GDB for Pintos

Problem Debug with GDB. The TA, who is responsible for project 1, have tried hard to finish the required functions for project 1. However, due to the limit of his poor debugging ability, the implemented version can't pass a specific test case for task 3, i.e. `mlfqs-load-60`. He became self-closing for about one day but still can't figure out the reason. Could you smart guys help him out and save this poor man? As you can see in the following figure, there is a value leap happening during the update of `load average`, resulting the failure of this test case. **You are required to tell us the reason why this value leap happen and show us the step-by-step process that how you figure it out using GDB. Please finish above tasks and hand in with your final report.** For this task, please first save your implemented pintos src and replace it with the released src for GDB task, i.e. `pintos-GDB.tar.gz`. This src is only for GDB task and you do not need to submit your code for this section. For task1-task3, please still submit your implemented pintos src using original released pintos.

```
perl -I../.. ../../tests/threads/mlfqs-load-60.ck tests/threads/mlfqs-load-60 tests/threads/mlfqs-load-60.result
FAIL tests/threads/mlfqs-load-60
Some load average values were missing or differed from those expected by more than 3.5.
  time   actual <-> expected explanation
------ -------- --- -------- -----------------------------------
     2     3.23  =  2.95
     4     5.11  =  4.84
     6     6.92  =  6.66
     8     8.68  =  8.42
    10    10.37  =  10.13
    12    12.01  =  11.78
    14    13.60  =  13.37
    16    15.13  =  14.91
    18    16.62  =  16.40
    20    18.05  =  17.84
    22    19.44  =  19.24
    24    20.78  =  20.58
    26    22.07  =  21.89
    28    23.33  =  23.15
    30    24.54  =  24.37
    32    25.71  =  25.54
    34    26.85  =  26.68
    36    27.94  =  27.78
    38    29.00  =  28.85
    40    37.72 >>> 29.88    Too big, by 4.34.
    42    37.72 >>> 30.87    Too big, by 3.35.
    44    37.72 >>> 31.84    Too big, by 2.38.
    46    37.72 >>> 32.77    Too big, by 1.45.
    48    38.11 >>> 33.67    Too big, by 0.94.
    50    38.11 >>> 34.54    Too big, by 0.07.
    52    38.11  =  35.38
    54    38.11  =  36.19
    56    38.87  =  36.98
    58    38.87  =  37.74
    60    38.87  =  37.48
    62    37.58  =  36.24
    64    36.34  =  35.04
    66    35.14  =  33.88
    68    33.98  =  32.76
    70    32.85  =  31.68
    72    31.77  =  30.63
```

(a) Present Bug and Failure

```
    30    24.54  =  24.37
    32    25.71  =  25.54
    34    26.85  =  26.68
    36    27.94  =  27.78
    38    29.00  =  28.85
    40    37.72 >>> 29.88    Too big, by 4.34.
    42    37.72 >>> 30.87    Too big, by 3.35.
    44    37.72 >>> 31.84    Too big, by 2.38.
    46    37.72 >>> 32.77    Too big, by 1.45.
    48    38.11 >>> 33.67    Too big, by 0.94.
    50    38.11 >>> 34.54    Too big, by 0.07.
    52    38.11  =  35.38
    54    38.11  =  36.19
    56    38.87  =  36.98
    58    38.87  =  37.74

After 30 seconds, load average=24.37.
After 32 seconds, load average=25.54.
After 34 seconds, load average=26.68.
After 36 seconds, load average=27.78.
After 38 seconds, load average=28.85.
After 40 seconds, load average=29.88.
After 42 seconds, load average=30.87.
After 44 seconds, load average=31.84.
After 46 seconds, load average=32.77.
After 48 seconds, load average=33.67.
After 50 seconds, load average=34.54.
After 52 seconds, load average=35.38.
After 54 seconds, load average=36.19.
After 56 seconds, load average=36.98.
After 58 seconds, load average=37.74.
```

(b) Value Leap & Correct Behavior

Hints: Think about the shortage of **fixed-point** when performing calculations. Try to print out the priority update at those bugged time ticks and see the values of related variable, i.e. **load average, recent cpu and priority**, using the commands defined in GDB macros.