Savitri

1313619015

Assignment 1 Operating System

CODE MODIFICATION REPORT

Compilation Test

1. makefile (add code, line 3)

```
3. CS333_PROJECT ?= 1
```

System Call Tracing

1. makefile (add code, line 3-4)

```
3. CS333_PROJECT ?= 1
4. PRINT_SYSCALLS ?= 0
```

2. syscall.c (add code, line 184-186)

```
184. #ifdef PRINT_SYSCALLS
185. cprintf ("%s %d \n", syscallnames[num], syscalls[num]());
186. #endif
```

Date System Call

1. user.h (add code, line 29-31)

```
29. #ifdef CS333_P1
30. int date(struct rtcdate*);
31. #endif // CS333_P1
```

2. usys.s (add code, line 33)

```
33. SISCALL(date)
```

3. syscall.h (add code, line 25)

```
25. #define SYS_date    SYS_halt+1
```

4. syscall.c
   - add code, line 109-111

```
109. #ifdef CS333_P1
```

```
110. extern int sys_date(void);
111. #endif // CS333_P1
```

- add code, line 138-140

```
138. #ifdef CS333_P1
139. [SYS_date]    sys_date,
140. #endif // CS333_P1
```

- add code, line 168-170

```
169. #ifdef PDX_XV6
170.  [SYS_date]    "date",
171. #endif // CS333_P1
```

5. sysproc.c (add code, line 101-109)

```
101. int
102. sys_date(void)
103. {
104.  struct rtcdate *d;
105.  if(argptr(0, (void*)&d, sizeof(struct rtcdate)) < 0)
106.    return -1;
107.  cmostime(d);
108.  return 0;
109. }
```

6. date.c (add code, line 38-42)

```
38.  //r.hour %= 12;
38.  //if (r.hour == 0) r.hour = 12;
40.
41.  printf(1, "%s %s  %d %s%d:%s%d:%s%d UTC %d\n", days[day], months[r.mo
nth], r.day,
42.    PAD(r.hour), r.hour, PAD(r.minute), r.minute, PAD(r.second), r.seco
nd, r.year, s);
```

7. makefile (delete #, line 16)

```
16. CS333_UPROGS += _date
```

Process Information

1. proc.h (add code, line 52)

```
52. uint start_ticks;
```

2. proc.c
   - add code, line 152

```
152. p->start_ticks = ticks;
```

   - delete code, line 567

```
567.  cprintf("TODO for Project 1, delete this line and implement procd
umpP1() in proc.c to print a row\n");
```

   - add code, line 567-579

```
567. int second;
568.  int milisecond;
569.    milisecond=ticks-p->start_ticks;
570.    second=milisecond/1000;
571.    milisecond=milisecond%1000;
572.  char* num="";
573.    if(milisecond <100 && milisecond >=10)
574.      num="0";
575.  if(milisecond <10)
576.    num="00";
577.
578.  cprintf("%d\t%s\t%s%d.%s%d\t%s\t%d\t",
579.    p->pid, p->name, "     ", second, num, milisecond, states[p-
>state], p->sz);
```