HW4. XV6 System Call 구현하기

학번: 20170404

이름: 한종수

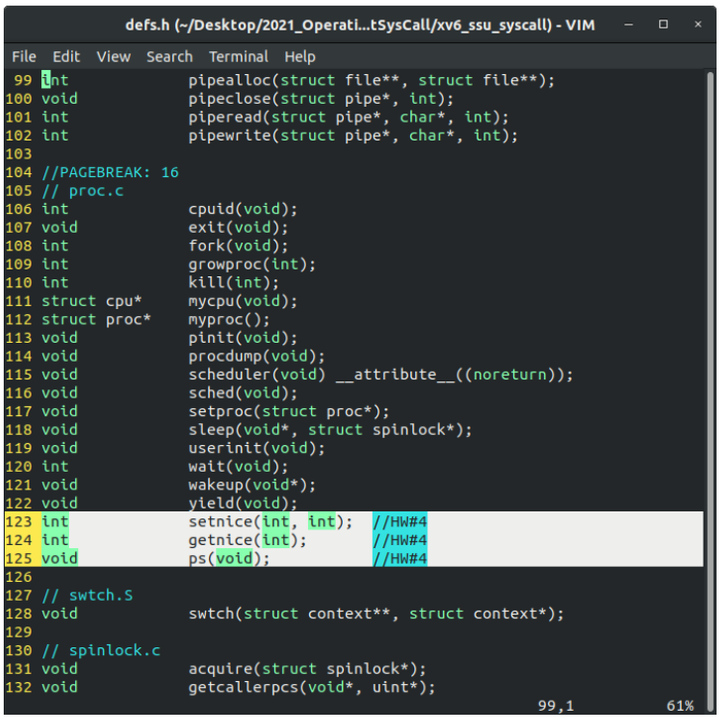
제출일: 2021. 4. 5

1.과제 개요

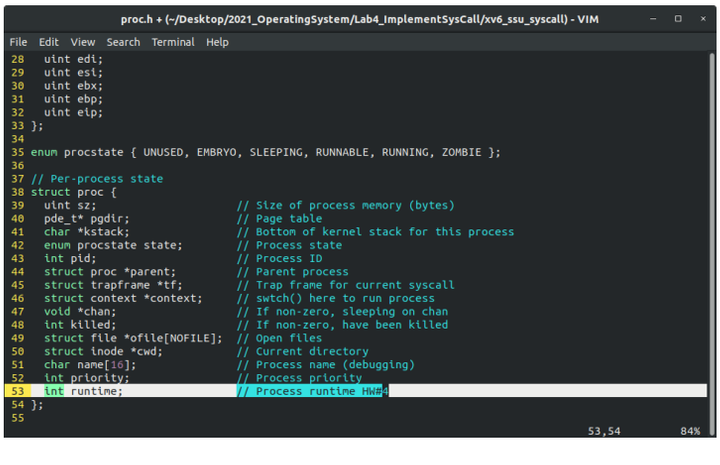
본 과제에서는 프로세스와 관련된 세 가지 시스템콜인 getnice, setnice, ps를 구현하고 test 코드를 통해 올바른 동작을 확인한다.

2.소스코드

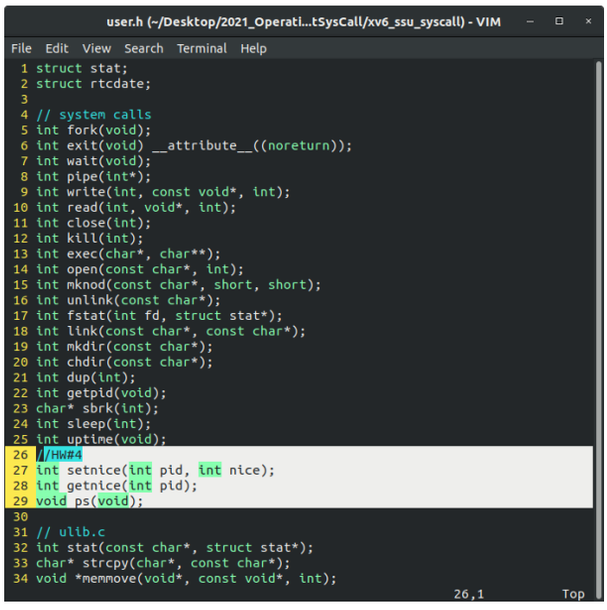
(수정한 코드를 캡처하고 간단히 설명함)



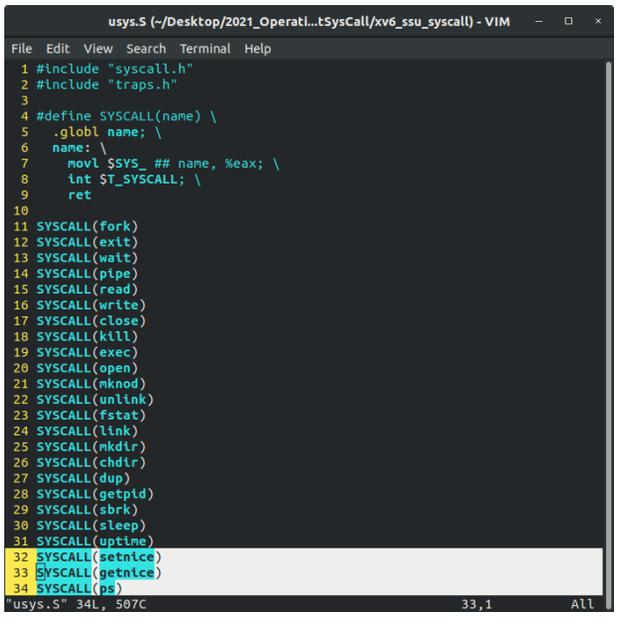
defs.h에서 sys call 등록



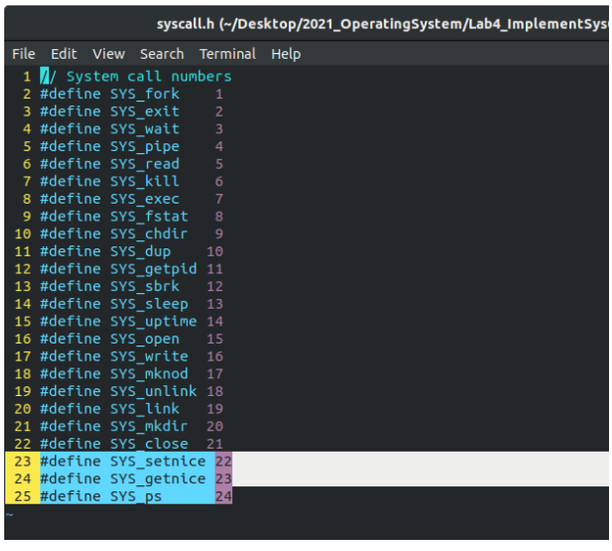
proc.h에서 proc struct에 runtime value 추가



user.h에 system call 지정



usys.S에 SYSCALL 호출 시 syscall 번호 넣도록 수정



syscall.h에서 syscall_number 지정

syscall.c (~/Desktop/2021_OperatingSystem/L

File Edit View Search Terminal Help

```
 85 extern int sys_chdir(void);
 86 extern int sys_close(void);
 87 extern int sys_dup(void);
 88 extern int sys_exec(void);
 89 extern int sys_exit(void);
 90 extern int sys_fork(void);
 91 extern int sys_fstat(void);
 92 extern int sys_getpid(void);
 93 extern int sys_kill(void);
 94 extern int sys_link(void);
 95 extern int sys_mkdir(void);
 96 extern int sys_mknod(void);
 97 extern int sys_open(void);
 98 extern int sys_pipe(void);
 99 extern int sys_read(void);
100 extern int sys_sbrk(void);
101 extern int sys_sleep(void);
102 extern int sys_unlink(void);
103 extern int sys_wait(void);
104 extern int sys_write(void);
105 extern int sys_uptime(void);
106 extern int sys_setnice(void);
107 extern int sys_getnice(void);
108 extern int sys_ps(void);
109
110 static int (*syscalls[])(void) = {
111 [SYS_fork]    sys_fork,
112 [SYS_exit]    sys_exit,
113 [SYS_wait]    sys_wait,
114 [SYS_pipe]    sys_pipe,
115 [SYS_read]    sys_read,
116 [SYS_kill]    sys_kill,
117 [SYS_exec]    sys_exec,
118 [SYS_fstat]   sys_fstat,
119 [SYS_chdir]   sys_chdir,
120 [SYS_dup]     sys_dup,
121 [SYS_getpid]  sys_getpid,
122 [SYS_sbrk]    sys_sbrk,
123 [SYS_sleep]   sys_sleep,
124 [SYS_uptime]  sys_uptime,
125 [SYS_open]    sys_open,
126 [SYS_write]   sys_write,
127 [SYS_mknod]   sys_mknod,
128 [SYS_unlink]  sys_unlink,
129 [SYS_link]    sys_link,
130 [SYS_mkdir]   sys_mkdir,
131 [SYS_close]   sys_close,
132 [SYS_setnice] sys_setnice,
133 [SYS_getnice] sys_getnice,
134 [SYS_ps]      sys_ps
135 };
136
```

syscall.c system call table에 핸들러 등록

sysproc.c + (~/Desktop/2021_OperatingSys..

File Edit View Search Terminal Help

```
 92
 93 //HW#4
 94 int
 95 sys_setnice(void)
 96 {
 97     int pid;
 98     int nice;
 99
100     if(argint(0, &pid) < 0)
101         return -1;
102
103     if(argint(1, &nice) < 0)
104         return -1;
105
106     return setnice(pid, nice);
107 }
108
109 int
110 sys_getnice(void)
111 {
112     int pid;
113
114     if(argint(0, &pid) < 0)
115         return -1;
116
117     return getnice(pid);
118 }
119
120 int
121 sys_ps(void)
122 {
123     ps();
124     return 0;
125 }
```

sysproc.c의 setnice, getnice, ps의 핸들러 인자가 제대로 들어왔는지 검사해서 인자들을 시스템 콜 핸들러를 실행.

proc.c

1) setnice : nice 값이 잘 들어왔는지 범위체크해서 범위 넘으면 -1 리턴. 아니면 ptable 돌면서 일치하는 pid 검색. 있으면 해당 pid의 nice를 인자로 들어온 nice값으로 수정. 찾는 pid가 없으면 -1 리턴

2) getnice : nice 값이 잘 들어왔는지 범위체크해서 범위 넘으면 -1 리턴. 아니면 ptable 돌면서 일치하는 pid 검색. 있으면 해당 pid의 nice를 리턴. 찾는 pid가 없으면 -1 리턴

3) ps : ptable을 돌며 현재 있는 process 들의 정보를 출력한다.

proc.c (~/Desktop/2021_Operating...entSysCall/xv6_ssu_syscall) - VIM

File Edit View Search Terminal Help

```
69 // Look in the process table for an UNUSED proc.
70 // If found, change state to EMBRYO and initialize
71 // state required to run in the kernel.
72 // Otherwise return 0.
73 static struct proc*
74 allocproc(void)
75 {
76   struct proc *p;
77   char *sp;
78
79   acquire(&ptable.lock);
80
81   for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
82     if(p->state == UNUSED)
83       goto found;
84
85   release(&ptable.lock);
86   return 0;
87
88 found:
89   p->state = EMBRYO;
90   p->runtime = 0;
91   p->pid = nextpid++;
92
93   release(&ptable.lock);
94
95   // Allocate kernel stack.
96   if((p->kstack = kalloc()) == 0){
97     p->state = UNUSED;
98     return 0;
99   }
100  sp = p->kstack + KSTACKSIZE;
101
102  // Leave room for trap frame.
103  sp -= sizeof *p->tf;
104  p->tf = (struct trapframe*)sp;
105
106  // Set up new context to start executing at forkret,
107  // which returns to trapret.
108  sp -= 4;
109  *(uint*)sp = (uint)trapret;
110
111  sp -= sizeof *p->context;
112  p->context = (struct context*)sp;
113  memset(p->context, 0, sizeof *p->context);
114  p->context->eip = (uint)forkret;
115
116  return p;
117 }
                                                    72,1          12%
```

proc.c (~/Desktop/2021_Operating...entSysCall/xv6_ssu_syscall) - VIM

File Edit View Search Terminal Help

```
178
179 // Create a new process copying p as the parent.
180 // Sets up stack to return as if from system call.
181 // Caller must set state of returned proc to RUNNABLE.
182 int
183 fork(void)
184 {
185   int i, pid;
186   struct proc *np;
187   struct proc *curproc = myproc();
188
189   // Allocate process.
190   if((np = allocproc()) == 0){
191     return -1;
192   }
193
194   // Copy process state from proc.
195   if((np->pgdir = copyuvm(curproc->pgdir, curproc->sz)) == 0){
196     kfree(np->kstack);
197     np->kstack = 0;
198     np->state = UNUSED;
199     return -1;
200   }
201   np->sz = curproc->sz;
202   np->parent = curproc;
203   *np->tf = *curproc->tf;
204   np->priority = curproc->priority; //HW#4
205
206   // Clear %eax so that fork returns 0 in the child.
207   np->tf->eax = 0;
208
209   for(i = 0; i < NOFILE; i++)
210     if(curproc->ofile[i])
211       np->ofile[i] = filedup(curproc->ofile[i]);
212   np->cwd = idup(curproc->cwd);
213
214   safestrcpy(np->name, curproc->name, sizeof(curproc->name));
215
216   pid = np->pid;
217
218   acquire(&ptable.lock);
219
220   np->state = RUNNABLE;
221
222   release(&ptable.lock);
223
224   return pid;
225 }
226
                                                    226,0-1       31%
```

proc.c (~/Desktop/2021_Operating...entSysCall/xv6_ssu_syscall) - VIM

File Edit View Search Terminal Help

```
120 // Set up first user process.
121 void
122 userinit(void)
123 {
124   struct proc *p;
125   extern char _binary_initcode_start[], _binary_initcode_size[];
126
127   p = allocproc();
128
129   initproc = p;
130   if((p->pgdir = setupkvm()) == 0)
131     panic("userinit: out of memory?");
132   inituvm(p->pgdir, _binary_initcode_start, (int)_binary_initcode_size);
133   p->sz = PGSIZE;
134   memset(p->tf, 0, sizeof(*p->tf));
135   p->tf->cs = (SEG_UCODE << 3) | DPL_USER;
136   p->tf->ds = (SEG_UDATA << 3) | DPL_USER;
137   p->tf->es = p->tf->ds;
138   p->tf->ss = p->tf->ds;
139   p->tf->eflags = FL_IF;
140   p->tf->esp = PGSIZE;
141   p->tf->eip = 0;  // beginning of initcode.S
142
143   safestrcpy(p->name, "initcode", sizeof(p->name));
144   p->cwd = namei("/");
145
146   // this assignment to p->state lets other cores
147   // run this process. the acquire forces the above
148   // writes to be visible, and the lock is also needed
149   // because the assignment might not be atomic.
150   acquire(&ptable.lock);
151
152   p->state = RUNNABLE;
153   p->priority = 5;  //HW#4
154
155   release(&ptable.lock);
156 }
157
                                                    120,1         20%
```

proc.c

allocproc() 에서 프로세스 새로 할당시 runtime을 0으로 초기화 시킨다

userinit() 에서 init 프로세스 생성시 priority를 5로 설정한다.

fork() 에서 child 생성시 parent의 priority를 복사한다.

getnice.c (~/Desktop/2021_Operat...sCall/xv6_ssu_syscall/user) - VIM

File Edit View Search Terminal Help

```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc, char **argv)
{
    int result = 0;
    int pid = 0;

  if (argc < 2)
  {
    printf(2, "usage: getnice pid ...\n");
    exit();
  }
pid = atoi(argv[1]);
  result = getnice(atoi(argv[1]));

  if(result < 0)
  {
      printf(2, "non-existing process\n");
  }
  else
  {
      printf(2, "pid : %d nice : %d\n", pid, result);
  }
  exit();
}
```

1,1 All

user/getnice.c 시스템 콜이 잘 되는지 시험하기 위한 파일. 인자가 잘 들어오는지 확인 후 시스템 콜을 출력한다. 호출 결과가 0 미만이면 에러 문장출력, 호출결과가 0 이상이면 찾으려 했던 pid와 nice 값을 출력

setnice.c (~/Desktop/2021_Operat...sCall/xv6_ssu_syscall/user) - VIM

File Edit View Search Terminal Help

```c
#include "types.h"
#include "stat.h"
#include "user.h"

int main(int argc, char **argv)
{
    int result =0;
    int pid = 0;
    int nice = 0;
   if (argc < 3)
  {
    printf(2, "usage: setnice pid nice ...\n");
    exit();
  }
   pid = atoi(argv[1]);
   nice = atoi(argv[2]);
  result = setnice(pid, nice);

  if(result < 0)
  {
      printf(2, "non-existing process\n");
  }
  else
  {
      printf(2, "set pid:%d nice to %d\n", pid, nice);
  }
  exit();
}
```

1,1 All

user/setnice.c 시스템 콜이 잘 되는지 시험하기 위한 파일. 인자가 잘 들어오는지 확인 후 시스템 콜을 출력한다. 호출 결과가 0 미만이면 에러 문장출력, 호출결과가 0 이상이면 설정하려 했던 pid와 수정된 nice 값을 출력

3.결과

(테스트 실행결과를 캡처)