

- **Makefile**

Mengganti dari angka 1 ke 2

Baris ke 3

```
CS333_PROJECT ?= 2
```

Penambahan

- **defs.h**

Baris 12-14

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

Baris 130-132

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

- **Proc.c**

Baris 9-11

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

Baris 157-160

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

Baris 197-200

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

```
#endif // CS333_P2
```

Baris 266-269

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

Baris 409-411

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

Baris 452-454

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

Baris 581-593

```
// cprintf("TODO for Project 2, delete this line and implement procdumpP2P3P4() i
n proc.c to print a row\n");
int elapsed = ticks - p->start_ticks;
int total = p->cpu_ticks_total;
int ppid;
if(p->parent)
{
    ppid = p->parent->pid;
}
else
{
    ppid = p->pid;
}
cprintf("%d\t%s\t\t%d\t\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/1000, elapsed%1000, total/1000, total%1000, state_string, p-
>sz);
return;
```

Baris 954-984

```
#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**

Baris 57 – 62

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

- **Syscall.c**

Baris 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
```

Baris 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
```

Baris 160

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

Baris 187 -194

```
#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**

Baris 25-30

```
#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**

Baris 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**

Baris 3-5

```

#ifdef CS333_P2
struct uproc;
#endif // CS333_P2

```

Baris 35-42

```

#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**

Baris 34-39

SYSCALL(getuid)

SYSCALL(getgid)

SYSCALL(getppid)

SYSCALL(setuid)

SYSCALL(setgid)

SYSCALL(getprocs)

Membuat file baru

- **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, table[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
```