## Task 1 → Syscall Tracing

Syscall.c

```
void
syscall(void)
{
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
    curproc->tf->eax = syscalls[num]();
    #ifdef PRINT_SYSCALLS
      cprintf("%s -> %d \n",syscallnames[num], num);
    #endif
  } else {
    cprintf("%d %s: unknown sys call %d\n",
            curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```

## 4. Date System Call

makefile

```
CS333_PROJECT ?= 0
PRINT_SYSCALLS ?= 1
CS333_CFLAGS ?= -DPDX_XV6
ifeq ($(CS333_CFLAGS), -DPDX_XV6)
CS333_UPROGS += _halt _uptime
endif

ifeq ($(PRINT_SYSCALLS), 1)
CS333_CFLAGS += -DPRINT_SYSCALLS
endif

ifeq ($(CS333_PROJECT), 1)
CS333_CFLAGS += -DCS333_P1
CS333_UPROGS += _date
endif
```

user.h

```
// system calls
#ifdef CS333_P1
int date(struct rtcdate*);
#endif // CS333_P1
```

usys.S

```

SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
SYSCALL(halt)
SYSCALL(date)
```

syscall.h

```
#define SYS_halt    SYS_close+1
#define SYS_date    SYS_halt+1
```

syscall.c

```
#ifdef CS333_P1
[SYS_date]    sys_date,
#endif // PDX_XV6
```

```
#ifdef CS333_P1
extern int sys_date(void);
#endif // CS333_P1
```

sysproc.c

```
int
sys_date ( void )
{
  struct rtcdate *d ;
  if (argptr ( 0 ,( void*)&d , sizeof ( struct rtcdate)) < 0)
    return -1;
  cmostime(d);
  return 0;
}
```

5. Process Information

proc.h

```
uint start_ticks;
```

proc.c

```
 p->start_ticks = ticks;
  return p;
```

```
void
procdumpP1(struct proc *p, char *state_string)
{
  int elapsed_s;
  int elapsed_ms;

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

  char* nol = "";
  if(elapsed_ms < 100 && elapsed_ms >= 10)
    nol = "0";
  if(elapsed_ms < 10)
  nol = "00";

  cprintf("%d\t%s\t%s%d.%s%d\t%s\t%d\t",
  p->pid, p->name, "      ",elapsed_s, nol, elapsed_ms, states[p->state], p-
>sz);
  return;
}
```