|  |  |  |  |
| --- | --- | --- | --- |
| Ready\_List | Current Thread | Printf message | Context Switches |
| Empty | Main |  |  |
| Child1 | Main |  |  |
| Child1,Child2 | Main | Thread 0 looped 0 times | Main->Child1 |
| Child2, Main | Child1 | Thread 1 looped 0 times | Child1->Child2 |
| Main, Child1 | Child2 | Thread 2 looped 0 times | Child2->Main |
| Child1, Child2 | Main | Thread 0 looped 1 times | Main->Child1 |
| Child2, Main | Child1 | Thread 1 looped 1 times | Child1->Child2 |
| Main, Child1 | Child2 | Thread 2 looped 1 times | Child2->Main |
| Child1, Child2 | Main | Thread 0 looped 2 times | Main->Child1 |
| Child2, Main | Child1 | Thread 1 looped 2 times | Child1->Child2 |
| Main, Child1 | Child2 | Thread 2 looped 2 times | Child2->Main |
| Child1, Child2 | Main | //Finishing Thread Main  //Sleeping Thread Main | Main->Child |
| Child2 | Child1 | //Finishing Thread Child1  //Sleeping Thread Child1 | Child1->Child2 |
| Empty | Child2 | //Finishing Thread Child2  //Sleeping Thread Child2 |  |

**QUESTION:** Describe what and how a short-term CPU scheduler is implemented in Nachos by examining relevant source code files in the threads directory. Does the output from the test program reflect this scheduling discipline? Justify your answer.

The short-term CPU scheduler is implemented in Nachos as when the thread is created, Fork() is called on the created thread. This turns the thread into one that can be scheduled and executed by the CPU. The CPU scheduler then schedules the thread for execution. Furthermore, the output of the program above reflects this as the CPU schedules each thread in a loop until each corresponding thread is complete.