Introduction to Computer Science W 1113 – Lab (C) Lab12

Suhit Gupta

4/22/04

Questions about HW6

2

Recap from Lab 10

- const Pointers
- Pointer arithmetic
- Pointers and Arrays
- Pointers and Strings
- Pointers and Structs
- Command Line Arguments (Pointers)
- Pointer to a Pointer
- How not to use pointers

3

Recap from Lab 11

- malloc
- free
  - Dangling pointers
- calloc
- Pointers and Linked Lists

4

A repeat of the linked list example

- So how does malloc help us here?

```
struct linked_list {
    char data[30];
    struct linked_list *next_ptr;
}
struct linked_list *first_ptr = NULL;
```

- So we want to use malloc instead of creating an array of linked lists that will limit the number of nodes in the linked list to the size of the array
- How can we do this?

5

Pointers and Linked Lists contd…

```
new_node_ptr = malloc(sizeof(struct linked_list));
```

- This created the new node and allocates the correct amount of memory

```
(*new_node_ptr).data = item;
```

- This will store the value of item into data

```
(*new_node_ptr).next_ptr = first_ptr;
```

- The node now points to first_ptr

```
first_ptr = new_node_ptr;
```

- The new element is now the first element

6

File I/O

- Now that you know pointers and malloc, you are ready for file I/O
- *Usage*: FILE *file;
- To open a file – fopen();
- *Usage*: void *fopen(*name, mode*);
  - file = fopen (name, mode);
  - NULL is returned on error
  - *name* is the actual name of the file
  - *mode* indicate the property with which to open the file

7

Options for mode

- *mode* indicates whether the file is open for reading or writing
- 'w' for writing
- 'r' for reading
- Example

```
FILE *in_file;
in_file = fopen("input.txt", "r");
if (in_file == NULL) {
    fprintf (stderr, "Error: Could not open the input file 'input.txt'\n);
    exit (8);
}
```

8

Close a file – fclose()

- fclose() will close a file
- *Usage*: fclose (pointer to file);
- status = fclose(in_file);
  - You don't need status
    - fclose(in_file);
      - This will just throw away the return value
  - 'status' will be 0 is file was closed successfully
  - It will be non-zero is there is an error
    - Do a man on fclose to see the different error codes

9

Simple operations

- fputc – This function writes a single character to a file
  - *Usage*: fputc (character, file)
- fputs – This function writes a string to a file
  - *Usage*: fputs (string, size, file)
  - *Usage*: fputs (string, sizeof(string), file)
    - This will return a pointer to the string if successful or NULL if there is an error
  - Sometimes there are problems when you try to write strings that are very long

10

Simple operations II

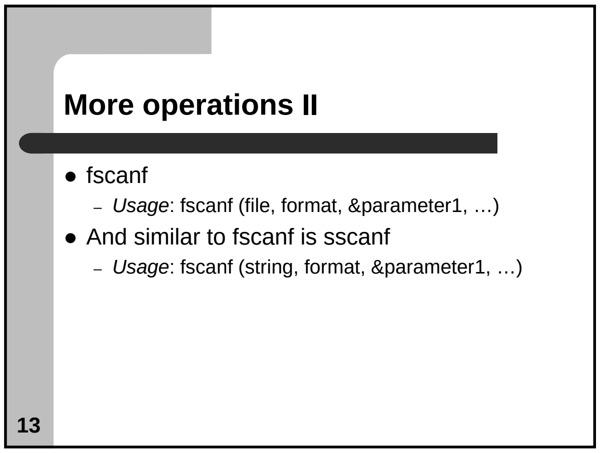
- fgetc – This function gets a single character from a file
  - *Usage*: fputc (character, file)
  - Typically used when you have a stream of data coming in and you need to read the characters coming in one at a time
- fgets – This function gets a string to a file (similar to fputs)
  - *Usage*: fgets (string, size, file)
  - *Usage*: fgets (string, sizeof(string), file)
    - This will return a pointer to the string if successful or NULL if there is an error
  - Read the text book as well as the man page to see the intricacies with fgets
    - You need to worry about the \n, \0, etc at the end of the string as well as the end of the file

11

More operations

- fprintf
  - *Usage*: count = fprintf (file, format, parameter1, parameter2, …)
    - count is the number of characters sent (-1 if error)
    - format describes how the arguments are to be printed
    - parameters – to be converted and sent
- Similar function
  - sprintf
    - *Usage*: sprintf (string, format, parameter1, parameter2, …)

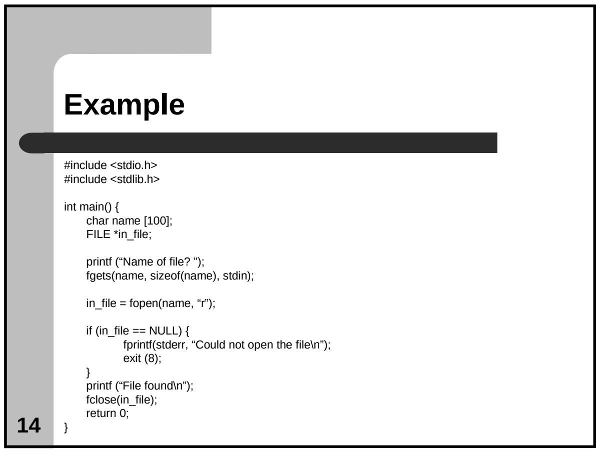
12



More operations II

- fscanf
  - *Usage*: fscanf (file, format, &parameter1, …)
- And similar to fscanf is sscanf
  - *Usage*: fscanf (string, format, &parameter1, …)

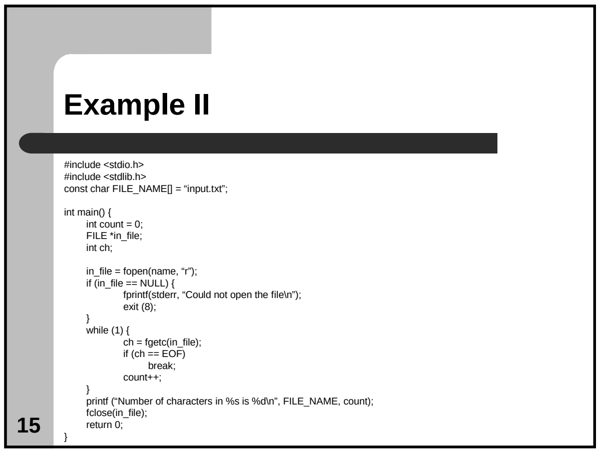
13



Example

```
#include <stdio.h>
#include <stdlib.h>

int main() {
	char name [100];
	FILE *in_file;

	printf (“Name of file? ”);
	fgets(name, sizeof(name), stdin);

	in_file = fopen(name, “r”);

	if (in_file == NULL) {
			fprintf(stderr, “Could not open the file\n”);
			exit (8);
	}
	printf (“File found\n”);
	fclose(in_file);
	return 0;
}
```

14



Example II

```
#include <stdio.h>
#include <stdlib.h>
const char FILE_NAME[] = “input.txt”;

int main() {
	int count = 0;
	FILE *in_file;
	int ch;

	in_file = fopen(name, “r”);
	if (in_file == NULL) {
			fprintf(stderr, “Could not open the file\n”);
			exit (8);
	}
	while (1) {
			ch = fgetc(in_file);
			if (ch == EOF)
				break;
			count++;
	}
	printf (“Number of characters in %s is %d\n”, FILE_NAME, count);
	fclose(in_file);
	return 0;
}
```

15

Example III

```
#include <stdio.h>
#include <stdlib.h>
#ifndef __MSDOS__
#include <unistd.h>
#endif __MSDOS__

int main() {
      int cur_char;
      FILE *out_file;

      out_file = fopen (“test.out”, “w”);
      if (out_file == NULL) {
                fprintf(stderr, “Cannot open output file\n”);
                exit (8);
      }
      for (curr_char = 0; cur_char < 128; cur_char++)
                fputc(cur_char, outfile);
      fclose (out_file);
      return 0;
}
```

16

Advanced concept - strtok()

- Used to tokenize a given string
- *Usage*: char *strtok (char *s1, const char *s2)
- It searches for tokens in s1, using the character in s2 as token separator
- If s1 contains one or more tokens
  - the first token in s1 is found
  - the character immediately following it is overwritten with a NULL
  - the remainder of s1 is stored elsewhere
  - the address of the first character in the token is returned
  - subsequent calls with s1 equal to NULL return the base address of a string supplied by the system that contains the next token
  - If no additional tokens are available, NULL is returned

17

Example using strtok

```
char s1[] = “ this is,an   example ; ”;
char s2[] = “,;  ”;

printf (“\“%s\””, strtok (s1, s2));
while ((p=strtok(NULL, s2)) != NULL)        // p here is a pointer to the
     printf(“ \“%s\””, p);                  // character we are checking
putchar(‘\n’);
```

- This will print out
  - “this” “is” “an” “example”

18

strdup()

- Duplicates a string
- *Usage*: char *strdup(const char *s);
- Basically, given a string, it will duplicate it
  - it will return a pointer to the duplicate string

19

Things to remember

- Always close the file before leaving the program
- Functions can take file pointers as arguments
  - void my_func (FILE *, FILE *) { … }
- All functions take file pointers and not the file names themselves

20

Assignment

- Read Ch. 18 from the Practical C Programming book
- **HW6**

21