CS302 Operating System: Pintos GDB

Name: 傅伟堡(Weibao Fu)

SID: 11812202

Make Check

After run the command `make & make check`, we find that there is a testcase fails.

```
 40    37.56 >>> 29.88    Too big, by 4.18.
 42    37.56 >>> 30.87    Too big, by 3.19.
 44    37.56 >>> 31.84    Too big, by 2.22.
 46    37.56 >>> 32.77    Too big, by 1.29.
 48    37.95 >>> 33.67    Too big, by 0.78.
```

```
pass tests/threads/mlfqs-block
pass tests/threads/alarm-priority
pass tests/threads/mlfqs-load-1
FAIL tests/threads/mlfqs-load-60
pass tests/threads/mlfqs-load-avg
pass tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
pass tests/threads/mlfqs-nice-2
pass tests/threads/mlfqs-nice-10
pass tests/threads/mlfqs-block
1 of 10 tests failed.
```

Configuration

In `pintos-GDB/src/utils/pintos-gdb`, change the `GDBMACROS=../../misc/gdb-macros`

Run

Step 1: Open a terminal and run the command `pintos -v --gdb -- -q -mlfqs run mlfqs-load-60`.

Step 2: Open another terminal and run the command `pintos-gdb kernel.o`

Step 3: Run the command `debugpintos` in the second terminal.

Step 4: Then use `c` (continue for short) to run the test

Step 5: Then hit `crtl+c`, and use command `bt` to look what has happened before.

```
pintos-debug: dumping backtrace of thread 'main' @0xc000e000
#0  intq_full (q=q@entry=0xc00348c0 <txq>) at ../../devices/intq.c:31
#1  0xc0026211 in intq_putc (q=q@entry=0xc00348c0 <txq>, byte=byte@entry=108 'l') at ../../devices/intq.c:64
#2  0xc0024ace in serial_putc (byte=byte@entry=108 'l') at ../../devices/serial.c:125
#3  0xc002a7ad in putchar_have_lock (c=c@entry=108 'l') at ../../lib/kernel/console.c:189
#4  0xc002a7cd in vprintf_helper (c=108 'l', char_cnt_=0xc000ee8c) at ../../lib/kernel/console.c:178
#5  0xc0026b01 in format_string (string=string@entry=0xc0007d4c "mlfqs-load-60", length=13, c=c@entry=0xc000ee40,
    output=output@entry=0xc002a7ba <vprintf_helper>, aux=aux@entry=0xc000ee8c) at ../../lib/stdio.c:569
#6  0xc00271bc in __vprintf (format=<optimized out>, format@entry=0xc002ebd8 "(%s) ", args=0xc000eec8 "$", args@entry=0xc000eec4 "L}",
    output=output@entry=0xc002a7ba <vprintf_helper>, aux=aux@entry=0xc000ee8c) at ../../lib/stdio.c:301
#7  0xc002a8c0 in vprintf (format=format@entry=0xc002ebd8 "(%s) ", args=args@entry=0xc000eec4 "L}") at ../../lib/kernel/console.c:131
#8  0xc0026b55 in printf (format=format@entry=0xc002ebd8 "(%s) ") at ../../lib/stdio.c:85
#9  0xc002a971 in msg (format=format@entry=0xc002ef88 "After %d seconds, load average=%d.%02d.") at ../../tests/threads/tests.c:54
#10 0xc002afbe in test_mlfqs_load_60 () at ../../tests/threads/mlfqs-load-60.c:138
#11 0xc002a9ca in run_test (name=name@entry=0xc0007d4c "mlfqs-load-60") at ../../tests/threads/tests.c:39
#12 0xc00201b3 in run_task (argv=0xc0032fc8 <argv+8>) at ../../threads/init.c:290
#13 0xc0020738 in run_actions (argv=0xc0032fc8 <argv+8>) at ../../threads/init.c:340
#14 main () at ../../threads/init.c:133
```

Step 6: We can use `l *address` to check a few lines of code around address.

Step 7: Also, `btthreadlist &all_list allelem` is helpful to us, it iterates through the list of threads and prints the backtrace for each thread.

```
pintos-debug: dumping backtrace of thread 'load 48' @0xc0134000
---Type <return> to continue, or q <return> to quit---
#0  schedule () at ../../threads/thread.c:671
#1  0x00000092 in ?? ()

pintos-debug: dumping backtrace of thread 'load 49' @0xc0135000
#0  schedule () at ../../threads/thread.c:671
#1  0x00000092 in ?? ()

pintos-debug: dumping backtrace of thread 'load 50' @0xc0136000
#0  schedule () at ../../threads/thread.c:671
#1  0x00000092 in ?? ()
```

Step 8: We find that there is some mistakes in calculating `load_avg`. According to the formula, the value of `load_avg` depends on `recent_cpu`. In this case, we check the process of calculating `load_avg` and `recent_cpu`, and we find mistakes.

Before modifying:

```
 t->recent_cpu = FP_ADD_MIX(FP_DIV( FP_MULT ( FP_MULT_MIX(load_average, 2), t-
>recent_cpu), FP_ADD_MIX ( FP_MULT_MIX (load_average, 2), 1)) , t->nice);
```

After modifying:

```
t->recent_cpu = FP_ADD_MIX (FP_MULT (FP_DIV (FP_MULT_MIX (load_average, 2),
FP_ADD_MIX (FP_MULT_MIX (load_average, 2), 1)), t->recent_cpu), t->nice);
```

After modifying the codes, we can pass all test cases.

```
pass tests/threads/mlfqs-block
pass tests/threads/alarm-priority
pass tests/threads/mlfqs-load-1
pass tests/threads/mlfqs-load-60
pass tests/threads/mlfqs-load-avg
pass tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
pass tests/threads/mlfqs-nice-2
pass tests/threads/mlfqs-nice-10
pass tests/threads/mlfqs-block
All 10 tests passed.
```