CS232 LAB ASSIGNMENT 5

Pintos Project 1 PART1

Anugunj Naman: 1801022

Diali Kundu: 1801055

Project environment

Centos 7.6 Operating System Environment, PintoOS, gdb debugging tool.

Busy Waiting Analysis:

Timer_sleep analysis function, it first looks timer_ticks function call:

```
69 /* Returns the number of timer ticks since the OS booted. */
70 int64_t
71 timer_ticks (void)
72 {
73   enum intr_level old_level = intr_disable ();
74   int64_t t = ticks;
75   intr_set_level (old_level);
76   return t;
77 }
```

Find intr_disable, trying to find out the return value:

```
102 /* Disables interrupts and returns the previous interrupt status. */
103 enum intr_level
104 intr_disable (void)
105 {
106   enum intr_level old_level = intr_get_level ();
107
108   /* Disable interrupts by clearing the interrupt flag.
109      See [IA32-v2b] "CLI" and [IA32-v3a] 5.8.1 "Masking Maskable
110      Hardware Interrupts". */
111   asm volatile ("cli" : : : "memory");
112
113   return old_level;
114 }
```

Close the terminal assembler function, and returns to a state before the interrupt disable interrupts.

In other words, enum intr_level old_level = intr_disable (); Close and save the state before the interruption, intr_set_level (old_level); interrupt status prior to recovery. The middle of these two statements is an atomic operation.

According to previous analysis of timer_sleep, we know: thread_yield, if the current thread is not idle threads, it will be inserted into the end of the queue, the state becomes THREAD_READY. Therefore, the thread will continue to run in the ready queue and queue switch, continue to consume CPU resources, resulting in problems such as busy waiting. So, we will consider the thread is blocked, the thread remaining recording time ticks_blocked is blocked, and detected by the clock interrupt status

for all threads, each of the corresponding decrement ticks_blocked 1, 0 if the corresponding thread wake-up.

Add members thread.h in:

```
92
93    /* Solution Code */
94      int64_t ticks_blocked;              /* Ticks that the thread need to be
   blocked */
95
```

Join initialization statement thread.c of thread_create in:

```
196
197   /* Solution Code */
198   t->ticks_blocked = 0;
```

In order to be able to traverse all the time clock interrupt threads, we use thread_foreach function:

```
349 void
350 thread_foreach (thread_action_func *func, void *aux)
351 {
352   struct list_elem *e;
353
354   ASSERT (intr_get_level () == INTR_OFF);
355
356   for (e = list_begin (&all_list); e != list_end (&all_list);
357        e = list_next (e))
358     {
359       struct thread *t = list_entry (e, struct thread, allelem);
360       func (t, aux);
361     }
362 }
```

Using this function timer.c in timer_interrupt function in:

```
184 /* Timer interrupt handler. */
185 static void
186 timer_interrupt (struct intr_frame *args UNUSED)
187 {
188   /* Solution Code */
189   thread_foreach(checkInvoke, NULL);
190
191   ticks++;
192   thread_tick ();
193 }
```

Statement checkInvoke function in thread.h, the realization in thread.c in:

```
77 /* Solution Code */
78 void
79 checkInvoke(struct thread *t, void *aux UNUSED)
80 {
81   if (t->status == THREAD_BLOCKED && t->ticks_blocked > 0)
82   {
83     --t->ticks_blocked;
84     if (t->ticks_blocked == 0)
85       thread_unblock(t);
86   }
87 }
```

Last Modified timer_sleep function:

```
89 void
90 timer_sleep (int64_t ticks)
91 {
92   /*
93   int64_t start = timer_ticks();
94   ASSERT (intr_get_level () == INTR_ON);
95   while (timer_elapsed(start) < ticks)
96     thread_yield();
97   */
98
99   /* Solution Code */
100
101  ASSERT (intr_get_level () == INTR_ON);
102  enum intr_level old_level = intr_disable();
103
104  /* Blocks current thread for ticks */
105  thread_current()->ticks_blocked = ticks;
106  thread_block();
107
108  intr_set_level(old_level);
109 }
```

That is a long time to block the current thread ticks, this operation is atomic. Try to make check, this should be found by the alarm-negative and did not pass the alarm-zero, both looking at the code of the test program, the parameters found timer_sleep -100 and 0, respectively, while the program is not required to collapse. So, we add conditional:

```
 99   /* Solution Code */
100
101  /* For alarm-negative && alarm-zero */
102  if (ticks <= 0) return;
103
104  ASSERT (intr_get_level () == INTR_ON);
105  enum intr_level old_level = intr_disable();
106
107  /* Blocks current thread for ticks */
108  thread_current()->ticks_blocked = ticks;
109  thread_block();
110
111  intr_set_level(old_level);
```

Success.