Nama : Ikhsan Agil Kusuma

NIM : 1313619005

Prodi : Ilmu Komputer 2019

Modification Report

Project 2

A. Makefile

Line 3

Makefile

```
# Set flag to correct CS333 project number: 1, 2, ...
# 0 == original xv6-pdx distribution functionality
CS333_PROJECT ?= 2
PRINT_SYSCALLS ?= 0
CS333_CFLAGS ?= -DPDX_XV6
ifeq ($(CS333_CFLAGS), -DPDX_XV6)
CS333_UPROGS += _halt _uptime
endif

ifeq ($(PRINT_SYSCALLS), 1)
CS333_CFLAGS += -DPRINT_SYSCALLS
endif

```

B. defs.h

1. Line 1 - 3

defs.h

```
#ifdef CS333_P2
#include "uproc.h"
#endif
struct buf;
struct context;
struct file;
struct inode;
struct pipe;
struct proc;
struct rtcdate;
struct spinlock;
struct sleeplock;
struct stat;
struct superblock;
```

2. Line 132 - 134

defs.h

```
int             kill(int);
struct cpu*     mycpu(void);
struct proc*    myproc();
void            pinit(void);
void            procdump(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            setproc(struct proc*);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(void);
void            wakeup(void*);
void            yield(void);

// project 2
#ifdef CS333_P2
int             getprocs(uint max, struct uproc* upTable);
#endif // CS333_P2

```

C. proc.c

1. Line 9 - 11

proc.c

```
1   #include "types.h"
2   #include "defs.h"
3   #include "param.h"
4   #include "memlayout.h"
5   #include "mmu.h"
6   #include "x86.h"
7   #include "proc.h"
8   #include "spinlock.h"
9   #ifdef CS333_P2
10  #include "uproc.h"
11  #endif
```

2. Line 157 - 160

proc.c

```
157   #ifdef CS333_P2 // project2
158     p->cpu_ticks_total = 0;
159     p->cpu_ticks_in = 0;
160   #endif // CS333_P2
161
162   return p;
163 }
164
```

3. Line 254 - 257

proc.c

```
249   np->sz = curproc->sz;
250   np->parent = curproc;
251   *np->tf = *curproc->tf;
252
253   // project 2
254   #ifdef CS333_P2
255   np->uid = curproc->uid;
256   np->gid = curproc->gid;
257   #endif //CS333_P2
258
259   // Clear %eax so that fork returns 0 in the child.
260   np->tf->eax = 0;
```

4. Line 460 - 462

proc.c

```
454   panic("sched running");
455 if(readeflags()&FL_IF)
456   panic("sched interruptible");
457 intena = mycpu()->intena;
458
459 // project 2
460 #ifdef CS333_P2
461 p->cpu_ticks_total += (ticks - p->cpu_ticks_in);
462 #endif // CS333_P2
463
464 swtch(&p->context, mycpu()->scheduler);
465 mycpu()->intena = intena;
466 }
```

5. Line 592 - 636

proc.c

```
591   // project 2
592   uint elapsed_s;
593   uint elapsed_ms;
594
595   elapsed_ms = ticks - p->start_ticks;
596   elapsed_s = elapsed_ms / 1000;
597   elapsed_ms = elapsed_ms % 1000;
598
599   uint elapsed_cpu_s;
600   uint elapsed_cpu_ms;
601   uint ppid;
602   if(p->parent){
603     ppid = p->parent->pid;
604   }
605   else{
606     ppid = p->pid;
607   }
608
609   elapsed_cpu_ms = p->cpu_ticks_total;
610   elapsed_cpu_s = elapsed_cpu_ms / 1000;
611   elapsed_cpu_ms = elapsed_cpu_ms % 1000;
612
613   char* zero = "";
614   if(elapsed_ms < 100 && elapsed_ms >= 10)
615     zero = "0";
616   if(elapsed_ms < 10)
617     zero = "00";
618
619   char* cpu_zero = "";
620   if(elapsed_cpu_ms < 100 && elapsed_cpu_ms >= 10)
621     cpu_zero = "0";
622   if(elapsed_cpu_ms < 10)
623     cpu_zero = "00";
624
625   cprintf(
626     "\n%d\t%s\t%s%d\t%s%d\t%s%d\t%d.%s%d\t%d.%s%d\t%s\t%d\t",
627     p->pid,
628     p->name, "     ",
629     p->uid, "       ",
630     p->gid, "",
631     ppid,
632     elapsed_s, zero, elapsed_ms,
633     elapsed_cpu_s, cpu_zero, elapsed_cpu_ms,
634     state_string,
635     p->sz
636   );
```

D. proc.h

Line 54 - 59

proc.h

```
51    char name[16];               // Process name (debugging)
52    uint start_ticks;
53
54    #ifdef CS333_P2 //project2
55    uint uid;
56    uint gid;
57    uint cpu_ticks_total;
58    uint cpu_ticks_in;
59    #endif // CS333_P2
60  };
61
```

E. ps.c
Line 1 – 56

```
#ifdef CS333_P2
#include "types.h"
#include "user.h"
#include "uproc.h"

#define MAX 16

int
main(void)
{
  struct uproc *proc = malloc(sizeof(struct uproc)*MAX);
  int proc_num = getprocs(MAX, proc);
  printf(1,"PID\tName\t\tUID\tGID\tPPID\tElapsed\tCPU\tState\tSize\n");

  int i;
  for(i = 0; i<proc_num; i++){
    struct uproc current_proc = proc[i];
    uint elapsed_ticks = current_proc.elapsed_ticks;
    uint elapsed_s = elapsed_ticks/1000;
    uint elapsed_ms = elapsed_ticks%1000;

    uint elapsed_cpu_ticks = current_proc.CPU_total_ticks;
    uint elapsed_cpu_s = elapsed_cpu_ticks/1000;
    uint elapsed_cpu_ms = elapsed_cpu_ticks % 1000;

    char* zero = "";
    if(elapsed_ms < 100 && elapsed_ms >= 10)
      zero = "0";
    if(elapsed_ms < 10)
      zero = "00";

    char* cpu_zero = "";
    if(elapsed_cpu_ms < 100 && elapsed_cpu_ms >= 10)
      cpu_zero = "0";
    if(elapsed_cpu_ms < 10)
      cpu_zero = "00";

    printf(
      1,
      "%d\t%s\t\t%d\t%d\t%d\t%d.%s%d\t%d.%s%d\t%s\t%d\n",
      current_proc.pid,
      current_proc.name,
      current_proc.uid,
```

```
    current_proc.gid,
    current_proc.ppid,
    elapsed_s, zero, elapsed_ms,
    elapsed_cpu_s, cpu_zero, elapsed_cpu_ms,
    current_proc.state,
    current_proc.size
   );
 }

 free(proc);
 exit();
}
#endif
```

F. syscall.c

1. Line 112 - 120

```
syscall.c
109   #ifdef CS333_P1
110   // internally, the function prototype must be 'int' not 'uint' for sys_date()
111   extern int sys_date(void);
112   #endif // CS333_P1
113   #ifdef CS333_P2
114   extern int sys_getuid(void);
115   extern int sys_getgid(void);
116   extern int sys_getppid(void);
117   extern int sys_setuid(void);
118   extern int sys_setgid(void);
119   extern int sys_getprocs(void);
120   #endif // CS333_P2
121
```

2. Line 153 - 160

```
syscall.c
151   #endif // CS333_P1
152
153   #ifdef CS333_P2
154   [SYS_getuid]  sys_getuid,
155   [SYS_getgid]  sys_getgid,
156   [SYS_getppid] sys_getppid,
157   [SYS_setuid]  sys_setuid,
158   [SYS_setgid]  sys_setgid,
159   [SYS_getprocs]  sys_getprocs,
160   #endif // CS333_P2
161   };
162
```

3. Line 195 - 202

```
syscall.c
193   #endif // CS333_P1
194
195   #ifdef CS333_P2
196     [SYS_getuid]  "getuid",
197     [SYS_getgid]  "getgid",
198     [SYS_getppid] "getppid",
199     [SYS_setuid]  "setuid",
200     [SYS_setgid]  "setgid",
201     [SYS_getprocs]  "getprocs",
202   #endif // CS333_P2
203   };
204   #endif // PRINT_SYSCALLS
```

G. syscall.h
Line 31 - 36

```
syscall.h
26
27   // project 1
28   #define SYS_date    SYS_halt+1
29
30   // project 2
31   #define SYS_getuid  SYS_date+1
32   #define SYS_getgid  SYS_getuid+1
33   #define SYS_getppid SYS_getgid+1
34   #define SYS_setuid  SYS_getppid+1
35   #define SYS_setgid  SYS_setuid+1
36   #define SYS_getprocs  SYS_setgid+1
37
38
```

H. sysproc.c
Line 116 – 179

```c
#ifdef CS333_P2
int
sys_getuid(void)
{
  return myproc()->uid;
}

int
sys_getgid(void)
{
  return myproc()->gid;
}

int
sys_getppid(void)
{
  if(myproc()->parent == NULL)
    return myproc()->pid;
  else
```

```
    return myproc()->parent->pid;
}

int
sys_setuid(void)
{
  int test;
  if(argint(0, &test)<0)
    return -1;
  if(test < 0 || test >32767)
    return -1;
  else{
    myproc()->uid = test;
    return 0;
  }
}

int
sys_setgid(void)
{
  int test;
  if(argint(0, &test)<0)
    return -1;
  if(test < 0 || test >32767)
    return -1;
  else{
    myproc()->gid = test;
    return 0;
  }
}

int
sys_getprocs(void)
{
  struct uproc *p;
  int max;

  if(argint(0,&max)<0){
    return -1;
  }
  if(argptr(1, (void*)&p, sizeof(struct uproc) * max) < 0)
    return -1;
  return getprocs(max, p);
}
#endif // CS333_P2
```

I. testsetuid.c
Line 1 – 12

testsetuid.c

```c
#ifdef CS333_P2
#include "types.h"
#include "user.h"

int
main(int argc, char *argv[])
{
  printf(1, "***** In %s: my uid is %d\n\n", argv[0], getuid());
  exit();
}
#endif

```

J. time.c
Line 1 – 46

```c
#ifdef CS333_P2
#include "types.h"
#include "user.h"

int main(int argc, char *argv[]){
    if(argc == 1) {
      printf(1, "(null) ran in 0.00\n");
    } else {
      int start = uptime();
      int pid = fork();

      if (pid > 0) {
        pid = wait();
      } else if (pid == 0) {
        exec(argv[1], argv+1);
        printf(1, "ERROR: Unknown Command\n");
        kill(getppid());
        exit();
      } else {
        printf(1, "ERROR: Fork error return -1\n");
      }

      int end = uptime();
      int timelapse = end - start;
      int seconds = timelapse/1000;
      int ms = timelapse%1000;
      char *msZeros = "";

      if (ms < 10) {
        msZeros = "00";
      } else if (ms < 100) {
        msZeros = "0";
```

```
      }

      printf(
        1,
        "%s ran in %d.%s%d\n",
        argv[1],
        seconds,
        msZeros,
        ms
      );
    }
    exit();
}
#endif // CS333_P2
```

K. user.h
Line 33 - 41

```
h user.h
33  #ifdef CS333_P2
34  uint getuid(void);      // UID of the current process
35  uint getgid(void);      // GID of the current process
36  uint getppid(void);     // Process ID of the current process
37
38  int setuid(uint);       // set UID
39  int setgid(uint);       // set GID
40  int getprocs(uint max, struct uproc* table);
41  #endif // CS333_P2
42
```

L. usys.S
Line 33 - 39

```
usys.S
30  SYSCALL(sleep)
31  SYSCALL(uptime)
32  SYSCALL(halt)
33  SYSCALL(date)
34  SYSCALL(getuid)
35  SYSCALL(getgid)
36  SYSCALL(getppid)
37  SYSCALL(setuid)
38  SYSCALL(setgid)
39  SYSCALL(getprocs)
40
```