과제 #2 : system calls in xv6

○ 과제 목표

- xv6에서 새로운 시스템 호출 추가

○ 기본 지식

- 시스템 호출 추가 방법 이해

✓ 기존 시스템 호출의 구현을 따라 새 시스템 호출을 추가하는 방법을 이해

- (1) 인자가 없는 시스템 호출 : 특정 시스템 호출은 인자가 없고 정수 값만 리턴
  - ☞ 예 : sysproc.c 내 구현된 uptime()
- (2) 인자가 있는 시스템 호출 : 특정 시스템 호출은 문자열 및 정수 등 여러 인수를 받아 간단한 정수 값을 리턴
  - ☞ 예 : sysfile.c 내 구현된 open()
- (3) 구조체 인자/리턴 시스템 호출 : 특정 시스템 호출은 여러 정보를 사용자가 정의한 구조체로 사용자 프로그램에 리턴
  - ☞ 예. fstat()은 파일에 대한 정보를 struct stat를 넣고 이 구조체를 가져와서 ls 응용 프로그램에 의해 파일에 대한 정보를 표준 출력

- Cross Compile 방법 학습

✓ xv6에는 텍스트 편집기 또는 gcc 컴파일러가 없음. 따라서 자신의 리눅스 시스템에서 vi를 이용하여 프로그램 작성하고 컴파일하고 나온 실행파일을 xv6 상에서 수행

- xv6 커널 이해

✓ proc.c, proc.h, syscall.c, syscall.h, sysproc.c, user.h, usys.S 수정 필요

- user.h : xv6의 시스템 호출 정의
- usys.S : xv6의 시스템 호출 리스트
- syscall.h : 시스템 호출 번호 매핑. -> 새 시스템 호출을 위해 새로운 매핑 추가
- syscall.c : 시스템 호출 인수를 구문 분석하는 함수 및 실제 시스템 호출 구현에 대한 포인터
- sysproc.c : 프로세스 관련 시스템 호출 구현. -> 여기에 시스템 호출 코드를 추가
- proc.h는 struct proc 구조 정의 -> 프로세스에 대한 추가 정보를 추적을 위해 구조 변경
- proc.c : 프로세스 간의 스케줄링 및 컨텍스트 전환을 수행하는 함수

○ 과제 내용

1. date() 시스템 콜 추가 및 이를 호출하는 간단한 쉘 프로그램 구현

✓ 날짜 및 시간 정보를 출력하는 date() 시스템 호출 구현

- date.h에 struct rtcdate 사용
- 커널 모드의 cmostime() 사용

(예시 1). user.h에 date 추가

```
// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, const void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(const char*, int);
int mknod(const char*, short, short);
int unlink(const char*);
int fstat(int fd, struct stat*);
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
```

(예시 2). usys.S에 date 추가

```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name) \
  .globl name; \
  name: \
    movl $SYS_ ## name, %eax; \
    int $T_SYSCALL; \
    ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
```

(예시 3). syscall.h에 date 시스템 콜 번호 부여

```
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
```

(예시 4). syscall.c에 date 추가

```
extern int sys_chdir(void);
extern int sys_close(void);
extern int sys_dup(void);
extern int sys_exec(void);
extern int sys_exit(void);
extern int sys_fork(void);
extern int sys_fstat(void);
extern int sys_getpid(void);
extern int sys_kill(void);
extern int sys_link(void);
extern int sys_mkdir(void);
extern int sys_mknod(void);
extern int sys_open(void);
extern int sys_pipe(void);
extern int sys_read(void);
extern int sys_sbrk(void);
extern int sys_sleep(void);
extern int sys_unlink(void);
extern int sys_wait(void);
extern int sys_write(void);
extern int sys_uptime(void);

static int (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
```

(예시 5). date 시스템 콜 구현

```
int
sys_date(void)
{
  struct rtcdate *d;
  return 0;
}
```

(예시 6). datetest 쉘 프로그램 구현

```
int main(int argc, char *argv[])
{
    struct rtcdate r;

    exit();
}
```

✓ date() 실행 확인
- datetest 쉘 프로그램 구현
- 아래 예시와 동일한 결과가 나와야 함

(예시 7). datetest 실행 결과

```
$ datetest
Current time : 2023-9-11 7:58:57
```

2. alarm 시스템 콜 추가 및 이를 호출하는 간단한 쉘 프로그램

✓ 입력받은 시간(second)이 지날 시 "SSU_Alarm!" 과 종료되는 시간을 출력과 동시에 프로그램을 종료하는 alarm()
시스템 호출 구현
- timer 역할을 하는 [seconds]를 인자로 받음
- 타이머 인터럽트 사용을 위해 trap.c 내부의 trap(struc trapframe *tf) 수정
  ☞ 각 프로세스 마다 alarmticks 증가를 위해 타이머 인터럽트 내부 코드 추가

(예시 8). proc.h에서 alarmticks, alarm_timer 선언

```
struct proc {
          uint sz;                              // Size of process memory (bytes)
          pde_t* pgdir;                 // Page table
          char *kstack;                 // Bottom of kernel stack for this process
          enum procstate state;         // Process state
          int pid;                      // Process ID
          struct proc *parent;          // Parent process
          struct trapframe *tf;         // Trap frame for current syscall
          struct context *context;      // swtch() here to run process
          void *chan;                   // If non-zero, sleeping on chan
          int killed;                   // If non-zero, have been killed
          struct file *ofile[NOFILE];   // Open files
          struct inode *cwd;            // Current directory
          char name[16];                // Process name (debugging)
};
```

(예시 9). sysproc.c에 sys_alarm 구현

```
int
sys_alarm(void)
{

}
```

(예시 10). 타이머 인터럽트 사용을 위한 trap() 수정

```
void
trap(struct trapframe *tf)
{
  if(tf->trapno == T_SYSCALL){
    if(myproc()->killed)
      exit();
    myproc()->tf = tf;
    syscall();
    if(myproc()->killed)
      exit();
    return;
  }

  switch(tf->trapno){
  case T_IRQ0 + IRQ_TIMER:
    if(cpuid() == 0){
      acquire(&tickslock);
      ticks++;
      wakeup(&ticks);
      release(&tickslock);
    }

  lapiceoi();
  break;
```

✓ alarm 시스템 콜 실행 확인

- alarm_test 쉘 프로그램 구현
- alarm_test [seconds]

☞ 입력받은 [seconds](초)가 지나면 alarm_test가 종료되는 프로그램

(예시 11). alarm_test 코드 (수정 불가)

```
#include "types.h"
#include "user.h"
#include "date.h"

int main(int argc, char *argv[])
{
    int seconds;
    struct rtcdate r;

    if(argc <= 1)
        exit();

    seconds = atoi(argv[1]);

    alarm(seconds);

    date(&r);
    printf(1,"SSU_Alarm Start\n");
    printf(1,"Current time : %d-%d-%d %d:%d:%d\n",r.year,r.month,r.day,r.hour,r.minute,r.second);

    while(1)
        ;
    exit();
}
```

(예시 12). alarm_test 실행 예시

```
$ alarm_test 3
SSU_Alarm Start
Current time : 2023-9-12 7:25:12
SSU_Alarm!
Current time : 2023-9-12 7:25:15
```

○ 과제 제출 마감

- 2023년 10월 15일 (일) 23시 59분 59초까지 구글클래스룸으로 제출
- 보고서 (hwp, doc, docx 등으로 작성 – 총 2개의 테스트 프로그램이 수행된 결과 (캡쳐 등 포함)
- xv6에서 변경한 소스코드 및 테스트 쉘 프로그램 2개(datetest, alarmtest)
- 마감시간 이후 24시간까지 지연 제출 가능. 그 이후 제출은 0점 처리. 설계과제 마감시간 이후 지연 제출은 30% 감점.

○ 필수 구현(설치 및 설명 등)

- 1

○ 배점 기준

- 1 : 10점
- 2 : 90점