Project Four Report
Introduction to Operating Systems
New Beginnings Spring 2018

Gavin Megson

20 May 2018

Description

For this assignment, I implemented a multi-level feedback queue (MLFQ) for scheduling processes. It involves multiple priorities for processes, each with its own RUNNABLE process list. Every process has a budget; after it is used up, the process is demoted one priority level, and put in the appropriate list. To prevent starvation, processes are periodically promoted one level. Within a priority, round-robin scheduling is still enforced. In addition, previous console commands were updated to display priorities properly.

Deliverables

In general, a MLFQ will allow interactive processes to stay at a higher priority due to their voluntary giving up time while still preventing starvation due to periodic promotion. When adjusting a process into or out of a ready list, care must be taken to ensure it is at the proper priority.

The following features were added to xv6:

- The `RUNNABLE` list has been modified to consist of multiple lists, one for each priority set by the `MAXPRIO` definition. The process table now has an array of pointers to the heads and tails of each `RUNNABLE` priority.
- A new system command `setpriority` was created to set the priority of a process and change its process state list, appending it to the end. It also resets its budget.
- Periodically, the scheduler will update the priority of every process, according to the constant number of ticks defined by the user and kept in the process table. If the priority of a process is greater than 0 (0 is the highest priority), the priority is promoted (made one closer to 0). The budget is not affected.
- Every time a process is descheduled, the ticks spent in the CPU are subtracted from the budget. If the budget falls to 0 or below, the process is demoted one priority level, unless it is at the lowest priority, and the budget is reset.
- The console commands `Ctrl-P` and `Ctrl-R` have been updated to display the priority of the processes displayed. `Ctrl-R` lists the ready processes by each priority level, and now displays the budget.

Implementation

Unless otherwise noted, all modifications were made in the `proc.c` file.

Priority Queues

- The ready priority list in the process table was changed to an array of lists (lines 14-15). The initialization function for the process lists was changed to initialize each list in the array (lines 1116-1119). The number of priorities is defined by `MAXPRIO` in `param.h` (line 21).
- The `proc` structure in `proc.h` was given a field to track the integer priority (line 84). It is initialized in the `allocproc` function in `proc.c` (line 95) while the process is an embryo.
- All functions which added and removed processes to and from the ready state were altered in `proc.c` to account for the priorities. These were:
  - Function `userinit` lines 189-195
  - Function `fork` lines 286-293

  - Function `scheduler` lines 642-657 (during process promotion) and lines 686-692 (during usual scheduling routine)
  - Function `sched` lines 746-752 (during process demotion)
  - Function `yield` lines 771-781
  - Function `wakeup1` lines 878-883
  - Function `kill` lines 943-952 (when waking a sleeping process)
  - Function `setpriority` lines 1321-1329 (when adjusting the priority of a `RUNNABLE` process)

MLFQ Scheduler

- When the scheduler runs (`proc.c` lines 611-717), it must take the highest priority process. To do this, it loops through all priority ready lists, beginning with 0 and ending with `MAXPRIO`, until one is found. If none are found, the scheduling section is skipped, as before.

Process Promotion

- The number of ticks between each periodic promotion is defined in `param.h` (line 23). The number of ticks to reach until the next promotion is stored in `ptable` as the unsigned integer `PromoteAtTime` (`proc.c` line 35) and is initialized when `userinit` runs (`proc.c` line 160).
- `PromoteAtTime` is updated every time the scheduler function determines it is time (`proc.c` lines 624-672). The scheduler compares the current ticks against the target value (`PromoteAtTime`) before scheduling any processes.
- Every process in the running, ready, and sleep lists has their priorities updated. The budget of each process is left as is.
- Promotion to the tail of the next priority avoids starvation by ensuring every process will eventually be able to run. In the worst case scenario, a process, through inactivity, will eventually be promoted to priority 0, and since round robin scheduling is retained within each priority, it will not fail to run.

Process Demotion

- In `proc.h`, the `proc` structure has a new field called `budget` (line 83), which is initialized to the `BUDGET` defined in `param.h` (line 23) when `allocproc` initializes the process (`proc.c` line 94).
- When function `sched` is called, it decreases the process' budget by the amount of time spent in CPU. If it falls to 0 or below, the budget is reset, the priority is demoted (unless it is at the maximum priority already), and if it is currently on a ready list, the list is changed to the corresponding new priority, if necessary. (`proc.c` lines 741-758)

`setpriority` System Call

- As with all other system calls, the following files were updated to implement the `setpriority` function:
  - `user.h` line 44, added a prototype
  - `syscall.h` line 32, defined the system call number
  - `syscall.c` line 118, created function prototype; line 156, added entry in system call function dispatch table;
  - `sysproc.c` lines 189-198, implemented main function

  - `proc.c` lines 1272-1369, implemented system function

- The `setpriority` system call in `proc.c` checks whether the given arguments are within the proper ranges (0 ¡= priority ¡= `MAXPRIO`); if they aren't, the function returns -1. It then acquires a lock and traverses the running list, each ready list from 0 to `MAXPRIO`, and sleep list, attempting to match the given PID with each process in the lists. If it matches, and the priority is not already at the correct level, the priority and budget are both updated. If the priority is already at the correct level, the budget is still updated. If the process was found on a ready list, the process is removed from that list and placed on the new priority list in the usual way. If no process is found, `setpriority` returns -1.

Updated Commands

- The header for the `Ctrl-P` command was updated to include the priority field for each process (`proc.c` line 1016). The `procdump` function was `amended` to display the priority (`procdump.c` line 20).

- The `Ctrl-R` command now loops through every ready list from 0 to `MAXPRIO` and displays a line for each list (`proc.c` lines 1190-1212). It has been updated to display each process' budget alongside the PID.

Testing

Single Priority Level Round Robin

With `MAXPRIO` set to 0, I will run test 1 in `p4test`.

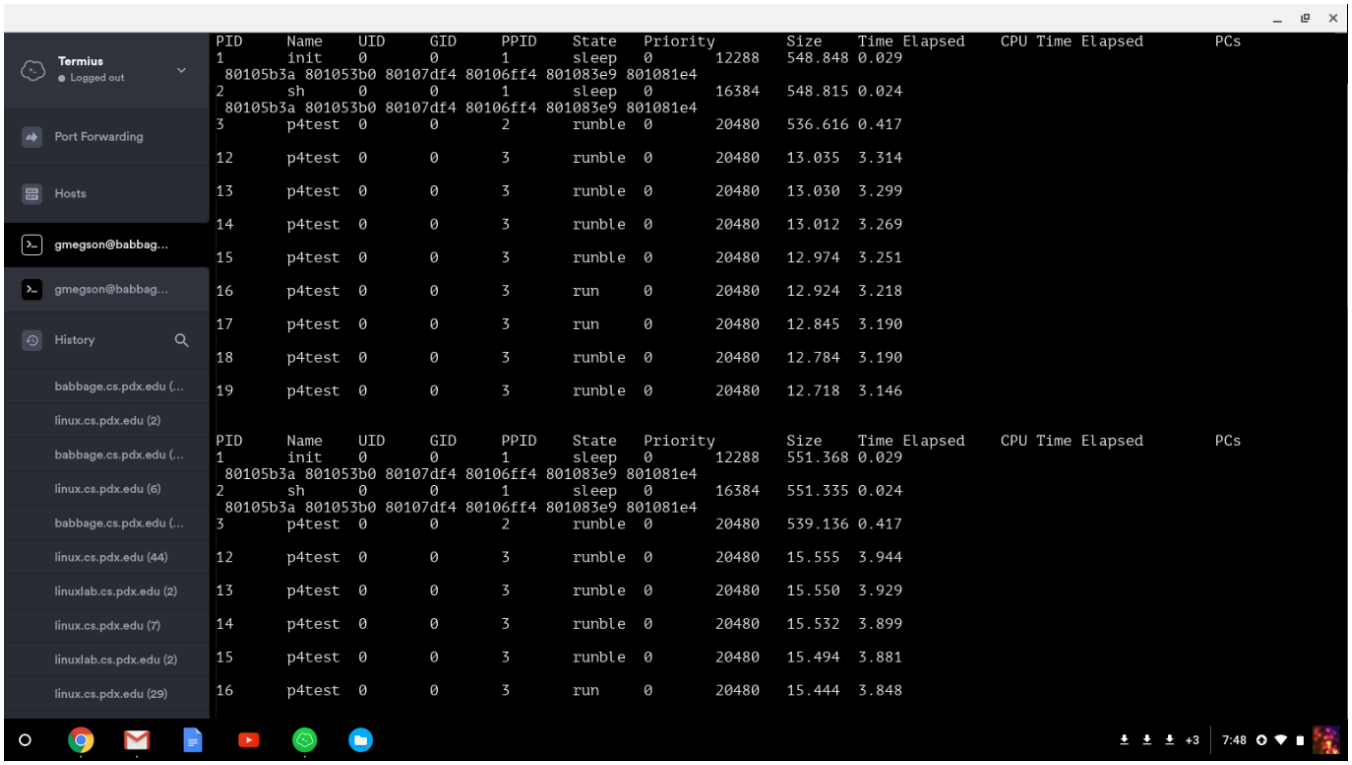

Figure 1:

Through several rounds of scheduling, the priorities update by approximately the same amount of CPU time each round. (Only two rounds are shown here.)

This test `PASSES`.

Changing Priority Level in Sleeping and Running Processes

With MAXPRIO set to 3, I will run test 2 in p4test.
After running the test, the process immediately panicked when calling the yield function. Somehow, the newly formed process is calling yield while not being on the running list. I was unable to debug this error.

This test `FAILS`.

Project update: After receiving peer feedback, I was able to debug this error. I was inconsistent in how I updated the budget after a promotion.

Budget Reset After Demotion

With `MAXPRIO` set to 3 and `BUDGET` set to 2500, I will run test 3 in p4test.

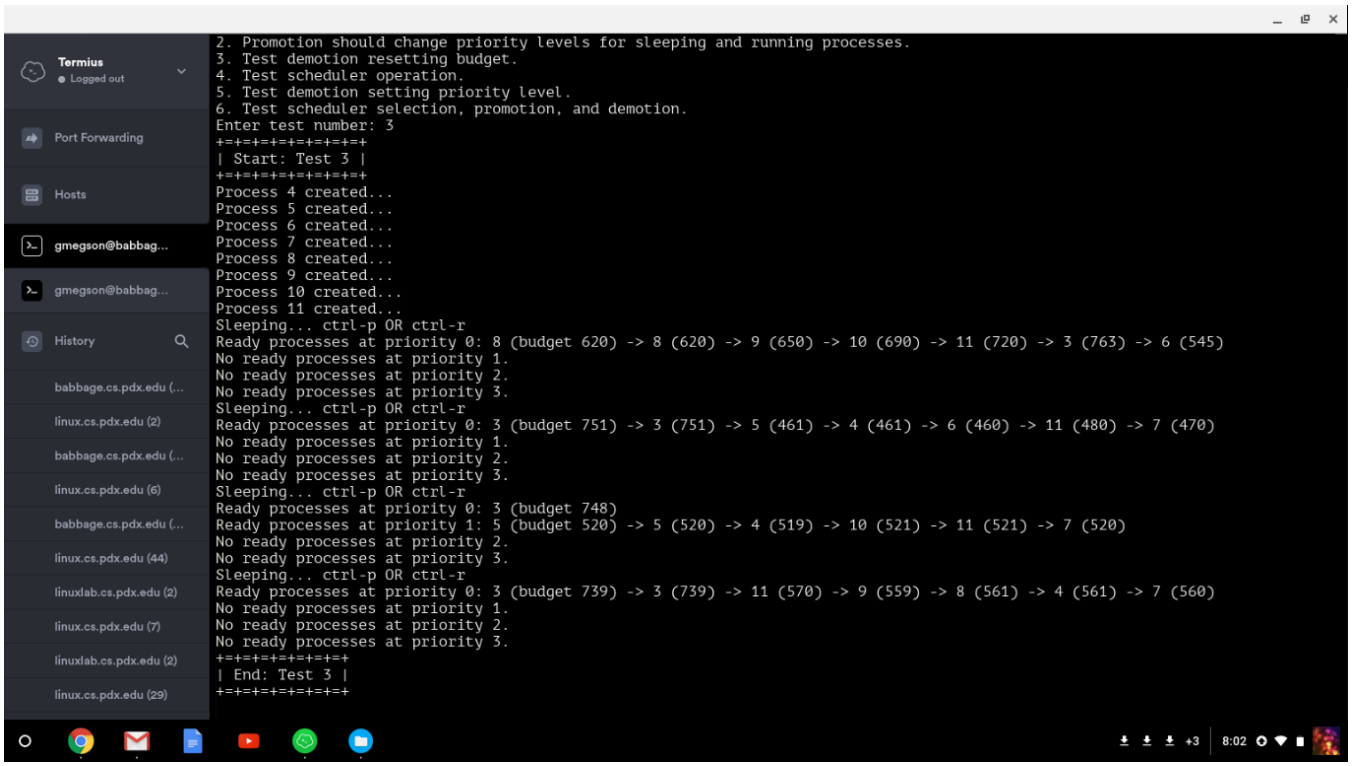

Figure 2:

After several seconds of running, the processes' budgets appear at different numbers, some higher than before. This indicates the budgets are being reset at some point.

This test `PASSES`.

Test 4

With `MAXPRIO` set to 2 and `BUDGET` set to 2500, I will run test 4 in p4test.
Unfortunately, the yield panic occurs again.

This test `FAILS`.

Project update: After receiving peer feedback, I was able to debug this error. It had the same root cause as the previous error.

Test 5

With `MAXPRIO` set to 6 and `BUDGET` set to 2500, I will run test 5 in p4test.
Test 5 also fails upon the yield panic.

This test `FAILS`.

Project update: After receiving peer feedback, I was able to debug this error. It had the same rood cause as previous errors.

Test 6

With `MAXPRIO` set to 0 and `BUDGET` set to 2500, I will run test 6 in p4test.

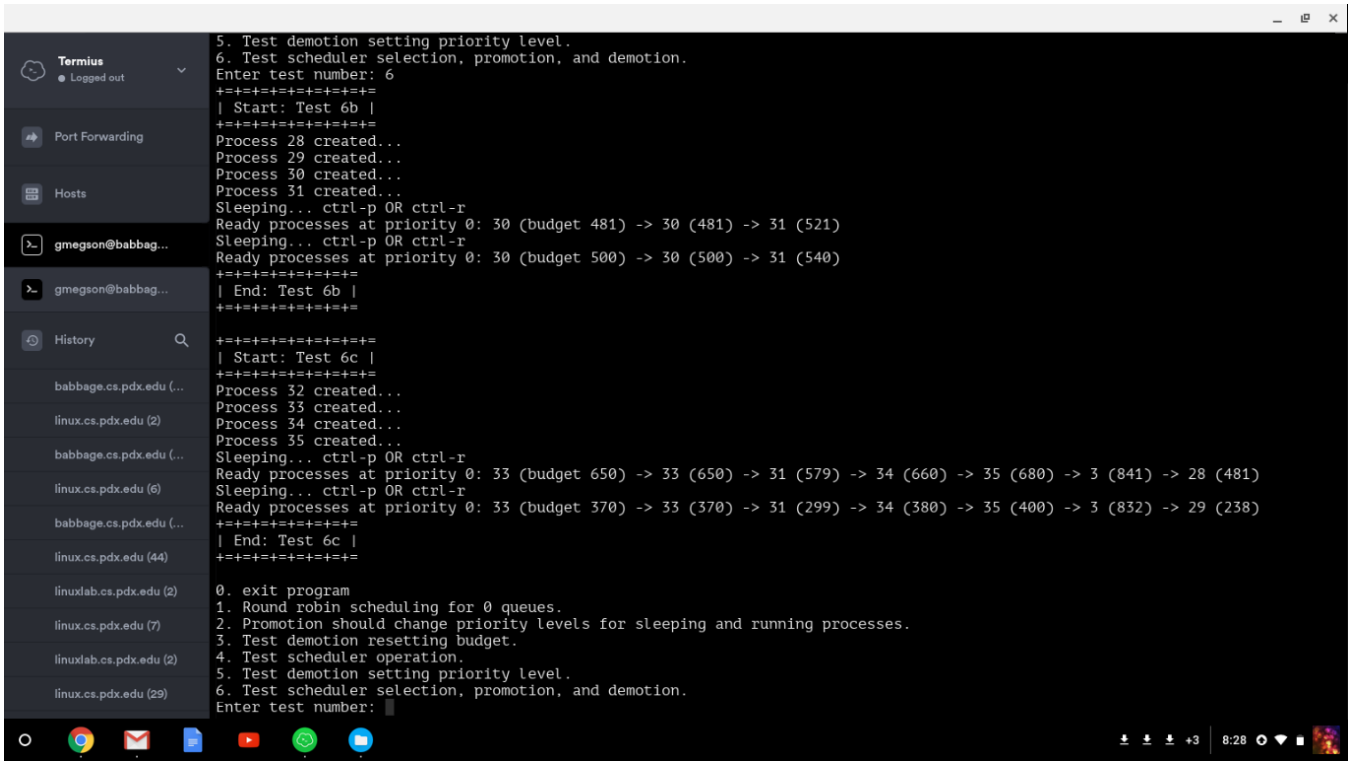

Figure 3:

In every step called by the process, the newly created processes all have their budgets descended by the same amount. This implies the scheduler is enforcing round robin scheduling.

This test `PASSES`.

Updated Commands

Previous tests indicated correct budget readings for `Ctrl-P` and `Ctrl-R`.