DIYAH ANGGRAENY
1313618005
ILMU KOMPUTER 2018

Nama file dan function yang diubah:

Proc.h

- Struct-proc

```
#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
```

User.h

```
#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
```

Makefile

```
CS333_PROJECT ?= 2
```

Proc.c

- Allocproc(void)

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

- Fork(void)

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

- Scheduler(void)

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

- Sched(void)

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

- Getprocs(uint max, struct uproc * table)

```
int
getprocs(uint max, struct uproc * table)
{
  struct proc *p;
  int ppid = 0;
  int i = 0;


  //Loop over process table looking for process
  acquire(&ptable.lock);
  for(p = ptable.proc; (p < &ptable.proc[NPROC]&& i < max); p++)
  {

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

    if(p->state == SLEEPING|| p->state == RUNNING || p->state ==
RUNNABLE ||p->state == ZOMBIE)
    {
        table[i].pid = p->pid;
        table[i].uid = p->uid;
        table[i].gid = p->gid;
        table[i].ppid = ppid;
        table[i].elapsed_ticks = ticks - p->start_ticks;
        table[i].CPU_total_ticks = p->cpu_ticks_total;
        safestrcpy(table[i].state, states[p->state], STRMAX);
        table[i].size = p->sz;
        safestrcpy(table[i].name, p->name, STRMAX);

        i++;
    }
  }

    release(&ptable.lock);

return i;
}
```

- procdumpP2P3P4(struct proc *p, char *state_string)

```
void
procdumpP2P3P4(struct proc *p, char *state_string)
{
  #ifdef CS333_P2

  //Decimal conversion for Elapsed time
    int elapsed = 0;
    elapsed = ticks - p->start_ticks;
    int secs = (elapsed)/1000;
    int ms = (elapsed)%1000;
```

```
    int hundreds = (ms/100);
    int tens = ((ms%100)/10);
    int ones = (ms%10);

  //Decimal conversion for CPU time
    int cpu_s = p->cpu_ticks_total/1000;
    int cpu_ms = p->cpu_ticks_total%1000;

    int cpu_hundreds = cpu_ms/100;
    int cpu_tens = (cpu_ms%100)/10;
    int cpu_ones = cpu_ms%10;

    //Checking if ppid == null or not
    int ppid = 0;
      if(p->parent == NULL)
        ppid = p->pid;
      else
        ppid = p->parent->pid;

   //print statement to kernel

  cprintf("\n%d\t%s\t\t%d\t%d\t%d\t%d.%d%d%d\t%d.%d%d%d\t%s\t%d\t",p-
>pid, p->name, p->uid, p->gid, ppid, secs, hundreds, tens, ones,
cpu_s, cpu_hundreds, cpu_tens, cpu_ones, state_string, p->sz);

  return;
}
#endif //CS333_P2
```

Sysproc.c

- sys_getuid(void)

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

- sys_getgid(void)

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

- sys_getppid(void)

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

- sys_setuid(void)

```
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;
}
```

- sys_setgid(void)

```
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;
}
```

- sys_getprocs(void)

```
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);
}
```

Time.c

- main(int argc, char* argv[])

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

int
main(int argc, char* argv[])
{
  int t1 = 0, t2 = 0, elapsed = 0, dec = 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();
      dec = (t2 - t1) % 1000;
      elapsed = (t2 - t1) / 1000;
      printf(1, "%s ran in %d.", argv[0], elapsed);
      if(dec < 10)
        printf(1, "00");
      else if(dec < 100)
        printf(1, "0");
      printf(1, "%d seconds\n", dec);
    }
  }
  exit();
}
#endif // CS333_P2
```

Ps.c

- main(void)

```
#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, decs, secs, secs_decs;
  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,"MAX = 72");
    printf(1,
"\nPID\tName\tUID\tGID\tPPID\tElapsed\tCPU\tState\tSize");
    for(i = 0;i < catch;++i) {
      decs = table[i].elapsed_ticks % 1000;
      elapsed = table[i].elapsed_ticks / 1000;
      secs_decs = table[i].CPU_total_ticks % 1000;
      secs = 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(decs < 10)
        printf(1, "00");
      else if(decs < 100)
        printf(1, "0");
      printf(1, "%d\t%d.", decs, secs);
```

```
      if(secs_decs < 10)
        printf(1, "00");
      else if(secs_decs < 100)
        printf(1, "0");
      printf(1, "%d\t%s\t%d", secs_decs, table[i].state,
table[i].size);
    }
    printf(1, "\n");
  }
  free(table);
  exit();
}
#endif // CS333_P2
```

Syscall.c

```
#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[]

```
#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[]

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

Usys.s

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

Defs.h

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

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
```