Nama : Ridwan Syah

NIM : 1313618016

Prodi : Ilmu Komputer 2018

Code Modification Report

Makefile

Line 3 :

```
CS333_PROJECT ?= 2
```

defs.h :

Line 12-14 :

```
#ifdef CS333_P2
struct uproc;
#endif // CS333_P2
```

Line 135-137 :

```
#ifdef CS333_P2
int  getprocs(uint max, struct uproc* table);
#endif // CS333_P2
```

param.h

Line 18-20 :

```
#ifdef CS333_P2
#define DEFAULTID     0  // default ID for UID/GID of process
#endif // CS333_P2
```

proc.c

Line 9-11 :

```
#ifdef CS333_P2
#include "uproc.h"
#endif //CS333_P2
```

Line 155-158 :

```
  #ifdef CS333_P2
  p->cpu_ticks_total = 0;
  p->cpu_ticks_in = 0;
  #endif // CS333_P2
```

Line 195-198 :

```
#ifdef CS333_P2
  p->uid = DEFAULT_UID;
  p->gid = DEFAULT_GID;
#endif
```

Line 264-267 :

```
#ifdef CS333_P2
  np->uid = curproc->uid;
  np->gid = curproc->gid;
#endif // CS333_P2
```

Line 408-410 :

```
#ifdef CS333_P2
      p->cpu_ticks_in = ticks;
#endif // CS333_P2
```

Line 451-453 :

```
#ifdef CS333_P2
  p->cpu_ticks_total += ticks - p->cpu_ticks_in;
#endif // CS333_P2
```

Line 577-602 :

```
#if defined(CS333_P2)
void
procdumpP2P3P4(struct proc *p, char *state_string)
{
  // cprintf("TODO for Project 2, delete this line and implement procdumpP2P3P
4() in proc.c to print a row\n");
  int elapsed = ticks - p->start_ticks;
  uint elapsed_sec = elapsed / 1000;
  uint elapsed_mod = elapsed % 1000;
  int total = p->cpu_ticks_total;
  int total_sec = total/1000;
  int total_mod = total%1000;
  int ppid;
  if(p->parent)
  {
    ppid = p->parent->pid;
  }
  else
  {
    ppid = p->pid;
```

```
  }
  cprintf("%d\t%s\t\t%d\t%d\t%d\t%d.%d\t%d.%d\t%s\t%d\t", p->pid, p->name, p-
>uid, p-
>gid, ppid, elapsed_sec, elapsed_mod, total_sec, total_mod, state_string, p-
>sz);
  return;
}
void
procdumpP2P3P4(struct proc *p, char *state_string)
{
  cprintf("TODO for Project 2, delete this line and implement procdumpP2P3P4()
 in proc.c to print a row\n");
  return;
}
```

Line 967-997 :

```
#ifdef CS333_P2
int
getprocs(uint max, struct uproc* table)
{
  int i = 0;
  struct proc* p;
  acquire(&ptable.lock);
  if(!table || max <= 0){
    release(&ptable.lock);
    return -1;
  }
  for(p = ptable.proc;p < &ptable.proc[NPROC];p++){
    if(i >= max)
      break;
    if(p->state != EMBRYO && p->state != UNUSED){
      table[i].pid = p->pid;
      table[i].uid = p->uid;
      table[i].gid = p->gid;
      table[i].ppid = (!p->parent) ? p->pid:p->parent->pid;
      table[i].elapsed_ticks = ticks - p->start_ticks;
      table[i].CPU_total_ticks = p->cpu_ticks_total;
      table[i].size = p->sz;
      safestrcpy(table[i].state, states[p->state], sizeof(table[i]).state);
      safestrcpy(table[i].name, p->name, sizeof(table[i]).name);
      i++;
    }
  }
  release(&ptable.lock);
  return i;
}
#endif // CS333_P2
```

proc.h

Line 53-58 :

```
#ifdef CS333_P2
  uint uid;                   // UID
  uint gid;                   // GID
  uint cpu_ticks_total;       // For process execution time
  uint cpu_ticks_in;          // For process execution time
#endif
```

syscall.c

Line 113-120 :

```
#ifdef CS333_P2
extern int sys_getuid(void);
extern int sys_getgid(void);
extern int sys_getppid(void);
extern int sys_setuid(void);
extern int sys_setgid(void);
extern int sys_getprocs(void);
#endif // CS333_P2
```

Line 150-157 :

```
#ifdef CS333_P2
[SYS_getuid]   sys_getuid,
[SYS_getgid]   sys_getgid,
[SYS_getppid]  sys_getppid,
[SYS_setuid]   sys_setuid,
[SYS_setgid]   sys_setgid,
[SYS_getprocs] sys_getprocs,
#endif // CS333_P2
```

Line 160 :

```
#if defined(CS333_P1) && defined(PRINT_SYSCALLS)
```

Line 186-193 :

```
#ifdef CS333_P2
  [SYS_getuid]   "getuid",
  [SYS_getgid]   "getgid",
  [SYS_getppid]  "getppid",
  [SYS_setuid]   "setuid",
  [SYS_setgid]   "setgid",
```

```
  [SYS_getprocs] "getprocs".
#endif // CS333_P2
```

syscall.h

Line 26-31 :

```
#define SYS_getuid   SYS_date+1
#define SYS_getgid   SYS_getuid+1
#define SYS_getppid  SYS_getgid+1
#define SYS_setuid   SYS_getppid+1
#define SYS_setgid   SYS_setuid+1
#define SYS_getprocs SYS_setgid+1
```

sysproc.c

Line 113-170 :

```
#ifdef CS333_P2
// Get process UID
uint sys_getuid(void)
{
  return myproc()->uid;
}

// Get process GID
uint sys_getgid(void)
{
  return myproc()->gid;
}

// Get process PPID
uint sys_getppid(void)
{
  if(!myproc()->parent)
    return myproc()->pid;
  else
    return myproc()->parent->pid;
}

// Set Process UID
int sys_setuid(void)
{
  uint uid;
  if(argint(0, (int*)&uid) < 0)
    return -1;
  if(uid < 0 || uid > 32767)
    return -1;
  myproc()->uid = uid;
```

```
  return 0;
}

// Set Process GID
int sys_setgid(void)
{
  uint gid;
  if(argint(0, (int*)&gid) < 0)
    return -1;
  if(gid < 0 || gid > 32767)
    return -1;
  myproc()->gid = gid;
  return 0;
}

// Get process information
int sys_getprocs(void)
{
  uint max;
  struct uproc* table;
  if(argint(0, (void*)&max) < 0)
    return -1;
  if(argptr(1, (void*)&table, sizeof(&table) * max) < 0)
    return -1;
  return getprocs(max, table);
}
#endif // CS333_P2
```

user.h

Line 3-5 :

```
#ifdef CS333_p2
struct uproc;
#endif // CS333_P2
```

Line 34-41 :

```
#ifdef CS333_P2
uint getuid(void);
uint getgid(void);
uint getppid(void);
int setuid(uint);
int setgid(uint);
int getprocs(uint max, struct uproc* table);
#endif // CS333_P2
```

usys.S

Line 34-39 :

SYSCALL(getuid)

SYSCALL(getgid)

SYSCALL(getppid)

SYSCALL(setuid)

SYSCALL(setgid)

SYSCALL(getprocs)

ps.c

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

int
main(void)
{
  struct uproc* table;
  int i;
  uint max = 72;
  int catch = 0;
  uint elapsed, decimal, seconds, seconds_decimal;
  table = malloc(sizeof(struct uproc) * max);
  catch = getprocs(max, table);
  if(catch == -1)
    printf(1, "\nError: Invalid max or NULL uproc table\n");
  else {
    printf(1, "\nPID\tName\tUID\tGID\tPPID\tElapsed\tCPU\tState\tSize");
    for(i = 0;i < catch;++i) {
      decimal = table[i].elapsed_ticks % 1000;
      elapsed = table[i].elapsed_ticks / 1000;
      seconds_decimal = table[i].CPU_total_ticks % 1000;
      seconds = table[i].CPU_total_ticks / 1000;
      printf(1, "\n%d\t%s\t%d\t%d\t%d\t%d.", table[i].pid, table[i].name, tabl
e[i].uid, table[i].gid, table[i].ppid, elapsed);
      if(decimal < 10)
        printf(1, "00");
      else if(decimal < 100)
        printf(1, "0");
      printf(1, "%d\t%d.", decimal, seconds);
      if(seconds_decimal < 10)
        printf(1, "00");
      else if(seconds_decimal < 100)
        printf(1, "0");
      printf(1, "%d\t%s\t%d", seconds_decimal, table[i].state, table[i].size);
```

```
    }
    printf(1, "\n");
  }
  free(table);
  exit();
}
#endif // CS333_P2
```

time.c

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

int
main(int argc, char* argv[])
{
  int t1 = 0, t2 = 0, elapsed = 0, decimal = 0, pid = 0;
  if(argc < 2)
    printf(1, "(null) ran in 0.000 seconds\n");
  else {
    ++argv;
    t1 = uptime();
    pid = fork();
    if(pid < 0) {
      printf(1, "Ran in 0.000 seconds\n");
      exit();
    }
    else if(pid == 0) {
      exec(argv[0], argv);
      printf(1, "Error: No such command\n");
    }
    else {
      wait();
      t2 = uptime();
      decimal = (t2 - t1) % 1000;
      elapsed = (t2 - t1) / 1000;
      printf(1, "%s ran in %d.", argv[0], elapsed);
      if(decimal < 10)
        printf(1, "00");
      else if(decimal < 100)
        printf(1, "0");
      printf(1, "%d seconds\n", decimal);
    }
  }
  exit();
}
#endif // CS333_P2
```