

# Advanced Programming COEN 11

## Lab 6

## Lab 6

- ❑ Waiting List with File I/O
  - Extension of project 5
    - Add saving/retrieving

## Lab 6

- ❑ Initially
  - The waiting list may be either
    - empty
    - formed with information read from a file
- ❑ At the end
  - The updated waiting list is saved into a file

## Lab 6

- ❑ The info should be saved in a text file according to the following format:

Name	Group Size
Joe	5
Mary	3
Zoe	3

- ❑ It should be possible to read the file with commands such as cat and more

## Lab 6

- The name of the file is an argument for the program

- If the file does not exist

- fopen returns NULL for reading
- the list starts empty and is saved at the end into a file with the given name

- If the file does exist

- the list is initially formed with the information obtained from the file and is saved into the same file at the end

## Lab 6

- The name of the file is an argument for the program

- Example:

```
# ./wait_list <file_name>
```

or

```
# ./a.out <file_name>
```

## Lab 6

- The name of the file is the first argument for the program

- In the code:

```
main (int argc, char *argv[ ])
{
    ...
    if (argc == 1)
    {
        printf ("The name of the file is missing!\n");
        return 1;
    }
    else
        read_file (argv[1]);
    ...
}
```

## Lab 6

- The name of the file is an argument for the program

- In the code:

- argc gives the number of arguments
- argv is an array of strings, each of which is one of the arguments for the program
- argv[0] is the name of the executable
- argv[1] - argv[argc - 1] are the arguments

## Lab 6

- ❑ The waiting list is created/modified interactively, as in project 5.
  - 1 name number - insert a node with the name and number of people specified
  - 2 size - extract (show and delete) oldest node with a number  $\leq$  size
  - 3 - print the list, name and number, from oldest to newest
  - 4 (new) - save the list in the file specified and quit

## Lab 6

- ❑ Requirements
  - Use same insert function for inserting information from the file and from the keyboard.
  - Your insert function should have the following type:
    - void insert (char \*, int);
  - Read the name and number to local variables (char array and int) before calling the insert function.

## Lab 6

- ❑ Requirements
  - Names cannot repeat!
    - Need to deal with that before calling function insert
  - Use function fseek to read the beginning of the file (header) before reading the data (names/numbers).
    - Type <man fseek> to learn how to use the function

## Lab 6

- ❑ Requirements
  - Add two functions
    - void save\_data (\*char);
    - void read\_data (\*char);
  - Both receive the name of the file as argument.