Perbedaan terdapat pada beberapa file, yaitu:

- **proc.h**
- **user.h**
- **usys.S**
- **syscall.h**
- **syscall.c**
- **sysproc.c**
- **defs.h**
- **proc.c**
- **sh.c**
- **Makefile**

Penambahan beberapa file, yaitu:

- **testsetuid.c**
- **ps.c**
- **time.c**

Proc.h

Pada bagian struct proc ditambahkan:

```
// CS333_P2
  // Section uid gid
uint uid;
uint gid;

  // Section execution time
uint cpu_ticks_total;         // Total elapsed ticks in CPU
uint cpu_ticks_in;            // Ticks when scheduled

// Akhir CS333_P2
```

User.h

Ditambahkan:

```
// CS333_P2
#ifdef CS333_P2
    // Section uid gid
uint getuid(void); // UID of the current process
uint getgid(void); // GID of the current process
uint getppid(void); // parent PID of the current process

int setuid(uint); // Set UID
int setgid(uint); // Set GID

    // Section "ps" command
int getprocs(uint max, struct uproc*);

#endif// Akhir CS333_P2
```

usys.S

Ditambahkan:

```
SYSCALL(getuid)
SYSCALL(getgid)
SYSCALL(getppid)
SYSCALL(setuid)
SYSCALL(setgid)
SYSCALL(getprocs)
```

syscall.h

Ditambahkan:

```
// CS333_P2
#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
// Akhir CS333_P2
```

syscall.c

```
#ifdef CS333_P2
extern int sys_getuid(void); // Uintnya diganti jadi int
extern int sys_getgid(void); // Uintnya diganti jadi int
extern int sys_getppid(void); // Uintnya diganti jadi int

extern int sys_setuid(void);
extern int sys_setgid(void);

extern int sys_getprocs(void);

#endif // CS333_P2
```

```
#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
```

sysproc.c

```
#ifdef CS333_P2
int sys_getuid(void){ return myproc()->uid; }
int sys_getppid(void){
  if(myproc()->parent){
    return myproc()->parent->pid;
  }
  return myproc()->pid;
}
int sys_getgid(void){ return myproc()->gid; }

int sys_setuid(void){
  int uid;
  if(argint(0, &uid));
  if((0 <= uid) && (uid <= 32767)){
    myproc()->uid = uid;
    return 0;
  }
  return -1;
}
int sys_setgid(void){
  int gid;
  if(argint(0, &gid));
  if((0 <= gid) && (gid <= 32767)){
    myproc()->gid = gid;
    return 0;
  }
  return -1;
}
```

```c
#include "uproc.h"
int sys_getprocs(void){
  // Only have a state like: RUNNABLE, SLEEPING, RUNNING, or ZOMBIE.
  int max;
  struct uproc* table;

  if(argint(0, &max) < 0)
    return -1;
  if(argptr(1, (void*)&table, sizeof(struct uproc) * max) < 0)
    return -1;

  return getprocs(max, table);
}

#endif // CS333_P2
```

defs.h

```c
#ifdef CS333_P2
#include "uproc.h"
int             getprocs(uint max, struct uproc*);
#endif // Akhir CS333_P2
```

proc.c

Ditambahkan fungsi:

```
// Fungsi getprocs ini adalah untuk membantu akses sys_getprocs
#ifdef CS333_P2
#include "uproc.h"
int
getprocs(uint max, struct uproc* table)
{
  struct proc* p;
  int countPtr = 0;

  acquire(&ptable.lock);

  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
    if(countPtr >= max) break ;
    if(p->state != UNUSED && p->state != EMBRYO)
    {
      table[countPtr].pid = p->pid;
      safestrcpy(table[countPtr].name, p->name, STRMAX);
      table[countPtr].uid = p->uid;
      table[countPtr].gid = p->gid;
      table[countPtr].ppid = p->parent ? p->parent->pid : p->pid;

      table[countPtr].elapsed_ticks = ticks - (p->start_ticks);
      table[countPtr].CPU_total_ticks = p->cpu_ticks_total;
      safestrcpy(table[countPtr].state, states[p->state], STRMAX);
      table[countPtr].size = p->sz;
      countPtr++;
    }
  }
  release(&ptable.lock);
  if(countPtr)  return countPtr;
```

```
  release(&ptable.lock);
  if(countPtr)  return countPtr;
  return -1;
}
#endif // CS333_P2
```

```
#if defined(CS333_P2)
void
helper(int value, int value_dec){
    if(value_dec < 10)
      cprintf("%d.00%d\t", value, value_dec);
    else if(value_dec < 100 && value_dec >= 10)
      cprintf("%d.0%d\t", value, value_dec);
    else
      cprintf("%d.%d\t", value, value_dec);
}

void
procdumpP2P3P4(struct proc *p, char *state_string)
{
  cprintf("%d\t%s\t     %d\t\t%d\t%d\t", p->pid, p->name, p->uid, p->gid, p->parent ? p->parent->pid : p->pid);
  int elapsed = ticks-(p->start_ticks);
  int cpu = p->cpu_ticks_total;
  int decimal = elapsed%1000; helper(elapsed/1000, decimal);
  decimal = cpu%1000;         helper(cpu/1000, decimal);
  cprintf("%s\t%d", state_string, p->sz);
  return;
}
```

```
void
sched(void)
{
  int intena;
  struct proc *p = myproc();

  if(!holding(&ptable.lock))
    panic("sched ptable.lock");
  if(mycpu()->ncli != 1)
    panic("sched locks");
  if(p->state == RUNNING)
    panic("sched running");
  if(readeflags()&FL_IF)
    panic("sched interruptible");
#ifdef CS333_P2
  p->cpu_ticks_total += ticks-(p->cpu_ticks_in);
#endif
  intena = mycpu()->intena;
  swtch(&p->context, mycpu()->scheduler);
  mycpu()->intena = intena;
}
```

Pada fungsi scheduler ditambahkan:

```
399  #ifdef PDX_XV6
400          idle = 0;  // not idle this timeslice
401  #endif // PDX_XV6
402          c->proc = p;
403          switchuvm(p);
404  #ifdef CS333_P2
405          c->proc->cpu_ticks_in = ticks;
406  #endif
407          p->state = RUNNING;
408          swtch(&(c->scheduler), p->context);
409          switchkvm();
```

Pada fungsi fork ditambahkan:

```
257    pid = np->pid;
258
259    acquire(&ptable.lock);
260    np->state = RUNNABLE;
261    release(&ptable.lock);
262
263    // Copying uid & gid from current process to new process
264    np->uid = curproc->uid;
265    np->gid = curproc->gid;
266    // End of code CS333_P2 ==================================
267
268    return pid;
269  }
```

Pada fungsi allocproc ditambahkan:

```
153    p->start_ticks = ticks;
154
155    // CS333_P2
156    p->cpu_ticks_total = 0;
157    p->cpu_ticks_in = 0;
158    return p;
159  }
```

sh.c

```
int
getbuiltin(char *p)
{
  p += strlen("_get");
  while (strncmp(p, " ", 1) == 0) p++; // chomp spaces
  if (strncmp("uid", p, 3) == 0) {
    printf(2, "%d\n", getuid());
    return 0;
  }
  if (strncmp("gid", p, 3) == 0) {
    printf(2, "%d\n", getgid());
    return 0;
  }
  if (strncmp("ppid", p, 3) == 0) {
    printf(2, "%d\n", getppid());
    return 0;
  }
  printf(2, "Invalid _get parameter\n");
  return -1;
}
```

Makefile

Mengubah “CS333_PROJECT ?=” menjadi bernilai 2.

testsetuid.c

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

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

ps.c

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

void
helper(uint value, uint value_dec){
    if(value_dec < 10)
      printf(1, "%d.00%d\t", value, value_dec);
    else if(value_dec < 100 && value_dec >= 10)
      printf(1, "%d.0%d\t", value, value_dec);
    else
      printf(1, "%d.%d\t", value, value_dec);
}

int
main(int argc, char *argv[])
{
    int max = 72;

    if(argc > 1)
        max = atoi(argv[1]);

    struct uproc* table = malloc(sizeof(*table)*max);

    if(table == 0)
    {
        printf(1, "Unable to initialize table in ps.c\n");
        exit();
    }
```

```
32      int process = getprocs(max, table);
33      int time, cpu_time, decimal;
34
35      if(process < 1)
36      {
37          free(table);
38          printf(1, "Error\n");
39      }
40      else printf(1, "PID\tName\tUID\tGID\tPPID\tElapsed\tCPU\tState\tSize\n");
41
42      for(int i = 0; i < process; i++)
43      {
44          printf(1, "%d\t%s\t%d\t%d\t%d\t",
45              table[i].pid,
46              table[i].name,
47              table[i].uid,
48              table[i].gid,
49              table[i].ppid);
50          time = table[i].elapsed_ticks;
51          decimal = time % 1000;
52          time /= 1000;
53          helper(time, decimal);
54          cpu_time = table[i].CPU_total_ticks;
55          decimal = cpu_time % 1000;
56          cpu_time /= 1000;
57          helper(cpu_time, decimal);
58          printf(1, "%s\t", table[i].state);
59          printf(1, "%d\n", table[i].size);
60      }
61      free(table);
```

```
62      exit();
63  }
64  #endif
```

time.c

```c
#include "types.h"
#include "user.h"
int main(int argc, char * argv[])
{
    int time_start = uptime();
    int pid = fork();
    if(pid < 0) exit();
    if(pid == 0)
    {
        exec(argv[1], argv+1);
        exit();
    }
    if(pid > 0)
    {
        wait();
        int time = uptime();
        time = time - time_start;
        int decimal = time % 1000;
        time /= 1000;
        if(decimal < 10)
        printf(1, "%s ran in %d.00%d seconds\n",argv[1], time, decimal);
        else if(decimal < 100 && decimal >= 10)
        printf(1, "%s ran in %d.0%d seconds\n", argv[1], time, decimal);
        else
        printf(1, "%s ran in %d.%d seconds\n", argv[1], time, decimal);
        exit();
    }
    exit();
    return 0;
}
```