Input file type: .txt

Input file line structure:
Job id : Description : Machine number : Time : Client name

structure JOB :
5 character pointer type fields/variables

structure MACHINE :
2 pointers of JOB type : head and tail

#1 main() takes files as CML argument

1. Take multiple files as a input from user : in main()
   A : Check if argc >1
2. Send one file at a time in assignJob() function.

3. Create threads for each file, calling assignJob(). ######imp

4. Repeat steps 2 and 3 until all files are read.

#2 (core function) (void *) Assign_machine(void * job)

1. Type cast all the fields in the structure into equivalent data types
   //inside assignJob()
   1-a . Check
   : File is available in the directory or not

   1-b. Open each line of the file in "read" mode : use fgets()
   1-b- A: Check : if file is empty
   1-b-B: Validate String : call stringValid()
   addToList() if valid
   OR
   Store into invalidFile()
2. addToList() will give a Linked List of valid inputs.

3. Traverse the list to assign to different machines

4. Call validAdd() to update time of each machine schedule :
    - Use mutex locks for each validAdd() call, so that time is updated without interruptions.

#3 (int) stringValid(char *)

1. Check input:
    a. return 1 if input string matches correct format
    b. Else return 0

#4 addToList((machine*), (char *))

1. Char * points to each line of the file.
2. Machine * holds head and updated list.

3. Keep head pointing to the first node and update tail everytime a new node is added.

#5 validAdd((job *), int , int)

1. Open Schedule files for each machine in read and write mode to make entry of incoming schedule updates.

#6 invalidFile(char *)

1. Open invalidRecord.txt in “a+” mode, add incoming invalid inputs.
2. Put a lock because multiple threads may use it to access invalidRecord file and each invalid record should be preserved.