Code Modification Report

Nama : Putu Sanisa Pascaline

NIM : 1313619023

Prodi : Ilmu Komputer 2019

- **Defs.h**

Line 1 - 3:

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

Line 130 - 132:

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

- **Proc.c**

Line 9-11:

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

Line 158-161:

```
#ifdef CS333_P2 //project2
  p->cpu_ticks_total = 0;
  p->cpu_ticks_in = 0;
#endif // CS333_P2
```

Line 189-192

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

Line 257-260:

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

Line 419 – 421:

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

Line 462 – 464:

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

Line 592 – 636:

```
uint elapsed_s;
  uint elapsed_ms;

  elapsed_ms = ticks - p->start_ticks;
  elapsed_s = elapsed_ms / 1000;
  elapsed_ms = elapsed_ms % 1000;

  uint elapsed_cpu_s;
  uint elapsed_cpu_ms;
  uint ppid;
  if(p->parent){
    ppid = p->parent->pid;
  }
  else{
    ppid = p->pid;
  }

  elapsed_cpu_ms = p->cpu_ticks_total;
  elapsed_cpu_s = elapsed_cpu_ms / 1000;
  elapsed_cpu_ms = elapsed_cpu_ms % 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";

  cprintf(
    "\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"
,
    p->pid,
    p->name, "     ",
    p->uid, "        ",
    p->gid, "",
    ppid,
    elapsed_s, zero, elapsed_ms,
    elapsed_cpu_s, cpu_zero, elapsed_cpu_ms,
    state_string,
    p->sz
  );
```

Line 1008 – 1041:

```
#ifdef CS333_P2
int
getprocs(uint max, struct uproc* upTable){
  struct proc* p;
  int procsNumber = 0;
  acquire(&ptable.lock);

  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    if (procsNumber < max) {
      if(p->state != UNUSED && p->state != EMBRYO) {
        if(p->state >= 0 && p-
>state < NELEM(states) && states[p->state]){
          safestrcpy(upTable[procsNumber].state, states[p-
>state],STRMAX);
        } else {
          safestrcpy(upTable[procsNumber].state,"???",STRMAX
);
        }

        upTable[procsNumber].pid = p->pid;
        upTable[procsNumber].uid = p->uid;
        upTable[procsNumber].gid = p->gid;
        upTable[procsNumber].ppid = p->parent ? p->parent-
>pid : p->pid;
        upTable[procsNumber].elapsed_ticks = ticks - p-
>start_ticks;
        upTable[procsNumber].CPU_total_ticks = p-
>cpu_ticks_total;
        upTable[procsNumber].size = p->sz;
        safestrcpy(upTable[procsNumber].name, p-
>name, STRMAX);
        procsNumber++;
      }
    } else {
      break;
    }
  }
  release(&ptable.lock);
  return procsNumber;
}
#endif // CS333_P2
```

- **Proc.h**

Line 54-59:

```
#ifdef CS333_P2 //project2
  uint uid;
  uint gid;
  uint cpu_ticks_total;
  uint cpu_ticks_in;
#endif // CS333_P2
```

- **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
```

- **Syscall.c**

Line 114 - 121:

```
#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 152 - 159:

```
#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
```

Line 192 - 199:

```
#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 114 – 164:

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

```c
}
int
sys_getgid(void)
{
  return myproc()->gid;
}
int
sys_getppid(void)
{
  if(myproc()->pid == 1)
    return myproc()->pid;
  return myproc()->parent->pid;
}
int
sys_setuid(void)
{
  int tmp;
  if(argint(0,&tmp) < 0 || tmp > 32767 || tmp < 0)
    return -1;
  myproc()->uid = (uint)tmp;
  return 0;
}
int
sys_setgid(void)
{
  int tmp;
  if(argint(0,&tmp) < 0 || tmp > 32767 || tmp < 0)
    return -1;
  myproc()->gid = (uint)tmp;
  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
```

- **Time.c**

Line 1 – 46:

```
#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
```

- **User.h**

Line 33 - 40:

```
#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(getprocs)
```