
```

11

### Malloc/Free Example: main.c (part 2)

```
printf("How many weather days you want to create\n");
scanf("%d",&num);
day2 = (WeatherDay*)malloc(num * sizeof(WeatherDay));
if(day2 == NULL){
    printf("Could not allocate memory\n");
}
day2[1].highTemp = dayptr->highTemp;
day2[1].lowTemp = dayptr->lowTemp;
day2[1].precip = dayptr->precip;

PrintWeatherData(&day2[1]);

free(day2); //free allocated memory
```

12

## Makefile

- **makefile** is read by program called **make**
- Make compiles, assembles, and links your source files
- Think of it as the lcs file in LC3
- The file you write is called “makefile” without any file extensions
- Write a file called “makefile” (with no extensions)
- Then at prompt type : make

13

## Example: makefile

```
COMPILER=gcc
LIBS=
OPTIONS=-g
NAME1=weatherday
NAME2=main
NAME3= wDay

all:
    $(COMPILER) $(LIBS) $(OPTIONS) $(NAME1).c
    $(NAME2).c -o $(NAME3)

clean:
    rm -f *.o $(NAME3)
```

14