

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Operating System Principle |
| **Semester**: | 2nd semester of the academic year **2019-2020** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Fu Ruoxuan |
| **Student ID:** | 222019321062060 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Process scheduling in Linux: The CFS Scheduling | | | |
| Date | | April，2021 | Type | | □Confirmatory  □Design  √ Comprehensive |
| 1. **Objective & Requirements**    1. Understanding the concept of processor affinity    2. Learn how to set processor affinity in linux    3. Learn how to set process priority in linux    4. Understanding the CFS scheduling policy of linux    5. Review how to compile and load kernel module in linux. Learn how to pass parameters to kernel modules.    6. Learn how to access kernel scheduling information by loading kernel modules, and certify the CFS scheduling strategy. | | | | | |
| 1. **Experimental environment (**platform and software**)**   Virtualbox+Ubuntu linux | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results) 2. Tasks for this lab   Understand how the scheduling information is represented in PCB. Write, compile and load a kernel module to access the real runtime and virtual runtime of two processes with different priorities. Note that the two processes need to be bound to the same CPU core. To facilitate the task, you may need to learn how to pass parameters to kernel modules. Compute the ratio of real runtime and virtual runtime of the two processes, and compare the ratios with the ratio of the two processes’priorities. In this way you can certify the CFS scheduling policy.    Figure: The weights in CFS for different NICE values   1. Please provide your procedure to perform the tasks and source codes. 2. First compile set\_priority.c to change the priority of child process.Use sudo command to run the executable file.        1. And then add the kernel module for receiving the runtime and virtual time of parent process and child process to compare     3. According to the result of this experiment,  Ratio of parent process = 0.7778  Ratio of child process = 2.370583111  It is obvious that the ratio of child process is bigger than that of. parent process. | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   **Analysis of experimental results**  The experiment was able to obtain accurate results. The ratio of actual runtime to virtual runtime can be obtained from the formula of actual runtime to virtual runtime as ratio = weight/weight0. Also, since weight0 is a constant, the ratio can be used to represent the running weight of a process. In the experimental results, it is obvious that the child processes, that is, the processes with higher priority, have more weight  **Harvest**  CPU affinity means that the process should run on the specified CPU for as long as possible without being migrated to another processor. cpu affinity is divided into two categories: soft affinity and hard affinity. The Linux kernel process scheduler inherently has a feature called soft CPU affinity. Hard affinity, on the other hand, uses the API provided by the Linux kernel to force a process or thread to run on a given CPU core. What is used in Linux is int sched\_setaffinity(pid\_t pid, size\_t cpusetsize, cpu\_set\_t \*mask); to force the process identified as pid to be bound to the cpu set of mask. In this set cpu is represented by a bit mask. If the process is running on this cpu, the value of the corresponding position is 1, otherwise it is 0.  In a multitasking operating system, the cpu usage of processes must be controlled by humans. The reason for this is that some processes are very important and some processes are not so important at the moment. So there is a need to have priorities to distinguish them. Processes need to be switched, which means that in the same scheduling cycle, higher priority processes take up more time on the cpu, and lower priority processes take up less time. In linux, PRI is used to indicate the priority of a process. And in this experiment the priority of a process is modified by modifying NI (the value that indicates the priority of a process can be modified). Use setpriority(PRIO\_PROCESS, 0, -10);where PRIO\_PROCESS denotes the priority of a process, 0 denotes the current process, and -10 denotes the new priority  A scheduling algorithm called Completely fair scheduler is used in linux. The principle of this scheduling algorithm is to allocate runtime based on the weight of each process. Essentially the runnable processes form a red-black tree with virtual time seats for each node's key value. The higher the weight, the higher the priority of the process  Existing problems  None | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |