Assignment 2 Report

Anoop (2015CS10265)

March 29, 2018

Part 1 - Add and print priority of the process

Modify sys_ps to print priority

To implement this, process structure has been changed in **proc.h** file [Line 44]. Default priority has been set to 5 in **proc.c** file [Line 90]. To print the priority while listing processes, changes are made to **ps** function in **proc.c** file.

Add sys_setpriority system call

Listing 1: sys_setpriority function implementation - sysproc.c

```
125 extern int setpriority(int,int);
126
127 // set priority
128 int
129 sys_setpriority(void)
130 {
131     int pid;
132     int priority;
133
134     if(argint(0, &pid) < 0)
135     return -1;
136     if(argint(1, &priority) < 1 && argint(1, &priority) > 20){
137     cprintf("Error\n");
138     return -1;
139     }
140     return setpriority(pid,priority);
141 }
```

Listing 2: setpriority function implementation - proc.c

```
303 // set priority
304 int
305 setpriority(int pid,int priority)
306 {
307   struct proc *p;
308
309   acquire(&ptable.lock);
```

```
310
311   for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
312     if(p->pid == pid){
313       p->priority = priority;
314       break;
315     }
316   }
317
318   release(&ptable.lock);
319   return pid;
320 }
```

For the `sys_setpriority` system call implementation, a function takes that takes two arguments - PID and Priority and returns pid if successful or -1 on failure is used. `ptable` is locked while setting the priority. It report "Error" on passing invalid priority (> 1 or < 20).

Part 2 - Implement Priority Scheduler

To implement Priority based Round Robin, **scheduler** function has been updated in **proc.c** file. Lines 415 - 423 find the process with highest priority in the ptable. Lines 435 - 447 ensure the Round Robin fashion running of processes.

Part 3 - Starvation

Add sys_getpriority system call

Listing 3: `sys_getpriority` function implementation - `sysproc.c`

```
143 extern int getpriority(int);
144
145 // get priority
146 int
147 sys_getpriority(void)
148 {
149     int pid;
150
151     if(argint(0, &pid) < 0)
152         return -1;
153     return getpriority(pid);
154 }
```

Listing 4: `getpriority` function implementation - `proc.c`

```
322 // get priority
323 int
324 getpriority(int pid)
325 {
326     struct proc *p;
327     int priority = -1;
```

```
328
329     acquire(&ptable.lock);
330
331     for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
332         if(p->pid == pid){
333             priority = p->priority;
334             break;
335         }
336     }
337     release(&ptable.lock);
338     if(priority == -1)
339         cprintf("PID not found");
340     return priority;
341 }
```

For the `sys_getpriority` system call implementation, a function takes that takes two arguments - PID and returns priority if successful or -1 on failure is used. `ptable` is locked while setting the priority. It report "PID not found" on passing PID which does not exist.

Handle starvation

To handle starvation a counter is implemented which counts the number of context switches. It adds 1 to priority after counter reaches 50 and resets counter to 0. This has been implemented in **scheduler** function in **proc_3.c** file [Line 449 - 460].