KERNEL LEVEL THREAD SCHEDULING

Operating systems

SIRI NAMBURI

CSCE 611

FALL 2021

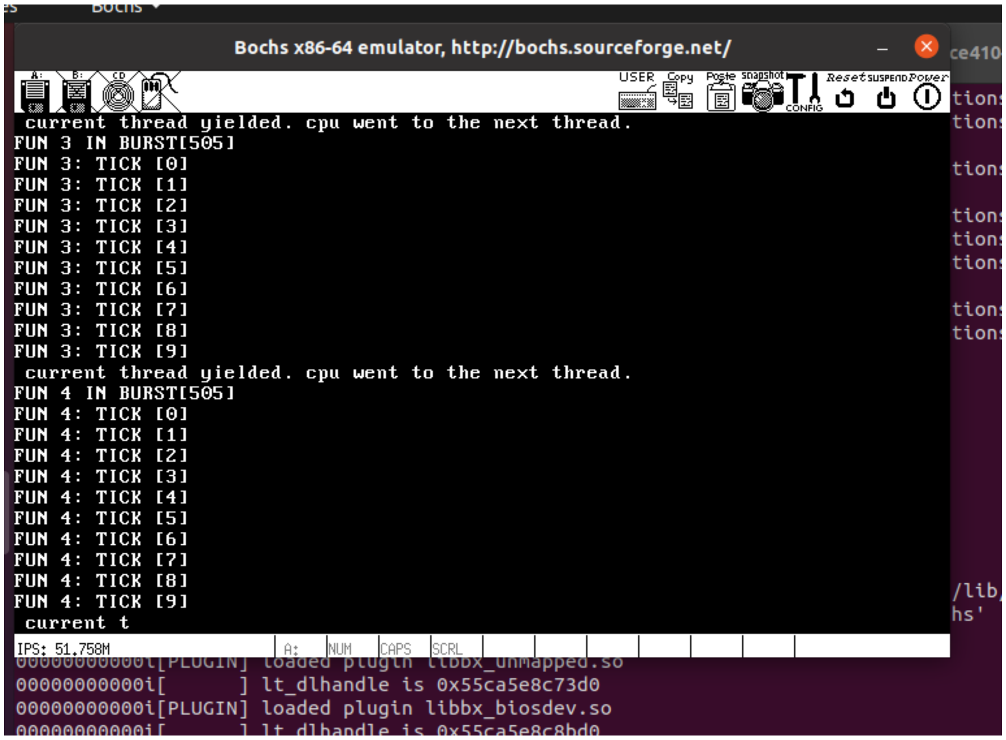
In this machine problem we mainly do scheduling of kernel level threads. In this we design a very basic “First In First Out “ scheduler to schedule the kernel level threads. The Scheduler.C file contains the Scheduler class which can implement the following functions.

Functions in Scheduler class:

1. Constructor : This initialises the variables in the class. Basically sets the count of threads to zero and the node pointers for the queue to null. The queue is implemented using a simple linked list, with two threadnode as elements. Each threadnode is a struct with two fields one for thread pointer and one for the next thread node. threadnode pointers head and tail are initialized which are set to null here
2. Yield: the current thread yields the cpu to the next thread in the queue. For this we use the head thread node pointer defined.
3. Resume: adds the input thread to our fifo queue. Uses add function in scheduler
4. Add : adds a given thread to the fifo queue. For this we use a tail threadnode pointer.
5. Terminate: delete all threads corresponding to a given node. Also if the current thread needs to be deleted, we yield it first

Also the thread shutdown function in thread.C file is modified. We use terminate function in scheduler class to delete all instances of thread and finally delete the thread.

Testing:
We use the already defined testing code in kernel.c. we can see that although four threads are initialized, only 2 threads remain which is the expected output.



```
 current thread yielded. cpu went to the next thread.
FUN 3 IN BURST[505]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
 current thread yielded. cpu went to the next thread.
FUN 4 IN BURST[505]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
 current t
```

```
00000000000i[PLUGIN] loaded plugin libbx_unmapped.so
00000000000i[      ] lt_dlhandle is 0x55ca5e8c73d0
00000000000i[PLUGIN] loaded plugin libbx_biosdev.so
00000000000i[      ] lt_dlhandle is 0x55ca5e8c8bd0
```

Bonus 1:
For this we are enabling interrupts in thread_start() function in thread.C file. Also pushing 1 instead of zero in eflags register. In addition to this the critical sections in scheduler are taken care of by enabling and disabling interrupts when necessary