운영체제 Project 1 Wiki

정상윤
2016025032

1. getppid 시스템콜 구현

• 디자인, 구현

영역: proc.c proc.h sysproc.c

```
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)
};
```

proc.h 에 정의된 proc (프로세스) 구조체. 자신의 id인 pid(int형)와 부모 프로세스를 가르키는 parent(또다른 proc) 가 있음.

```
struct proc*
myproc(void) {
  struct cpu *c;
  struct proc *p;
  pushcli();
  c = mycpu();
  p = c->proc;
  popcli();
  return p;
}
```

proc.c 에 정의된 myproc()를 통해 현재 cpu에서 실행 중인 process를 얻을 수 있음. 그 외에 fork(), wait() 등 process 관련 함수들이 정의돼있음. 그 함수들의 래퍼 함수들은 sysproc.c에 정의됨. 예) sys_fork(), sys_wait()

```
39 int
40 sys_getpid(void)
41 {
42   return myproc()->pid;
43 }
44
45 //get parent pid
46 int
47 sys_getppid(void)
48 {
49     return myproc()->parent->pid;
50 }
51
```

sysproc.c 에 자신의 id를 가져오는 sys_getpid()가 있음. myproc()를 호출하여 자기 자신 process를 return한 다음 “-> pid” 로 접근하여 id를 return함. 이를 통해 getppid()를 어떻게 구현해야할지 감이 잡힘. 굉장히 비슷한 동작이기 때문에 바로 구현해줬다.

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

int
main(int argc, char *argv[])
{
    int pid = getpid();
    int p_pid = getppid();
    printf(1, "My pid is %d\n", pid);
    printf(1, "My ppid is %d\n", p_pid);
    exit();
}
```

유저프로그램 project01_1.c 코드.

- 실행결과

```
jsy@jsy-VirtualBox:~/xv6-public$ ./bootxv6.sh
WARNING: Image format was not specified for 'fs.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'xv6.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ project01_1
My pid is 3
My ppid is 2
$ project01_1
My pid is 4
My ppid is 2
```

실행할 때마다 pid가 1 씩 증가하는 것을 확인함.

- 트러블슈팅

```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie
-c -o project01_1.o project01_1.c
In file included from sysproc.c:3:0,
                 from project01_1.c:4:
defs.h:107:17: error: conflicting types for ‘exit’
 void          exit(void);
               ^
In file included from project01_1.c:3:0:
user.h:6:5: note: previous declaration of ‘exit’ was here
 int exit(void) __attribute__((noreturn));
     ^
In file included from sysproc.c:3:0,
                 from project01_1.c:4:
defs.h:118:17: error: conflicting types for ‘sleep’
 void          sleep(void*, struct spinlock*);
               ^
In file included from project01_1.c:3:0:
user.h:24:5: note: previous declaration of ‘sleep’ was here
 int sleep(int);
     ^
In file included from sysproc.c:3:0,
                 from project01_1.c:4:
defs.h:144:17: error: conflicting types for ‘memmove’
 void*         memmove(void*, const void*, uint);
               ^
In file included from project01_1.c:3:0:
user.h:32:7: note: previous declaration of ‘memmove’ was here
 void *memmove(void*, const void*, int);
       ^
In file included from sysproc.c:3:0,
                 from project01_1.c:4:
defs.h:147:17: error: conflicting types for ‘strlen’
 int           strlen(const char*);
               ^
In file included from project01_1.c:3:0:
user.h:37:6: note: previous declaration of ‘strlen’ was here
 uint strlen(const char*);
      ^
<builtin>: recipe for target 'project01_1.o' failed
make: *** [project01_1.o] Error 1
```

유저 프로그램에서 getpid()를 써야하니 sysproc.c 를 include 해줘야한다는 착각을 하고 넣어줬다가 중복 선언 때문에 에러가 뜬 모습이다. 사용할 시스템콜들은 이미 user.h에 있다는 것을 깨닫고 sysproc.c를 빼주니 정상적으로 작동했다. 헤더 파일의 중요성에 대해서 깨달았다.

2. Interrupt 128 구현

• 디자인, 구현

영역: trap.c traps.h vectors.S trapasm.S

```
 1 #include "types.h"
 2 #include "stat.h"
 3 #include "user.h"
 4
 5 int
 6 main(int argc, char *argv[])
 7 {
 8       __asm__("int $128");
 9       return 0;
10 }
```

유저 프로그램 project01_2.c 에서 __asm__("int $128") 를 통해 interrupt를 발생시킨다. 그러면 vectors.S에서 alltraps 로 jump하면 trapasm.S로 넘어가 레지스터 영역 처리를 한 후 trap을 호출한다. 그러면 trap.c로 넘어가 나머지 trap관련 처리들을 해준다.

```
void
tvinit(void)
{
  int i;

  for(i = 0; i < 256; i++){
    //128 interrupt can be called in user mode
    if(i == 128){
        SETGATE(idt[i], 1, SEG_KCODE<<3, vectors[i], DPL_USER);
    }
    else SETGATE(idt[i], 0, SEG_KCODE<<3, vectors[i], 0);
  }
  SETGATE(idt[T_SYSCALL], 1, SEG_KCODE<<3, vectors[T_SYSCALL], DPL_USER);

  initlock(&tickslock, "time");
}
```

trap.c 안 tvinit 함수다. 이 곳에서 각 interrupt에 대한 초기화를 해주는데 128번째 interrupt도 syscall과 마찬가지로 user mode에서 실행될 수 있게 해야하기 때문에 SETGATE 인자들을 syscall과 똑같이 줬다.

```
41 void
42 trap(struct trapframe *tf)
43 {
44   if(tf->trapno == T_SYSCALL){
45     if(myproc()->killed)
46       exit();
47     myproc()->tf = tf;
48     syscall();
49     if(myproc()->killed)
50       exit();
51     return;
52   }
53
54   //interrupt 128
55   if(tf->trapno == 128){
56     cprintf("user interrupt 128 called!\n");
57     exit();
58   }
```

그 밑에 n번째 trap들이 어떤 동작을 하는지 trap()에서 설정해준다. trapframe의 trapno가 128일 때 cprintf로 “user interrupt 128 called!”를 출력해준다.

• 실행 결과

```
jsy@jsy-VirtualBox:~/xv6-public$ ./bootxv6.sh
WARNING: Image format was not specified for 'fs.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operati
         Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'xv6.img' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operati
         Specify the 'raw' format explicitly to remove the restrictions.
xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ project01_2
user interrupt 128 called!
$ project01_2
user interrupt 128 called!
```

여러번 실행해줘도 정상적으로 동작한다.

• 트러블슈팅

처음에 SETGATE를 통해 user mode에서도 실행 가능하게 설정해줘야 한다는 부분을 놓쳐서 계속 error exception이 발생했다. 그 부분 말고는 어려운 부분은 없었다.