Advanced Programming
COEN 11

Lab 8

Lab 8

- List of tasks with File I/O

- Extension of lab 7
  - Add retrieving data from file
  - Add task priority lists (array of heads)
  - Name of file will be an argument to the program (argc and argv)

Lab 8

- Use an array of linked lists
  - The array has **three heads**, each starting a list
  - Tasks are added to a list according to a **priority**, new member in the struct
    - Priorities can be **1, 2, or 3**
  - Each list is sorted by **hour** and **minutes**
- Due in week 8

Menu

- 1 <task> **<prio>** <hour> <min> <extra info according to the hour>
  - Same as lab 7, but add the priority and insert in the **prio** list (index=prio-1)
  - Function **check_duplicate** needs to traverse the 3 lists
- 2, 3
  - Same as lab 7, but need to traverse the 3 lists
- 4 <task>
  - Delete the specified task, but need to search in all 3 lists
- 0
  - Save all data to a text file, file name is an argument passed to main function, need to traverse all 3 lists
  - Deletes all the nodes in all 3 lists
  - Quits

Requirements

- Use an array of linked lists. The list heads will be kept in the array

  NODE *heads[3];

- Several changes throughout the code
  - Traverse the arrays of lists in an outer loop
  - There are 3 lists in the array, so head should be heads[index]

More Requirements

- Initially, before interacting with user
  - Read text file and create lists with saved info

- At the end
  - Before quitting, all the data is saved to the same text file (same format as on lab 7, but include the priority)

```
Task1    prio1 h1:m1 extra1
task2    prio2 h2:m2 extra2
```

More Requirements

- 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 are saved at the end into a file with the given name
  - If the file does exist
    - the lists are initially formed with the information obtained from the file and is saved into the same file at the end

Lab 8

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

```
# ./lab8 file_name
```

or

```
# ./a.out file_name
```

Lab 8

- The name of the file is the first argument for the program
  - **In the code:**

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

   read_file (argv[1]);
   . . .
```

Lab 8

- 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

More Requirements

- One new function, called from main
  - New: read from file
    - Receive file name as an argument
    - Call insert to insert the data read from file
  - Change: save to file
    - Receive file name as an argument

More Requirements

- Use the same insert function for inserting information
  - from the file and
  - from the keyboard
- Your insert function should have the following type:

  void insert (struct node);
- Read the new task information to a local struct before calling the insert function
  - You may use an extra function for that

Grading

- Pre-lab
  - Test plan – 10%
- In the lab
  - Demo to the TA – 30%
  - Submit your code to Camino – 60%

Obs: If you had already submitted the code to Camino, you must demo the same code.