Reading and writing files for the Todo List app

CS 261 Lab #8

Oregon State University OSU

Todo List app architecture

main.c	Show the user a menu of commands, parse user input
toDoList.c	Create/compare/print *Tasks*, save and load *Tasks* to disk
dynamicArray.c	Heap implementation
main2.c	Test your sortHeap implementation
type.h	Sets our TYPE to a void pointer
program_demo.txt	Example of how users should be able to interact with your program

The app will prompt the user to enter a command

You need to figure out what to do after the user enters their command

Example: if the user enters “g”, you need to print the first task

Example: if the user enters “l”, you need to load a todo list from a file

You'll need to **open the files** for loading and saving the Todo List yourself

C uses **file pointers** to access files

Use *fopen(const char *filename, const char *mode)* to open a file pointer

```
FILE *fp = fopen("filename.txt", "r");
```

file to open

mode

```
assert(fp != NULL);
```

← ensure file was opened

Common file modes

“r” — read-only

“w” — write-only (overwrite existing file)

“a” — append-only

“r+” — read and write

“w+” — read and write (overwrite existing file)

“a+” — read and append

Read an entire line with **fgets**

Write to a file with **fprintf**

Close a file with **fclose**

```
FILE *reader = fopen(“input.txt”, “r”);
FILE *writer = fopen(“output.txt”, “w”);
char line[100];

while(fgets(line, sizeof(line), reader) != NULL) {
    fprintf(writer, “%s”, line);
}

fclose(fp);
```

You can also use **fgets** to read an entire line from the command prompt

"standard input" file pointer

```
char input[100];
if (fgets(input, sizeof(input), stdin) != NULL) {
    printf("You entered '%s'", input);
}
```

stdin, *stdout*, and *stderr* are also available

If the user is supposed to enter a number, **scanf** can read it directly into a variable

```
int inputNumber;
printf(“Enter a number: ”);
scanf(“%d”, &inputNumber);
```

Same format specifiers as *printf*

Memory location to store the result in

You can use a **case** statement to figure out what to do with the user's command

```
switch(cmd) {
    case 'a':
        // Prompt user to add task description
        // and priority, read input, create task,
        // and add it to the heap.
        break;
    case 'g':
        // Get the first task from the heap and
        // display it to the user.
        break;
}
```

Each *case* is a different command

If you don't *break*, the next *case* will also be executed!

No extra code this week!

Just work on **Assignment #5**

(slides are available at http://dropline.net/cs261/lab8)