Università della Svizzera italiana

Faculty of Informatics

Operating Systems

PintOS: Report Project **3**

Group 12

Michelangelo Bettini, Leonardo Birindelli,Alessandro della Flora

Spring Semester, March 26, 2024

Report Instructions

- Please report **all** changes, even if minor, that you did to complete the project.
- You have to list all files that have been modified, and for each of them, list all functions/structs that have been modified or added (clearly stating "modified"/"added"). Then add a brief explanation or motivation for all those changes.
- A single report is required for each group. For the first individual project, each student submits a report together with the source code files that were changed.

1 Files changed

- pintos/threads/thread.h
- pintos/threads/thread.c

2 Changes

pintos/threads/thread.h

- #define NICE_MIN -20 (*added*):
  (ln. 27) represents the minimum value for the field nice
- #define NICE_MAX 20 (*added*) :
  (ln. 28) represents the maximum value for the field nice

- #define NICE_INIT 0 (*added*):
  (ln. 29) represents the initial value for the field nice

- struct thread (*implemented*):
  It has been added to the struct two new fields : `int nice` (ln. 95) represents the niceness value of a thread ; `FPReal recent_cpu` (ln. 96) represents how much cpu the thread has used recently expressed in Fixed-Point real.

pintos/threads/thread.c

- FPReal load_avg (*added*):
  (ln. 46) it is a global variable that represent the average system load in number of ready and running threads per second in the last minute

- void thread_init(void) (*modified*):
  (ln. 102) added the initilization of the global variable `load_avg` (ln. 110)

- void thread_tick(void) (*modified*):
  (ln. 137) the function has been modified in a way that implements the advanced scheduler. For the current thread, it increments by 1 the `recent_cpu` value on every tick if the one is not idle, subsequently it updates the `load_avg` based on all the `recent_cpu` value of all ready and running threads and also updates the value of `recent_cpu` depending on the value of the system load average every second. In conclusion, the function updates the priority level of each thread only every 4 ticks.

- tid_t thread_create(const char *name, int priority,thread_func *function, void *aux) (*modified*):
  (ln. 198) the function has been modified in a way that makes the current thread yield if the created thread has an higher priority than the actual one

- bool compare_priority(const struct list_elem *a, const struct list_elem *b, void *aux) (*added*):
  (ln. 257) auxiliary function used to sort threads inside the waiting queue based on the highest priority value

- void thread_unblock(struct thread *t) (*modified*):
  (ln. 291) the function has been modified in a way that inserts the given thread in the ordered ready queue based on its priority.

- void thread_yield(void) (*modified*):
  (ln. 341) the function has been modified in a way that inserts the current thread in the ordered ready queue based on its priority.

- void thread_set_priority(int new_priority) (*modified*):
  (ln. 394) the function has been modified in a way that, if it is used the priority scheduler, updates the current thread priority with the given one and makes the current thread yield if the new priority is not the highest.

- void thread_set_nice(int nice) (*implemented*):
  (ln. 413) the function has been implemented in a way that if the advanced scheduler is set and given nice value is acceptable, it updates the current thread nice field, recalculates its priority based on that and yield if the new priority is not the highest anymore.

- int thread_get_nice(void) (*implemented*):
  (ln. 426) the function has been implemented to return the current thread nice value if the advanced scheduler is set.

- int thread_get_load_avg(void) (*implemented*):
  (ln. 434) the function has been implemented in a way that returns 100 times the system load average if the advanced scheduler is set.

- int thread_get_recent_cpu(void) (*implemented*):
  (ln. 441) the function has been implemented in a way that returns 100 times the current thread's `recent_cpu` value if the advanced scheduler is set.

- static void init_thread(struct thread *t, const char *name, int priority) (*modified*):
  (ln. 518) the function has been modified in a way that if the advanced scheduler is set, the given priority is not assigned to the given thread. In both scheduler options, the `recent_cpu` field of the given thread is initilized.

- void new_priority_mlfqs(struct thread *t) (*implemented*):
  (ln. 642) auxiliary function used to adjust the priority level of the given thread which calculation depends on the initial value of that threat nice field. If the given thread has the priority updated, it is in the ready state and it is different from the current thread, then it inserted once again inside the ready list according to its new priority level.

- bool not_highest_priority(void) (*implemented*):
  (ln. 663) auxiliary function used to check if the current thread has not the highest priority inside the ready queue of threads.

- void new_load_average(void) (*implemented*):
  (ln. 672) auxiliary function used to compute the new system load in number of ready and running threads per second in the last minute.

- void new_recent_cpu(struct thread *t) (*implemented*):
  (ln. 685) auxiliary function used to compute the new recent cpu value for the given thread.