Project 2- Process scheduling

Schedule.c Modifications

The following lines of `/usr/src/servers/sched/schedule.c` were modified to simulate CTSS:

Schedule.c BEFORE

```
102     if (rmp->priority < MIN_USER_Q) {
103         rmp->priority += 1; /* lower priority */
104     }
```

Schedule.c AFTER

```
102     /*PROJECT 2 MODIFICATION
103       Set lowest priority level to be 14 by replacing MIN_USER_Q with 14*/
104     if (rmp->priority < 14) {
105
106         rmp->priority += 1;/* lower priority */
107
108         /* PROJECT 2 MODIFICATION
109            The following line will double time quantum for each lower queue*/
110         rmp->time_slice = 2*rmp->time_slice;
111     }
```

Inside the `do_noquantum(message *m_ptr)` function `MIN_USER_Q` was replaced with `14` to set the lowest priority level that a process can be placed after running out of quantum to be 14.

The line `rmp->time_slice = 2*rmp->time_slice;` was added to double the time quantum as a progress is sent to a lower queue so that each lower queue will have twice the time quantum of the queue one priority level above it.

Schedule.c BEFORE

```
173     if (rmp->endpoint == rmp->parent) {
174         /* We have a special case here for init, which is the first
175            process scheduled, and the parent of itself. */
176         rmp->priority   = USER_Q;
177         rmp->time_slice = DEFAULT_USER_TIME_SLICE;
```

```
193     case SCHEDULING_START:
194         /* We have a special case here for system processes, for which
195          * quanum and priority are set explicitly rather than inherited
196          * from the parent */
197         rmp->priority   = rmp->max_priority;
198         rmp->time_slice = (unsigned) m_ptr->SCHEDULING_QUANTUM;
199         break;
```

```
201     case SCHEDULING_INHERIT:
202         /* Inherit current priority and time slice from parent. Since there
203          * is currently only one scheduler scheduling the whole system, this
204          * value is local and we assert that the parent endpoint is valid */
205         if ((rv = sched_isokendpt(m_ptr->SCHEDULING_PARENT,
206                 &parent_nr_n)) != OK)
207             return rv;
208
209         rmp->priority = schedproc[parent_nr_n].priority;
210         rmp->time_slice = schedproc[parent_nr_n].time_slice;
211         break;
```

Schedule.c AFTER

```
180     if (rmp->endpoint == rmp->parent) {
181         /* We have a special case here for init, which is the first
182            process scheduled, and the parent of itself. */
183
184
185         /* PROJECT 2 MODIFICATION
186            Replaced initial USER_Q with 7 and initial user time quantum with 1ms*/
187         rmp->priority   = 7 ; //USER_Q
188         rmp->time_slice = 1; //DEFAULT_USER_TIME_SLICE
```

```
204     case SCHEDULING_START:
205         /* We have a special case here for system processes, for which
206          * quanum and priority are set explicitly rather than inherited
207          * from the parent */
208
209
210         /* PROJECT 2 MODIFICATION
211            Replaced initial rmp->max_priority with 7 and initial
212            (unsigned) m_ptr->SCHEDULING_QUANTUM with 1ms*/
213         rmp->priority   = 7; //rmp->max_priority ;
214         rmp->time_slice = 1; //(unsigned) m_ptr->SCHEDULING_QUANTUM;
215         break;
```

```
217     case SCHEDULING_INHERIT:
218         /* Inherit current priority and time slice from parent. Since there
219          * is currently only one scheduler scheduling the whole system, this
220          * value is local and we assert that the parent endpoint is valid */
221         if ((rv = sched_isokendpt(m_ptr->SCHEDULING_PARENT,
222                 &parent_nr_n)) != OK)
223             return rv;
224
225         /* PROJECT 2 MODIFICATION
226            Replaced initial schedproc[parent_nr_n].priority with 7 and initial
227            schedproc[parent_nr_n].time_slice with 1ms*/
228         rmp->priority = 7; //schedproc[parent_nr_n].priority;
229         rmp->time_slice = 1; //schedproc[parent_nr_n].time_slice;
230         break;
```

Inside the `do_start_scheduling(message *m_ptr)` function `USER_Q` for the case of init, `rmp->max_priority` in the case for an explicitly set priority, and `schedproc[parent_nr_n].priority` in the case of inheriting were replaced with `7` to set the initial priority level to be 7.
`DEFAULT_USER_TIME_SLICE` for the case of init, `m_ptr->SCHEDULING_QUANTUM` in the case for an explicitly set quantum, and `schedproc[parent_nr_n].priority` in the case of inheriting were replaced with `1` to set the initial quantum for level 7 to be 1ms.

Schedule.c BEFORE

```
335 void init_scheduling(void)
336 {
337     balance_timeout = BALANCE_TIMEOUT * sys_hz();
338     init_timer(&sched_timer);
339     set_timer(&sched_timer, balance_timeout, balance_queues, 0);
340 }
```

Schedule.c AFTER

```
351  void init_scheduling(void)
352  {
353
354      /* PROJECT 2 MODIFICATION
355         Commented out code that schedules balancing to stop balancing*/
356
357      /*
358      balance_timeout = BALANCE_TIMEOUT * sys_hz();
359      init_timer(&sched_timer);
360      set_timer(&sched_timer, balance_timeout, balance_queues, 0);
361      */
362  }
```

Commented out the contents of `init_scheduling(void)` function that handled the balance scheduling to stop the scheduler from balancing.

Testing

To test I added a function key F2 to print out the endpoint, priority and quantum as required for the demonstration. The initial quantum was changed from 1ms to 1000ms so that I could see the quantum changing. Then I ran a program loop and used the function F2 to see if it the quantum and priority where changing correctly.

To add the function key dmp.c, dmp_kernel.c, proto.h where edited in /usr/src/servers/is/

In dmp.c

```
18  } hooks[] = {
19      { F1,   proctab_dmp, "Kernel process table" },
20
21      { F2, user_dmp, "User process loop"}, // PROJECT 2 MODIFICATION
22
23      { F3, image_dmp, "System image" },
24      { F4, privileges_dmp, "Process privileges" },
```

In dmp_kernel.c

```
391  /*===========================================================================*
392   *     PROJECT 2 MODIFICATION     *
393   *          user_dmp              *
394   *===========================================================================*/
395  void user_dmp()
396  {
397    /* Proc table dump */
398    register struct proc *rp;
399    static struct proc *oldrp = BEG_USER_ADDR;
400    int r;
401
402    /* First obtain a fresh copy of the current process table. */
403    if ((r = sys_getproctab(proc)) != OK) {
404      printf("IS: warning: couldn't get copy of process table: %d\n", r);
405      return;
406    }
407    char user_process[PROC_NAME_LEN] = "loop"; // can be changed to your program name
408    for (rp = oldrp; rp < END_PROC_ADDR; rp++) {
409      if (isemptyp(rp))
410        continue;
411      if(strcmp (user_process,rp->p_name) == 0)
412      {
413        printf("Loop: End-point: %d, Priority: %d; Quantum: %d;\n", rp->p_endpoint, rp->p_priority, rp->p_quantum_size_ms);
414      }
415      }
416  }
```

In proto.h

```
12  /* dmp_kernel.c */
13  void proctab_dmp(void);
14
15  void user_dmp (void); // PROJECT 2 MODIFICATION
16
17  void procstack_dmp(void);
```

Loop.c

```c
#include <stdio.h>
#include <unistd.h>
#include <math.h>

int main()
{
    double i = 0;
    double para = 5.0;
    int j = 0;

    while(1){

        i = exp(para);
        para = para + 0.1;
    }
    return 0;
}
```

Lessons from this Project

In this project I learned how the default scheduling works in Minix and how to change it to simulate CTSS. I had to find how to edit the quantum, the default minimum priority and the initial priority queue. I also learned about adding function keys to use for debug dumps and how the information server works. I was able to correctly add a function to display debugging information for when I edited the scheduling in Minix.

Biggest Challenge

The biggest challenge was by far finding which where the correct variables to edit to correctly change the initial priority queue and the initial quantum. I had to change a couple different values that I thought where the correct ones back and forth until I found just the right ones. Adding the function key greatly helped in figuring out if it was working correctly or not. Stopping the balancing was also a bit of a challenge since I wasn't sure which code to comment out. I followed the function to where it was first called and was able to correctly stop the balancing.