Project 2: CPU Scheduling

Programmers: Drew Rife and Alec Waddelow

Course: CMPE320

Section: 2(11-12:50pm)

Instructor: Dr. Lee

Documentation

**Description of the Program:**

This program is a real-time CPU scheduler that uses the earliest-deadline-first (EDF) scheduling algorithm. We took our previous lab on Round Robin scheduling and used that as a basis for how our program should run. It will first check if you entered in a valid number of threads as we don’t allow you to have more than 10. Then we ask for you user input, if user input is correct then we check to see if it is possible for the threads to be scheduled and meet their deadlines. If so, then we can continue. From there the threads are created and ran.

**Algorithm:**

We both found this program to be very complex and took a lot of time to finally complete. Overall we like the algorithm we implemented, maybe a few things we can optimize to be better but that’s about it.

1. Make sure all user input is valid
2. Compute which order the threads will be ran until the user’s requested time interval
3. Compute whether the threads can be ran or not
4. If threads can be ran, create the threads
5. Timer runs by itself counting and printing time
   1. If timer notices that it’s temporary time equals the current thread running’s execution time, then it sets a flag ‘change\_thread’ to true and the scheduler will then choose which thread to run next
   2. If timer notices that CPU is idling it will run in its own loop until it detects that CPU is not idling anymore
6. Scheduler waits for a flag ‘change\_thread’ to be true for when to schedule threads
   1. It looks left to see if any threads were skipped because they couldn’t be ran due to the total time elapsed was not equal or greater to that thread’s last period
   2. If nothing previous then look to the right until you can find a thread that can be ran.
   3. If anything to the right was found it will run that thread
   4. If nothing was found to the right then that must mean that the CPU should idle
   5. If something was found to the left run that thread
7. The Runner function has the threads running and they idle when the CPU is idling - (although they still check to see if the total time elapsed is equal to or greater than their previous deadline)

**Output Success:**

./scheduler 3

Execution time for Thread 00 : 10

Execution time for Thread 01 : 15

Execution time for Thread 02 : 5

Period for Thread 00 : 30

Period for Thread 01 : 40

Period for Thread 02 : 50

How long do you want to execute this program (sec) : 150

Thread being executed : 0

01

02

03

04

05

06

07

08

09

10

Thread being executed : 1

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

Thread being executed : 2

26

27

28

29

30

Thread being executed : 0

31

32

33

34

35

36

37

38

39

40

Thread being executed : 1

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

Thread being executed : 2

56

57

58

59

60

Thread being executed : 0

61

62

63

64

65

66

67

68

69

70

CPU is idling now

71

72

73

74

75

76

77

78

79

80

Thread being executed : 1

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

Thread being executed : 0

96

97

98

99

100

101

102

103

104

105

Thread being executed : 2

106

107

108

109

110

CPU is idling now

111

112

113

114

115

116

117

118

119

120

Thread being executed : 0

121

122

123

124

125

126

127

128

129

130

Thread being executed : 1

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

CPU is idling now

146

147

148

149

150

Killed

**Output failed:**

./scheduler 3

Execution time for Thread 00 : 15

Execution time for Thread 01 : 15

Execution time for Thread 02 : 10

Period for Thread 00 : 30

Period for Thread 01 : 40

Period for Thread 02 : 50

How long do you want to execute this program (sec) : 150

These threads can't be scheduled. Program will exit.