徐子修

B07902140

OS PROJECT 1 REPORT

1. Design

1.1. scheduler.c

The entry point. Assign the scheduler process to cpu core 0. Read input from stdin, sort the process by their ready time in ascending order and run the corresponding schedule algorithm.

1.2. utils.c

This file contains some syscall functions. SpawnProcess fork a new process. AssignCPU assign a process to a specified cpu core. SetProcessPriority sets priority for the process.

1.3. heap.c

A heap data structure that its top is the process with smallest remain execution time.

1.4. policies.c

All the schedule policy algorithms.

1.4.1. FIFO

Simply spawn a new process after the previous process is done.

1.4.2. SJF

Similar to FIFO algorithm, but when the current time is greater than a process's ready time, we insert it into the heap, and when a process has finished its work, we spawn the process that is on the top of the heap.

1.4.3. PSJF

Similar to SJF, but for every time unit, we compare the remain execution time of current running process with the one on the top of the heap, if the heap one is smaller, we replace the running process with the one in the heap and push it into heap.

1.4.4. RR

Use a linked list to maintain all the spawned processes, if current time is greater than a new process's ready time, we append it to the end of the linked list. The running process is always the first element in the linked list, after it's finished or a quantum is over, we remove it from the head and append it to the tail.

2. Kernel Version

V5.4.35

3. Comparison

For some of the algorithms like PSJF and RR, I simulate unit time in both scheduler process and task process. So the time in two process might have a little difference. Also There are some procedures to handle the switch of process like find the next process to run or set process priority…, etc. So there is a small period of time between two consecutive processes.