

Schedule

Wednesday

Status Reports

Robot Parade: working toward completion

Office hours: 2:30-4:00

Thursday

Office hours (no sign-ups): 2:30-4:00, 5:00-7:00

Friday

Robot Parade in class

Office hours: 2:30-4:00

Robot Parade Program

Process:

- robots placed in line toward starting line
- one partner signals when robots ahead clear space; other partner types at keyboard to start robot moving forward
- one partner signals when robot at starting line; other partner types to tell robot it has reached the starting line
- program starts reading commands from the file

Logging

- Project specifies that each line of the log file should have the format:

command, parameters, time stamp, (left brightness, middle brightness, right brightness)

- Since time stamp, brightness material common for all lines, convenient to have function that prints that material to the log file; then call function as needed.

Robot Parade Sequence

1. Place robots in line, heading toward starting line, and turn them on
2. Start program in terminal window, including command line argument for reading from a file
3. One partner (A) watches robot, and signals other partner (B) when space becomes open ahead of robot
4. Partner B enters something on keyboard to start robot moving forward slowly
5. Partner A signals B when robot gets to starting line
6. B enters something on keyboard, so program will start reading commands from the file
7. Robot follows commands (going through door and down the hall). A and/or B may adjust robot's position, if collision imminent

When a user types a character (and return), the robot should start moving. When the user types another character (and return), the robot should stop moving.

Assume: `char ch; int i; char line [80];`

```
1. scanf ("%c", &ch);  
   rmotors (0.25, 0.25);  
   scanf ("%c", &ch);  
   rmotors (0.0, 0.0);  
2. scanf (" %c", &ch);  
   rmotors (0.25, 0.25);  
   scanf (" %c", &ch);  
   rmotors (0.0, 0.0);
```

```
3. scanf ("%d", &i);  
   rmotors (0.25, 0.25);  
   scanf ("%d", &i);  
   rmotors (0.0, 0.0);  
4. fgets (line,81,stdin);  
   rmotors (0.25, 0.25);  
   fgets (line,81,stdin);  
   rmotors (0.0, 0.0);
```

Which of the these approaches work?

Vote 1:

1. 1 and 2
2. 1 not 2
3. 2 not 1
4. neither 1 nor 2

Vote 2:

1. 3 and 4
2. 3 not 4
3. 4 not 3
4. neither 3 nor 4

Clearing a line

Assume that `infile` has been opened to either `stdin` or a file, and assume an input line is no longer than 80 characters. Which of the following will read all characters to the end of the current line.

1. `char line [81];`
 `fgets (line, 81, infile);`
2. `while (fgetc(infile) != '\n');`
3. `char ch;`
 `fscanf (infile, "%c", &ch);`
 `while (ch != '\n')`
 `fscanf (infile, "%c", &ch);`
4. exactly two of the above
5. all of the above