IIT DELHI

Assignment-2 report for COL331(Operating Systems)

Scheduling Algorithms

Prof. Smruti Ranjan Sarangi

Souvagya Ranjan Sethi
2019CS10405

Department of Computer Science & Engineering

April 5, 2023

1 System Calls

1. **sys_sched_policy(int pid, int policy**
   This code is an implementation of the set_sched_policy function in an operating system. The function takes two arguments, the pid of the process for which the scheduling policy is to be set and the sched_policy to be set. The function then checks if the sched_policy is a valid policy, which in this case is either 0 for EDF scheduling or 1 for Rate Monotonic scheduling.

   If the sched_policy is valid, the function iterates over all the processes in the process table to calculate the appropriate value for each scheduling policy. For EDF scheduling, the function calculates the least common multiple (LCM) of all the periods of the processes and checks if the sum of the execution times of all the processes divided by their respective periods is less than or equal to the LCM. For Rate Monotonic scheduling, the function checks if the sum of the execution times of all the processes multiplied by their respective rates is less than or equal to 83.

   If the condition for the respective scheduling policy is met, the sched_policy of the process is set and the function returns 0. If the condition is not met, the state of the process is set to UNUSED and the function returns -22.

2. **sys_rate(int pid, int rate)**
   This function sets the rate and weight of a process identified by its PID. The weight is calculated based on the rate using a formula (3*(30-rate)/29.0) and is rounded up to the nearest integer. The weight and rate values are then assigned to the process. If the process with the given PID is not found in the process table, the function returns -22. Otherwise, it returns 0.

3. **sys_deadline(int pid, int deadline)**
   This system call sets the deadline for each task. The deadline is set to be arrival time plus the mentioned deadline for that task.

4. **sys_exec_time(int pid, int time)**
   This sets the execution time for a task.

2 Scheduling Algorithms

1. **EDF**
   EDF stands for Earliest Deadline First and it is a scheduling policy used in real-time systems. In EDF scheduling, tasks with the earliest deadline are executed first. The deadline is the point in time by which a task must complete execution.

   In EDF scheduling, the scheduler monitors the deadlines of each task and schedules the task with the earliest deadline to run next. This ensures that the tasks with the tightest deadlines are completed first. If a task misses its deadline, it is considered a scheduling failure and may result in system instability or loss of data.

   EDF is a popular scheduling policy in real-time systems because it can guarantee that all tasks meet their deadlines as long as the system load is within the capacity of the processor.
   The scheduling algorithm is defined in the scheduler function in proc.c.

   Also, at eachstep when a process is about to be scheduled, we check for the schedulabilityof the group of tasks as a lot.

2. **RMS**
   Rate-monotonic scheduling (RMS) is a preemptive scheduling algorithm used in real-time operating systems (RTOS) with a fixed-priority scheduling policy. RMS is based on assigning a priority to each process based on its period or its inverse, the rate.

   In RMS, a process with a shorter period has a higher priority than a process with a longer period. The idea behind this is that the shorter the period of a process, the more often it needs to execute, and therefore, it has a higher priority.

   The rate of a process is the inverse of its period. Thus, a process with a shorter period has a higher rate than a process with a longer period. In RMS, a process with a higher rate has a higher priority than a process with a lower rate.

   RMS ensures that all processes meet their deadlines by guaranteeing that a higher-priority process will preempt a lower-priority process if it becomes ready to execute. However, it assumes that all processes have a fixed execution time and that there are no aperiodic tasks.