Grading Program 2

CS344 – Benjamin Brewster

This document details how to grade Program 2.

How to Get Student Submissions and Upload the Grades

Once you have finished grading the assignment, upload the score to Canvas in this manner:

1) Go to Canvas in your web browser of choice.
2) Click on Assignments on the left menu.
3) Click on Program 2 – adventure.
4) In the upper right corner, click on SpeedGrader.
5) Pick the name of the person you're grading, above. Note that you can use the left and right arrows, and the drop down list, to select between students.
6) Once you've chosen a person, check in the upper right that the LATEST version is selected. Read their comments below.
7) Download their zip file by clicking it in the list on the left, placing it onto os1 (where all grading is done). Follow the instructions below to assign a grade (in points).
8) In order to commit the grade to Canvas, come back to this assignment, and enter the point value in the Assessment field on the right, in the middle. Then:
   a. If the student lost points, record the reasons why in the Add a Comment Field.
   b. If the assignment was late by more than an hour past the due date, it will be marked as late. Please review the Syllabus late policy.

      Once you have generated the final points value, adjust it downwards, if necessary, according to the late percentages given above.

How to Grade

Here is the process:

1) Read the Assignment as given on Canvas.
2) Download the student's .zip file to os1, which will contain two program files. Extract the two files and rename them to "`<username>.adventure.c`", and "`<username>.buildrooms.c`" if needed.
3) Compile the programs on os1 using these lines:

```
gcc -o <username>.adventure <username>.adventure.c -lpthread
gcc -o <username>.buildrooms <username>.buildrooms.c
```

4) If either program does not compile, the grade is zero. Please ask the student to resubmit the problem program(s) within 2 days, or the grade will stand as a zero. Let the student know there will be a penalty of 8 points because of the compilation issue.

5) Once both have compiled, run through the commands listed below.

Grading Criteria

There are a total of 160 points available for Program 2. My philosophy is that we should grade somewhat generously: give the student the benefit of the doubt, where possible.

Grade the program as follows.

1. Run the student's room-building program by executing:
   `<username>.buildrooms`
2. No output should be returned. Run ls, and look for a new directory to have been created with this name:
   `<STUDENT ONID USERNAME>.rooms.<PROCESS ID>`
3. Go into that directory, and look for 7 room files.
4. Open up each file in the directory – there should be 7 files, which will be named however the student chooses. The contents of these files should match the text below. If they do, award 3 points for each correct file/room:

   ```
   ROOM NAME: <room name>
   CONNECTION 1: <room name>
   …
   ROOM TYPE: <room type>
   ```
5. Now, run the game by typing:
   `<username>.adventure`

You should be presented with a prompt that looks like this:

```
CURRENT LOCATION: <current location>
POSSIBLE CONNECTIONS: <connection to room 1>, <connect 2>, <connect 3>.
WHERE TO? >
```

1. The cursor should be placed right after the > symbol, on the "Where To? >" line. If you see this, the current location header is worth 10 points, the connection list is worth 12 points, and the prompt line with correct cursor position is worth 5 points. If these details are not exact, but are close, awards half the points for each.
2. Type in a room name that matches one of the connections given (see the assignment for an example). You should be presented with a NEW prompt that matches this new room, with its connections. If you see this new prompt, award 15 more points. There is no partial credit here.
3. Type a room name that is NOT in the game. The program should tell you that that name can't be found, reached, etc. - it shouldn't crash, but should continue to run. If it crashes, or doesn't tell you there's a problem with the name, *subtract* 8 points off.
4. Next, attempt to follow connections repeatedly until you get to a screen that says you have won. It should then list the steps you took, and the number of steps you took, then it should exit to the shell prompt. Here are the points for those:
   a. Echo the ? variable (`%echo $?`) to get the exit value. It should return 0. If it does, award 10 points.
   b. If the number of steps is given, and correct, award 10 points.
   c. If the list of steps is given, and correct, award 15 points.

   d. If the program crashes during execution, or never ends in a win, attempt to continue the rest of the testing steps. If you can complete the rest of the steps (whether or not they get all of the points), subtract 15 points from the total at the end. If you cannot continue the steps, do the ones you can, awarding points as appropriate.

5. Re-run the rooms builder by executing:
   `<username>.buildrooms`
6. Now, re-run the game by executing:
   `<username>.adventure`
7. There should be a DIFFERENT path to victory, or at least a different set of connections between the rooms (it's unlikely, but possible, that the same connections are generated – re-run room builder and game to check if that happens). If there is a different winning path, award 10 points.
8. Look for a DIFFERENT rooms directory, now: it should have a different process ID number in the name. If you find it, award 5 points.
9. Re-run the room builder and game a THIRD time. There should be yet a third DIFFERENT path to victory. If there is, award 5 points.
10. Type the command "time" and hit enter. This should display the current date and time. If it does, and then presents the next game prompt, award 10 points
11. In `<username>.adventure.c`, look for a section that utilizes at least one mutex and a separate timekeeping thread to provide the time with the "time" command; if you see it, award 10 points (if you do not see both at least one mutex and a separate timekeeping thead, this check is worth zero points).
12. Look for a file called currentTime.txt. If it is present, open it up: it should contain a time stamp. If it does, award 6 points.

Grade comments generously. I expect to see comments describing what and why is happening frequently, as in every 4 or 5 lines, as appropriate. Fully commented code is worth 16 points. If there are no comments, the comment grade is 0 points. If there are comments, but not enough, give somewhere between 0 and 16 points, at your discretion.