Report

Introduction

The project is made up of several stages the first stage which is the one that we are already in is building the skeleton of different threads and testing some of the predefined schedulers. This will be done by creating multiple threads with some parameters and attributes and starting assigning them on different schedulers in our case that is going to be (FIRST IN FIRST OUT), (ROUND ROBINS) and (SCHEDULING OTHERS) and observe the resulted outputs and how they act differently from each others.

Target

Testing different components that the operating system works on for creating our own demo of operating system.

Methodology

Firstly, an array is created in the main method with Size 4 So it Stores the four needed threads. Secondly, an attribute is created for the threads, scheduled parameters of the type struct, cpuset with type CPU_SET and two attributes of type clock named start and end for the starting and ending of the clock time. A variable named time is assigned to method clock() so the starting time of the execution is recorded . As computers are multicore and such code runs in parallel and also for this code to be able to create and set scheduling algorithm it had to be in unicore manner, this has been done by writing:

CPU_ZERO (&cpuset);

CPU_SET (0, &cpuset);

pthread_attr_setaffinity_np(&attr,sizeof(cpu_set),&cpuset);

before applying this three lines the output was sequential, but after learning about the multicore we use this code to make it in unicore

After initializing the attributes the next step is setting the policy of the three scheduling algorithms found in C (FIFO, RR and Scheduling other), after that the thread priority was set to 50 Per each So all sets has equal priorities which will make priority won't change ever during executing and running so the scheduling algorithm is to be set properly.

The thread attribute is then set by assigning each attribute and pointer to the scheduling parameter.

The for loop was for the four threads needed by calling the pthread_create method setting a pointer to each thread inside the array, the thread method and null. The thread method is assigning an id for each thread and execute three print statements.

The first one is Printing that the thread Start implementing.

The second is reading from the thread.

The third is the thread ended implementing.

The pthread_destroy method in this code was just for saving memory spacing. Lastly, the end attribute which was initialized earlier is assigned to method clock() to record the time of finishing the execution. In order to get the time, the attribute end was subtracted from the start after that the cpu time taken in seconds was printed.

Results

After starting running the code with scheduling type FIFO it was executed sequentially (first thread in first thread out) as expected.

Then with the second scheduling which is Round Robins, the resulted threads wasn't as expected, the output was suppose be alternating between each others while the actual result was sequential same as FIFO.

Running the code with type scheduling other each thread gets the same output weather in the start or on reading or even in the end of the execution with also a sequential technique as the others.