NAMA : LISA ARIEF
PRODI : ILMU KOMPUTER 2018
Assignment 1 – Sistem Operasi

Pertama, melakukan perubahan pada file Makefile

Makefile

- Line 3

```
CS333_PROJECT ?= 2
```

Kemudian, menambah potongan kode pada beberapa file berikut:

user.h

- Line 33 – 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

- Line 36 – 41

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

syscall.h

```
#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
```

syscall.c

- Line 115 – 122

```
#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
```

syscalls[]

- Line 154 – 161

```
#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
```

syscallnames[]

- Line 195 – 202

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

sysproc.c

- Line 13 – 15

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

- Line 121 – 183

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

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

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

int sys_setuid(void)
{
  int uid;

  if(argint(0, (int*)&uid) < 0){
    return -1;
  }
  if(uid < 0 || uid > 32767){
    return -1;
  }
```

```
  myproc()->uid = uid;
  return 0;
}

int sys_setgid(void)
{
  int gid;

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

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

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

proc.h

proc{}

- Line 57 – 63

```
#ifdef CS333_P2
uint uid;
uint gid;
```

```
unit cpu_ticks_total;
uint cpu_ticks_in;
#endif // CS333_P2
```

proc.c

- Line 10 – 13

```
#ifdef CS333_P2
#include “uproc.h”
#include “pdx.h”
#endif // CS333_P2
```

allocproc(void)

- Line 161 – 165

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

userinit(void)

- Line 204 – 207

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

fork(void)

- Line 259 – 262

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

scheduler(void)

- Line 418 – 420

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

sched (void)

- Line 463 – 465

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

procdumpP2P3P4()

- Line 594 – 613

```
#ifdef defined(CS333_P2)
int ppid;

int elapsed = ticks - p->start_ticks;
int total = p->cpu_ticks_total;

int second = elapsed/1000;
int millisecond = elapsed%1000;

int total_cpus = total/1000;
int total_cpums = total%1000;

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

cprintf("%d\t%s\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, second, millisecond, total_cpus, total_cpums,
state_string, p->sz);
return;
```

getprocs()

- Line 948 – 988

```
#ifdef CS333_P2
int getprocs(uint max, struct uproc* table)
{
  int i = 0;

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

defs.h

- Line 13 – 15

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

- Line 13 – 15

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

Selanjutnya, membuat file baru sebagai berikut:

testsetuid.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
```

ps.c

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

int main(int argc, char * argv[])
{
  int max = 72; // 1, 16, 64, 72
  struct uproc * table = malloc(sizeof(*table)*max);
  int num_procs = getprocs(max, table);
  uint second;
  uint millisecond;
```

```
  uint second_cpu;
  uint millisecond_cpu;

  if(table == 0){
    printf(1, "Invalid, unable to make table \n");
    exit();
  }

  printf(1, "MAX = %d", max);
  printf(1, "\nPID\tName     UID\tGID\tPPID\tElapsed\tCPU\tState\tSize\n");

  for(int i = 0; i < num_procs; i++){
    printf(1, "%d\t%s\t     %d\t\t%d\t%d\t", table[i].pid, table[i].name,
table[i].uid, table[i].gid, table[i].ppid);

    // Elapsed time
    second  = table[i].elapsed_ticks/1000;
    millisecond = table[i].elapsed_ticks%1000;

    if(millisecond >= 100){
      printf(1, "%d.%d\t", second, millisecond);
    }
    else if(millisecond < 10){
      printf(1, "%d.00%d\t", second, millisecond);
    }
    else{
      printf(1, "%d.0%d\t", second, millisecond);
    }

    // CPU time
    second_cpu  = table[i].CPU_total_ticks/1000;
    millisecond_cpu = table[i].CPU_total_ticks%1000;

    if(millisecond_cpu >= 100){
      printf(1, "%d.%d\t", second_cpu, millisecond_cpu);
    }
    else if(millisecond_cpu < 10){
      printf(1, "%d.00%d\t", second_cpu, millisecond_cpu);
    }
```

```
    else{
      printf(1, "%d.0%d\t", second_cpu, millisecond_cpu);
    }
    // State & Size
    printf(1, "%s\t%d\n", table[i].state, table[i].size);
  }

  free(table);
  exit();
}
#endif // CS333_P2
```

time.c

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

int main(int argc, char * argv[])
{
  if(argc == 1){
    printf(1, "(null) ran in 0.000 seconds.\n");
    exit();
  }

  int start_time = uptime();
  int pid = fork();

  if(pid < 0){
    exit();
  }

  else if(pid == 0){
    exec(argv[1], argv+1);
    exit();
  }

  else if(pid > 0){
    wait();
```

```
    int end_time = uptime();
    int total = end_time - start_time;

    int second  = total/1000;
    int millisecond = total%1000;

    if(millisecond >= 100){
      printf(1, "%s run in %d.%d seconds.\n", argv[1], second,
millisecond);
    }
    else if(millisecond < 10){
      printf(1, "%s run in %d.00%d seconds.\n", argv[1], second,
millisecond);
    }
    else{
      printf(1, "%s run in %d.0%d seconds.\n", argv[1], second,
millisecond);
    }
  }
  else
    exit();

  exit();
}
#endif // CS333_P2
```